DECENTRALIZED DATA FUSION AND DATA HARVESTING FRAMEWORK FOR HETEROGENEOUS, DYNAMIC NETWORK SYSTEMS

DISSERTATION
DOKTOR-INGENIEUR (DR.-ING.)

VORGELEGT DER
FAKULTÄT FÜR INFORMATIK UND AUTOMATISIERUNG
DER TECHNISCHEN UNIVERSITÄT ILMENAU

VON
M.SC. -INF. JIN YANG

1. PROF. DR.-ING. HABIL. KAI-UWE SATTLER
2. PROF. DR. RER.NAT. JOCHEN SEITZ
3. PROF. DR. LIPYEOW LIM

urn:nbn:de:gbv:ilm1-2014000633

Abstract

With the increasing number of available smart phones, sensor nodes, and novel mobile smart devices such as Google glass, a large volume of data reflecting the environment is generated in the form of sensing data sources (such as GPS, received signal strength identification, accelerometer, microphone, images, videos and gyroscope, etc.).Some context-aware and data centric applications require the online processing of the data collected. Such data-centric applications include activity recognition, health monitoring or indoor location tracking as basic application; or relatively complex applications such as disaster management, change detection, mobile advertisement or meeting planner. Along with this unavoidable trend of data-centric applications, several challenges are to be faced:

- *The demands and requirements of data-centric applications are various. The underlay heterogeneous devices have distinct capabilities and various wireless communication interfaces. How to support various data-centric applications based on such platforms is a tough question.*
- *The energy consumption for wireless communication and other resource limitation form a headache for these applications.*
- *Although various frameworks exist, they suffer from several common problems, such as the conflicts between contexts of distributed computing and the resource limitation on the devices, the trade-offs between the degree of application specificity and the middleware generality, the reusable components and flexible configuration of the middleware framework.*

The thesis researches on the decentralized data fusion and data harvesting framework for heterogeneous dynamic network system consisting of various devices with resource constraints. In order to achieve the flexible design, a general architecture is provided while the detailed data fusion and data exchange functions can be dynamically configured. A novel method to use directed fusion graph to model the logical structure of the distributed information fusion architecture is introduced. This directed fusion graph can accurately portray the interconnection among different data fusion components and the data exchange protocols, as well as the detailed data streams. The directed fusion graph is then transformed into a format with marked language, so that both human and machine can easily understand and edit.

In the field of data exchange protocols, this thesis targets energy-efficiency considering the resource constraints of the devices and robustness, as the dynamic environment might cause failures to the system. It proposes a refined gossip strategy to reduce retransmission of redundant data. The thesis also suggests a design guideline to achieve different design aims for different applications. These results in this field can be integrated into the framework effortlessly.

The configuration mechanism is another feature of this framework. Different from other research work which consider configuration as a post-design work separated from the main design of any middle-ware. This thesis considers the configuration part as another dimension of the framework. The whole strategy in configuration sets up the foundation for the flexible architecture, and makes it easy to adapt to the dynamic environment.

The contributions in the above fields lead to a light-weight data fusion and data harvesting framework which can be deployed easily above wireless based, heterogeneous, dynamic network systems, even in extreme conditions, to handle data-centric applications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the increasing number of available smart phones, sensor nodes and other new mobile devices (for instance, Google glass), the volume of data reflecting the environment is generated in the form of sensing data sources (such as GPS, received signal strength identification, accelerometer, microphone, images, videos and gyroscope, etc.). Some context-aware and data centric applications require the on-line processing of the data collected. Such data-centric applications include activity recognition, health monitoring or indoor location tracking as basic application; or relatively complex applications such as disaster management, change detection, mobile advertisement or meeting planner. These require "in-network", collaborative processing of the data collected from different devices. The data collected from the devices is exchanged through wireless communication interfaces among the network. Wireless sensor network, and mobile communication networks (working either in peer to peer mode or work in client/server mode) are two examples of such networks.

The techniques that support any data-centric applications require general data fusion, data harvesting and data exchange protocols to acquire data from the sensing network, and to feed queries into the network. This scheme connects the data sources and the information demander(s) with some information fusion components.

It can fuse and process data while exchanging and processing data among nodes. These applications are fundamental for complex information extracting, decision-making, and support of other applications in information systems. The data exchange protocols and the data fusion components are combined to construct a service overlay for data fusion and data harvesting at the application layer.

According to [16], data fusion is the process of integration of multiple data and knowledge representing the same real-world object into a consistent, accurate, and useful representation. Data fusion processes are often categorized as low, intermediate or high, depending on the processing stage at which fusion takes place. Low level data fusion combines several sources of raw data to produce new raw data. The expectation is that fused data is more informative and synthetic than the original inputs. For example, sensor fusion is also known as (multi-sensor) data fusion and is a subset of information fusion.

Data harvesting, is the process of extracting hidden knowledge from large volumes of raw data and using it to make crucial business decisions. Data harvesting includes the concept of collecting the raw

data from the initial data sources, aggregating these initial data node by node to extract detailed decisions, and exchanging the middle results or final results among all the other nodes. In this thesis, the data harvesting denotes the combination of the processes of data collection, aggregation and data exchange, which is beneficial to conclude a final decision.

The novel data centric applications above heterogeneous sensing network systems under dynamic environment motivate this thesis. The heterogeneity feature not only comes from the device itself, but also comes from the various wireless communication interfaces, and the diversity of the sensing data itself. Constructing a data fusion and data harvesting framework at the application layer helps provide a unified interface for data-centric applications and hide the complexity of underlay caused by heterogeneity and mobility. Furthermore it is possible to optimize the utility of the resource-limited devices at this overlay. The advantage of such framework increases with the increasing popularity of smart phones and the demands in mobile pervasive computing.

The next section goes into details some of the applications and use cases. Through analyzing these applications, a clear understanding can be achieved regarding the design requirements and the challenges of the framework.

1.2 Application Scenarios and Use Cases

This section lists three various applications to show the scenarios which motivates the data-fusion and data-harvesting framework.

1.2.1 Disaster or Emergency Management and Resource Map Generation

In large-scale disasters such earthquakes or floods, the construction of a reliable temporary communication infrastructure is essential for first-responders. Through setting up this temporary communication infrastructure inside the disaster areas and monitoring the disaster area, some up-to-date status regarding the damages caused by disasters can be automatically transmitted outside the disaster area without human efforts. The existence of such infrastructure is critical and helpful, especially in some risky areas for human safety.

One example of the usage of such infrastructure is the Japan Fukushima Daiichi nuclear disasters, which occurred in March 2011. Several reports analyzed the reasons of this disaster. For example, as ' "Understanding Japan's Nuclear Crisis" [39] reported during and after the disaster: "At best, even those present at the site have a limited view of what is going on inside the reactors themselves, and the situation has changed rapidly over the last several days. Meanwhile, the terminology involved is somewhat confusing: some fuel rods have almost certainly melted, but we have not seen a meltdown; radioactive material has been released from the reactors, but the radioactive fuel currently remains contained. An additional threat has recently become apparent, as one of the inactive reactors at the site suffered from an explosion and fire in the area where its fuel is being stored. **There is almost no information available about how the tsunami affected the stored fuel**."

One of the major obstacles is the lack of up-to-date information in the disaster areas. Considering the extremely life-threatening environment, it is not possible to send in people to carry out manual data collections. This situation leads to difficulty in disaster management, especially in the decision making

concerning controlling, recovering, and rescuing processes. Hence, the situation caused by the crisis deteriorates to the worst without possible effective actions.

To acquire and process the information under extremely dangerous situation is critical for a human to take actions. This thesis suggests using a self-organized data fusion and data-harvesting framework above heterogeneous devices to solve the problem. The reason to adopt heterogeneous devices is that in such disaster/emergency scenarios, the participation of more devices leads to collection of different types of data, which could result in a thorough understanding of the disaster scenarios. For example, in the nuclear crisis example, the heterogeneous sensors may collaborate in the following way:

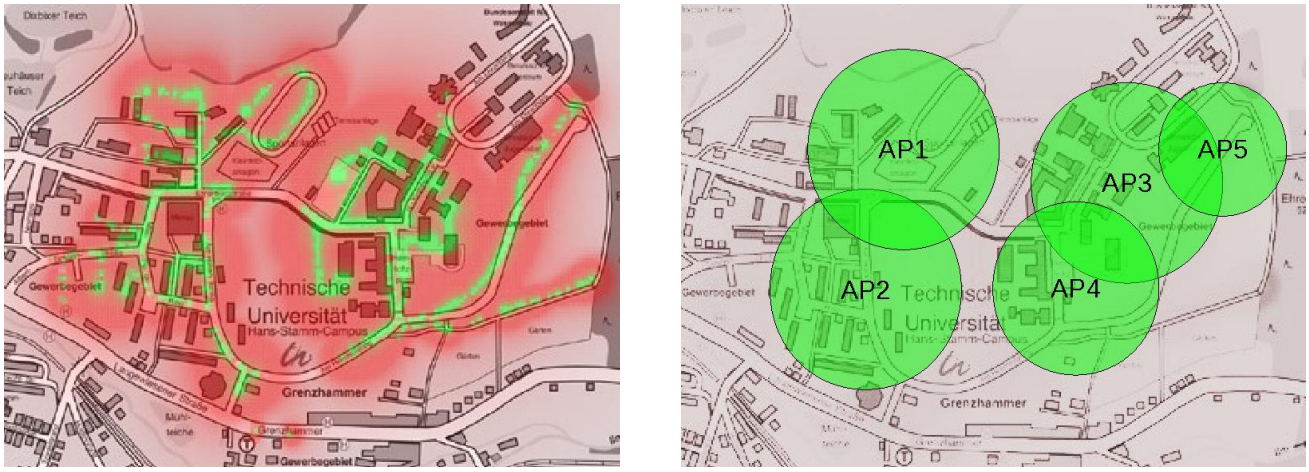
- Some **wireless sensor nodes** monitor the temperature and measure chemicals emissions, making it possible to detect from the interior fuel where there could be an area with high possibility of meltdown.
- While other **cameras** already deployed inside the fuel containment vessel could record the actual situations and transmit the videos or images through wireless communication network.
- **Multi-copters** may interconnect with the data interfaces of the deployed wireless sensor nodes within the disaster areas. They can also provide sensing and measurement data from the outside of the fuel. Or they can function as network relays to recover the communication infrastructure, which was partitioned by the damages of the disasters.

Such communication network consist of heterogeneous devices make it possible to provide measurement data from inside of the equipment reflecting the inside environment. Thus the information collected can be combined to form the entire picture of the disaster area.

At the application layer, all these heterogeneous devices cooperate to collect and send the data out to the decision makers or, in the other way, send the queries in and return the replies out. The information can be collected from the disaster area, which benefits further decision-making and rescuing process.

This application focuses on reconstructing/augmenting existing heterogeneous networks since this approach allows utilizing the remaining infrastructure even under disaster scenarios. In order to automatically repair a damaged communication infrastructure, the locations and other parameters such as network load, spectrum opportunities and the installed services of still operational resources need to be determined by a self-organizing network of both stable embedded sensors and mobile sensors. It is useful to store this information in a robust, distributed resource map. The construction process of this map utilizes several measurements, data fusion and data dissemination steps. With a resource map as figure 1.1(b), which is constructed from measurement data similar to the one depicted in figure 1.1(a), it is possible to reconstruct the network by deploying additional network resources.

In forming the global view of resources availability based on a partial view, data harvesting and data fusion techniques can be adopted by the devices listed above in a peer-to-peer mode. One of the most important challenges is that the map generation process should be robust enough to tolerate the failures caused by disasters. Another requirement in the disaster/emergency management scenario, is that, the data fusion and data harvesting should be combined together to reduce the energy consumption as much as possible. Thus, utilizing robust information exchange protocols, and generating redundancy in mitigating the interference of faults is necessary to provide reliability and fault tolerance features. Such basic requirements should motivate a common decentralized data fusion and data-harvesting framework above heterogeneous network system.



(a) Sensor Measurement Data Map (b) Radio Resource Map

Figure 1.1: From Radio Measurement Data to Access Points Resource Map, Figures From [73]

1.2.2 Decentralized Change Detection

Following the application of the disaster monitoring and management is the application of change detection. Effective change detection in a sensitive area contributes to the discovering of the occurrence of abnormal events. Data fusion and data harvesting framework can also be deployed to monitor such areas and trigger disaster warnings. This application originates from wireless sensor networks. Decentralized change detection above heterogeneous sensing network system brings up new opportunities and challenges to this problem.

Change detection is the process of identifying differences in the state of an object or phenomenon by observing it at different times [78]. Designing and developing a change detector in a heterogeneous sensing network posts the following main challenges:

- First, the data streams produced within networks must be processed in real time, which is a special case of change detection in distributed data streams.
- Second, processing data streams on some devices has to take into account resource limitations of some nodes which could be bottleneck of the system. These limitations may include low power batteries, small memory, and limited communication resources.
- Third, effective schemes should be adopted to overcome local detection failures and improve the detection accuracy. Last by not least, different applications have different requirements, for example, a network used for monitoring and alerting warnings of fire requires the warning of fire reported immediately to multiple event monitoring location in the network. It is a multi-source multi-sink problem with time constraints.

In facing all the challenges, a decentralized detection framework is necessary on the heterogeneous network system to enable in-network data processing. It can combine local change detection models and global decision fusion models. The nodes near the event locations could have higher detection accuracy

than those nodes, which are far away from the event locations. Therefore it is critical to reach a decision regarding the occurrence of the events among the nodes within the range near the event locations. And then the decisions can be forwarded to the sinks. This requires the in-network change detection algorithms be implemented on the nodes.

To enable an in-network change detection has two advantages: first, the communication cost could be reduced, due to the fact that only the actual detected events and the resulting warning messages should be transmitted from the event neighborhood to the sinks, instead of all the nodes of the network transmitting the original measurement data to the sinks. Second, the detection accuracy can be improved by utilizing the global data fusion algorithms near the event locations to explore the spatial correlations among local detection results.

The decentralized change detection framework relates to the following problems:

- First, the local change detection algorithm provides basic conclusions on event locations and change contents.
- Second, the decision fusion model fuses the local results and generates a global decision concerning the event with improvement on report accuracy. These two can be regarded as two major fusion functions. In addition, the data communication protocol is in charge of transmitting the decisions from the information sources to the information sinks.
- Third, the detection accuracy requirements should be met while not introducing too much extra communication cost. This can be solved by choosing proper communication protocols.
- These demands the fusion functions be connected through proper data exchange protocols to form a data processing and exchange topology. Thus flexibility in configuration is the key in deciding the actual topology of such framework.

This decentralized change detection framework is one of the special use cases of the decentralized data fusion and data-harvesting framework. As mentioned, the fusion functions in this application are both the local change detection algorithm and the global decision fusion function. The local change detection algorithm explores the time correlation of the measurement data. Whereas the global decision fusion algorithm can explore the spatial correlation of the change detection results from several sensor nodes. The data exchange among different fusion functions in sensor nodes should be specified and configured to benefit energy efficiency. Such a framework for change detection applications in the heterogeneous network system is novel with the following major features and contributions:

- The framework consists of three parts: the local change detection models, the global decision fusion models, and the information exchange protocols. These three parts can be integrated together as plug-ins to provide flexibility to the framework according to different scenarios.
- An incremental change detection algorithm that uses the incremental approach to computing the DFT(Discrete Fourier Transform) coefficients can be adapted. In this way, the local change detector is updated to meet the requirements for on-line processing of several sensor data streams.

- With the decision models, it is possible to fuse and achieve consensus of the event and its location, and generate a decision of warning or alarm with event location information. Various information exchange protocols can be adopted to transmit the messages on local change detection or the decision fusion results. Further it can be demonstrated that the flexibility in protocols results in the achievement of different design aims such as fault tolerance or reduction in communication cost.

1.2.3 Location Related Mobile Query Fusion

Smart phones or other novel generation of mobile devices (for instance, Google glass, iwatch, etc) can be used to sense the surrounding environment. The sensed information on environment can be combined with the device owners' current status to provide context related sensing information. An increasing number of context-aware mobile computing applications involve the processing of continuous queries over data streams generated by a variety of smart phone-embedded sensors (such as GPS, video recorder, camera, accelerometer, microphone and gyroscope). A majority of the initial applications, in areas such as activity recognition [55], health monitoring [74] or indoor location tracking, apply such query processing on an *individualized* basis, using data solely from an individual's personal smart phone-embedded or wearable sensors. Lately, however, a more sophisticated mobile computing paradigm is emerging that revolves around *collaborative sensing of shared context*:

Many emerging mobile computing applications, both in consumer and enterprise domains, require not just an individual's sensor data, but *collective context* obtained by processing sensor streams from a *group of mobile devices*, as well as cloud-based data sources. This represents the novel mobile pervasive computing scheme. This section describes two such representative use cases, then extracts relevant insight, and brings about the design requirements for such query fusion framework.

Example 1 ***Meeting Status Indicator:*** *In enterprises, participants in a meeting may be running an application on their phones that continually tracks the meeting's progress and automatically detects when the meeting has ended (i.e. when a single participant steps out briefly to take a phone call). The 'meeting ended' query may need to monitor:*

- a) *the location and current physical activity (sitting, standing) (using Wi-Fi and accelerometer data) of all the participants*
- b) *the light and sound levels of the room and*
- c) *the status of the projector and conferencing phone in the room, and combine multiple predicates into a single complex query, such as:*

"ALERT when' majority of participants stand up' AND ('room lights go on' OR 'room sound levels decrease') AND 'projector is switched off' ".

In this case, we can identify the following characteristics:

- *a)* each phone needs access to the 'activity' state of the other participants (obtained from their phone's accelerometer sensor)

- *b)* the light and sound levels are ambient environmental state that may be sensed by one smart phone and shared with the other smart phones, and
- *c)* the projector/phone status may need to be obtained from an infrastructure-based source (e.g., an enterprise Presence Server).

Example 2 ***Emergency Triage Management :*** *As a much more data-intensive application, consider an emergency management scenario of the future where individual victims in a disaster area (e.g., a building) have their location and other medical parameters, such as Sp02 (Blood Oxygen Saturation) and heart rate, being monitored by their individual smart phones (which may be locally connected to body-worn sensor devices). As rescue professionals move through the disaster area, each of them may choose to execute slightly different continuous queries, corresponding to their specific field of expertise. For example, we simply consider two rescuers, located near each other, who have the following two queries:*

Q1 *(Rescuer 1): "Track the nearest 3 victims' locations, such that each victim's SpO2* $< 95\%$ *AND heart rate* $> 120/min$*"*

Q2 *(Rescuer 2): "Track the nearest 3 victims' locations, such that heart rate*$> 95/min$ *and blood pressure* $> 150mmHg$*. "*

Given the rescuers' proximity to each other, it is easy to identify the following characteristics:

- *a)* Each phone may need to monitor data from a non-identical, but *overlapping* set of victims' smart phones;
- *b)* If a particular victim is relevant to both Q1 and Q2, then, while the query predicates related to 'heart rate' values are not identical, they may be effectively combined within a *common processing logic*–i.e., if the heart rate is $< 95/min$, it is automatically $< 120/min$ as well. Thus, unlike the case of 'ambient sound/light levels' in Example 1 (where the predicate is identical across multiple smart phones), Example 2 demonstrates the possibility of having a query predicate be *partially* evaluated on a mobile device, while the residual predicate evaluation is performed on another smart phone.

In both the above two pervasive mobile sensing applications, a potentially large number of physically-proximate smart phones simultaneously execute multiple continuous queries. Each operates on a collection of individual, shared and remote (cloud-based) sensor data sources. In particular, a single complex query, executing on a smart phone, can involve three distinct types of sensor sources:

a) *Mobile and Personal*: Here, the data streams are associated with sensors embedded on the local smart phone, representing the context that is *unique* to each specific individual; for instance, accelerometer data used to infer an individual's locomotive state.
b) *Mobile and Non-Personal*: While the data streams are still associated with smart phone-embedded sensors, unlike a), the context represented is now non-personal and thus can be inferred by utilizing sensor data from one or more alternate mobile devices for instance, a Web service that provides up-to-date temperature and pollutant concentrations in different urban areas.

While complexity comes from the various sources the network should process, special flexibility comes from the fact that the smart phone or mobile devices usually embed several various wireless communication interfaces (Bluetooth™, 3G/4G, WiFi). These wireless communication interfaces provide the possibility of constructing different paradigms of communication modes (peer to peer among phones, or client/server in between phones and servers) in a flexible way. The energy consumption issue can benefit from this based on the following observations:

- The energy overheads of wireless communication typically dominate the energy costs of on-board sensing (with GPS being a notable exception)–as reported in [80], wireless data transmission and reception typically consumes more than 21.5% of the energy budget on current smart phones (with displays being the other dominant energy consumer).
- Smart phones or new generation of mobile devices today are routinely equipped with multiple wireless communication interfaces (e.g., 3G, Wi-Fi and Bluetooth). Moreover, short-range radio interfaces (e.g., Bluetooth) consume significantly lower energy (in terms of Joules per bit transmitted) than long-range 3G/4G interfaces.

As the trend of such applications on mobile sensing networks becomes obvious and pervasive, energy overheads continue to be the major bottleneck in the large-scale deployment of such continuous mobile sensing-based applications. For instance, continuous use of GPS data streams is known [29] to drain a smart phone's battery in 4-5 hours or less. This problem can be mitigated through dedicated design and control over the usage of the energy on wireless communications. This is because that the power consumed for transmission per bit is usually of several orders of magnitude of the power consumed per bit for local computation.

Despite the uniqueness of these applications at first glance, they share similarities in the requirements for the supporting middleware or components:

- The query processing algorithms can be regarded as fusion functions on incoming data streams directly from data sources or some middle results of query executions.
- The data exchange protocols should handle the dynamic and mobile environments, and provide reliability even when network partition occurs.
- The relative location for the query fusion function from the data sources and queries initiators may determine the energy consumption of the processing, considering both wireless communication cost and computational cost. Therefore, the configuration of the query fusion function with the data exchange protocols above the devices is another decisive factor in the problem.

By differentiating the usage of wireless interfaces (Bluetooth™, WiFi, 3G/4G) based on distances between the relative locations of query processing units, it is possible to optimize energy usage on smart phones to extend the life time of the data processing. This introduces both challenges and new opportunists to solve the problem.

1.2.4 Summary of Use Cases

The above use cases represent three distinct application directions where the data fusion and harvesting framework can be applied. The resource map generation process requires data processing on filtering low

quality measurement data, and this application emphasizes robust data exchange, since the application is deployed in a disaster sensitive area. The reliability of the application is a challenge and key to this application.

The decentralized change detection requires relatively high detection accuracy. The fusion function first explores the time correlation to figure out possible exceptions. Furthermore, space correlation of the measurement data near neighbor nodes is explored to improve the accuracy of detection. The complexity of fusion functions is more obvious in this application.

The collaborative query processing framework's focus is the optimization of energy consumption with consideration of the features provided by available wireless communication interfaces. The trade-off in between communication (data exchange) and computation (query processing) is the difficult part of this application.

Summing up the obvious distinctions of these applications does not affect the fact that all of them share the following common characteristics. These observations and insights on similarities set the foundation of the common requirements and constructing components of the framework described in the thesis:

- **Application aim:** the applications try to connect data sources with the information demander, through the network of data processing and data exchange from source toward demander. The data is processed to reply certain queries, or to be extracted, filtered, and abstracted according to the application requirements. Some general design aims should be considered for these applications, such as energy consumption, data accuracy, reliability, etc.

- **General structure:** regardless of the difference between the specific data processing measures and the format of data fusion functions, the applications can be abstract to be a framework which consists generally of three parts. These parts include the general functions of data processing, the data exchange protocols to connect the functions, and the configuration of these two into an entire data processing network.

- **Hardware condition:** the applications are deployed on hardware, which are resource limited, but need to be capable of data collection, data processing and data exchange through wireless interfaces.

- **Complex, dynamic environment:** the applications are carried out under complex environment, where the topology of underlying network might be affected either by faults caused by disasters or the mobility of the constructed nodes themselves.

After checking the common characteristics of the applications, the requirements can be extracted in building a general-purpose middle-ware framework. The framework provides the specified functions in supporting the data-centric applications similar to the use cases listed above. The following common requirements need to be satisfied:

- **Functional requirements:** from data sources, collecting, processing and delivering required data and effective results (middle results, or final decisions) to the data demander. This is the basic requirements extracted from the data centric applications list above.

- **Availability on heterogeneous devices:** the applications rely on collaboration of heterogeneous devices underlay. This requires the application to be able to run cross platform and handle heterogeneity of underlay wireless communication interfaces, and the variety of devices.

- **Handling resource limitation:** the devices used in these applications differ from each other in their formats. The common features of these devices are that, they are powered by battery, and have limited lifetime to contribute to the functional requirements. This feature limits the computation, and communication capacity provided by these devices. Thus it requires planning and optimized usage of the resources provided by the hardware devices, to yield maximized utility in achieving the functional requirements.

- **Reliability under environmental dynamics:** the dynamics of the environment may interference with the existing data processing and exchange framework by introducing some failures of hardware or software, or causing network partitions. The interference of the underlay changes should be transparent to the application. This should be achieved by the supporting middle-ware framework.

1.3 Challenges

The unique feature of wireless communication based networked systems leads to several unique challenges for the thesis to deal with:

1.3.1 Environment Dynamics

The environment changes may cause system dynamics, errors, or failures to occur. The mobility of the devices or the nature of wireless signal might easily cause the problem of packet drop, disconnection of links and partition of the network. These interferences should be taken into consideration by the framework in extreme work conditions. The thesis summarizes the following forms of failures:

- **System failure:** some extreme environment may lead to direct damages to the hardware or software of the system, such as device power off, inaccurate measurement from sensors, or other malfunction of the system.

- **Wireless communication related vulnerabilities:** the wireless signal is vulnerable to the interferences from the environment. Unstable wireless communication links is prone to packet drop, in the worst case the partition of the network.

- **Value failure of collected data:** under extreme conditions, the readings of the measurement from the sensors might deviate from the true measurement.

While these failures or vulnerabilities are unavoidable It is necessary to adopt some mechanisms in mitigating the above-mentioned failures or errors. Based on the above failures, the framework should be capable to:

- Tolerate the hardware or software failures. So the performance of the rest of the system does not suffer.

- Exchange data on unreliable wireless communication links, with low energy consumption.

- Overcome partial failures of value readings. The failed readings can be recognized by statistic algorithms or by exploring the correlation (pattern) among data.

As reliability is one chief design goals of the framework, this thesis describes some design choices to meet such robustness aims. In a later chapter we evaluate the performance of the system under different kinds of failure models.

1.3.2 Different Design Aims

Another major challenge is to prioritize different design aims. The following list summarizes the general design aims:

- **Functional aim:** support data centric applications enabling data exchange and various data fusion functions at the application layer.

- **Resource consumption:** Provide energy efficient solutions to resource-constrained nodes. The heterogeneous network consists of devices with variable capacities. While some extremely powerful base stations exist in the network, most of the nodes executing the measurement tasks are battery powered sensor nodes or smart phones. The data fusion and data-harvesting framework should adopt energy efficient strategies to meet the requirement of most of the resource limitation nodes.

- **Fault tolerance under extreme conditions:** The underlay network is not impervious to failures. The instability of the underlay network is due to large amount of heterogeneous, resource constrained devices, utilizing wireless communication to connect to one another. the data fusion and data harvesting frameworks at the application layer should overcome these interferences caused by underlay and deliver reliable data collection and fusion services.

- **Handle mobile nodes and the network partition caused by mobility:** Some smart phones nodes carrying out either the data measurement tasks or the data fusion tasks may move around. It causes constant changes in the topology of the network; in the worst case, network partitions might occur. The data fusion and data-harvesting framework is built on such relatively unstable underlay network consists of nodes with mobility. It should tolerate mobility and its influence and provides relatively stable services to the data centric applications.

- **Support different application and requirements:** The data fusion and data-harvesting framework should support data centric applications in general, instead of targeting certain types of specific applications.

- **Flexible configuration:** The decentralized data fusion applications consist of small data processing services or functions distributed on devices or nodes. These individual data fusion functions combine together to complete a data processing task. How to configure the interconnection of these data fusion functions on different nodes is a key problem. Different configurations of the interconnection of these fusion functions result in different outputs of the data fusion and harvesting framework. The framework should be easy to configure in order to meet different requirements of applications.

It can be figure out that not all design aims can be met at the same time. For example, there is a trade off in between the fault tolerance requirement and the energy efficiency requirement: under extreme conditions, the robustness often comes from redundancy, either by back-up systems (software or hardware), or retransmission of the lost messages. This contradicts the requirement of energy efficiency. It is necessary to prioritize these different design aims for different applications. The framework, in general, should support the easy combination of different design aims to meet specific requirements of applications, thus **flexible configuration of the interconnections of various fusion and exchange components** is one possible solution to guarantee that the framework is adaptable for different application requirements.

1.4 Research Questions and Approaches

Research Questions: In order to develop a viable framework supporting general data centric applications above heterogeneous dynamic network systems, the thesis need to address the following research questions:

- **How to model the data fusion and data harvesting framework?** The question is related with the following quesitons: What are the main requirements that the data fusion and data exchange framework should meet? How to extract the logic model and necessary components of this framework? How to balance the generality required from usability and the specialty from individual applications, in the design of the framework? How to achieve the general design aim of this framework while not losing the flexibility to satisfy specific application requirements?
  The thesis analyzes the existing and emerging applications, summarizes corresponding characteristics and requirements of these applications in this chapter. The system model, and the general architecture of the framework is introduced in Chapter 2. The following Chapter 3, Chapter 4, and Chapter 5 explain in detail the three components of this general framework separately.

- **How to guarantee the reliable data exchange in disaster scenarios?** The thesis researches on the following related questions: How to model the possible failures in disaster scenarios? How much energy is consumed by different data exchange protocols under different scenarios? How to trade-off the redundancy required to guarantee reliability with the energy efficiency constraints of the resource limit devices?
  The thesis tackles these problems by classifying the possible failure models under disaster scenarios. It then evaluates quantitatively the effect of applying different data dissemination protocols. In order to deal with the trade-off relationship between energy efficiency and reliability, experiments results are analyzed. Based on that work, Chapter 3 introduces gossip and refined gossip protocols to solve the reliability problem in data exchange. The result of this part of research and

analysis work leads to a recommendation table, which guides engineering design in using specific protocols under specific failure scenarios to meet special application requirements.

- **How to model specific data centric applications with common components?** How to decompose different applications? How to describe the model?
  Chapter 4 introduces a novel logical model denoted directed fusion graph to describe the data processing structure. The graph consists of fusion nodes and data exchange edges, which is easy to understand and scalable in adding or deleting fusions functions from the graph. This graph can be translated into configuration files for devices in the network to compile and generate the actual data fusion and data exchange architecture.

- **How to map the logic fusion structure which reflects the data centric application onto the software framework to achieve certain data processing aims?** How to configure the model?
  Chapter 5 focuses on flexible configuration and deployment of the data fusion components, and describes the resulting framework in details. This chapter introduces the format of the configuration files corresponding to the directed fusion graph, the software architecture on the devices to generate the data fusion and exchange framework. The centralized and decentralized data fusion architecture is illustrated together with the configuration file. And the processes of mapping the configuration file into actual data transmission sockets and data fusion components is described step by step in this chapter.

1.5 Contributions

The contribution of this thesis is three-folded. In general, the thesis provides a novel method to use directed fusion graph to model the logical structure of the distributed information fusion architecture. It also describes a middle-ware framework, which supports general-purpose data-centric applications on wireless mobile ad-hoc systems consisting of heterogeneous and resource-limitation devices. A flexible adaptive configuration mechanism is designed to meet the specific design requirements on data accuracy, dissemination speed and robustness, etc. To be able to trade-off and meet the different design goals for different applications is the feature of this work. This section summarizes the main results and contributions of this thesis.

- **The first contribution of this thesis is that it proposes a novel middleware framework, which is used to effectively support the data-centric applications on wireless mobile ad-hoc systems consisting of heterogeneous devices.** The framework, on the one hand, is necessary to support the general, common design requirements of such data-centric applications. On the other hand, it needs to provide flexibility to support the specific application requirements. In order to provide general support, the thesis analyzes and extracts the common characteristics and requirements of the applications. Based on the general common features of the applications, the framework explores the solution dimensions, to meet the general requirements:
  - the data fusion functions
  - the data exchange protocols, and
  - the configuration of interconnection of the data fusion functions

These three solution dimensions can be implemented as independent software plug-ins, integrated together through proper configurations. By combinating these three parts flexibly, it is possible to meet specific requirements from particular applications.

- **The second contribution of this thesis is that it provides a design recommendation to satisfy various application specific requirements, especially in terms of resource limitation, fault tolerance under disaster scenarios.** While the first contribution focus on the *general* support of the data-centric applications, this contribution is to solve the problem of making proper design decisions to adapt to *specific* application requirements:

  a) *Resource limitation* on the mobile devices is a severe problem, considering these devices (smart phones, and sensor nodes, etc.) are powered by batteries. This might hinder the redundancy design to guarantee robustness under disaster scenarios. In order to guarantee robustness under extreme conditions, data exchange protocols need to generate huge traffic loading from retransmission of redundant data messages, thus leading to quick depletion of batteries. The thesis explores the contradiction in between **resource limitation** and **robustness**, proposes to utilize target selective exchange based gossip protocols to reduce the unnecessary message retransmission. This leads to reduction to the waste of bandwidth and traffic load. Considering the fact that bandwidth is even scarcer in wireless communications under disaster scenarios, this enables the adoption of gossip protocols for wireless mobile ad-hoc systems consisting of devices powered by batteries. The evaluation results show that the reinforced gossip protocols can save 30% of the data processing time to reach the same data accuracy comparing with traditional gossip protocols. Further details regarding more evaluation results are described in Chapter 6.

  b) *Fault tolerance under extreme scenarios*: one of the contributions, regarding the adaption of the framework to specific applications, is to consider the interferences in extreme scenarios such as **disasters** to the wireless communication based distributed system. This thesis considers the possible complex system failure models triggered by unexpected outside environmental factors besides mobility. The resulting failure models include: spatial related nodes failure, wireless communication link related message or data loss, etc. Their interferences on these models to the decentralized data-centric applications are evaluated and analyzed. Based on these research and evaluation, the thesis analyzes the usage of suitable data exchange protocols to achieve relatively good performance gains in terms of reliability under these complex system dynamics. The results provide design suggestions to guide the corresponding adaption of the framework in such extreme scenarios.

- **The third contribution of this thesis is the logic model resulting from mapping data centric applications into *distributed directed fusion graph*, and the following description, possible self-configuration and deployment schemes of the logic model.** Different from the classical specific data fusion algorithms or techniques, the data fusion in this thesis is a general concept representing the component or service where different input data streams join at and get processed to generate the resulting data streams. This abstraction facilitates the general black box model as representation of fusion functions. The fusion component consists of several "gates" for incoming and outgoing data streams and the central fusion function processes these data streams from incoming gates and generates the outgoing result data streams. The interconnection of various fusion

components within one device or across different devices completes a specific data centric application. This network of fusion components performs automatic data processing and forwarding among different fusion components on heterogeneous devices. The novelty of this work is that:

a) This work focus on **decentralized** data fusion and data harvesting, where it provides various data stream connection models, including send/receive, request/reply, publish/subscribe, etc. This is different from the existing work, which focuses only on the orders of connection orders of computer procedures, regardless the specific connection models.

b) This work aims to provide a common framework across heterogeneous nodes including some extremely resource-limited devices such as sensor nodes or mobile phones. The graphical specification of the system configuration can specify the consumption of energy on the edges and provides conveniences for further optimization work. Besides, the configuration graph can be translated into easy-to-understand-and-maintain text description. The portability of the system blueprint provides possibility for the resource limited devices to join into the data processing.

These contributions result into a lightweight software framework, which supports the deployment of data centric applications above heterogeneous, dynamic networked systems. This framework has the following practical features:

- **It provides general support to data-centric applications while offering flexibility to adapt to specific applications.** This is achieved by choosing proper data fusion functions corresponding to the applications. For instance, in the use case of *Change detection 1.2.2*, the fusion functions are DFT (Discrete Fourier Transform) based local change detection algorithm, and global decision fusion functions, which can summarize the results of local detections. In the use case of *Location based query fusion 1.2.3*, the query processing, query optimization functions are the main fusions functions adopted.

- **The framework enables the adaptive topology of the data processing framework at upper overlay to process data streams according to underlay system dynamics.** The configuration method of the framework features the dynamic load and compilation of the system descriptions to generate a data stream connection sockets. These sockets form the general backbone pipes for the data streams to flow in the network system. The update of the logic configuration plan is extremely easy, by editing corresponding lines in text files. The underlay changes such as node failure or low link connectivity can be dealt with properly by choosing the routes and modes of the data streams. It is even possible to update the local system configuration plans on the devices and dynamically generate new data stream sockets at runtime.

1.6 Reader's Guide to the Thesis

The rest of the thesis is organized as follows: Chapter 2 classifies the existing research work on middleware frameworks for data processing above resource constraint devices. After comparing with the current research work, this section further introduces the challenges and the features that the novel framework

should contain. The following two sections of this chapter introduce the assumptions and the general system model.

The thesis provides a middle-ware framework, which consists of three parts in three dimensions of the solution space. The following three chapters are dedicated to describe the three dimensions of the framework. Chapter 3 discusses the robust data exchange protocols, with its focus on the influence of different failure models to wireless communication based data exchange. In Chapter 4, a novel model called directed fusion graph for describing distributed data fusion is introduced. How to use this directed fusion graph is illustrated with three use cases. Chapter 5 discusses the deployment and configuration problem of the framework. The software framework, which supports the flexible configuration, is described in detail in this chapter. It also includes the specific mapping steps from the directed fusion graph to actual fusion architecture. Chapter 6 reports the experiments results on evaluating the decentralized framework. Last but not least, Chapter 7 concludes the thesis and summarizes the future work in this research.

Chapter 2

Related Work and Problem Definition

The thesis locates on the interdisciplinary areas which cover information fusion, robust data exchange and configuration of the software frameworks above heterogeneous networked systems. Lots of related research work is in these three major directions. Chapter 3 introduces related work on robust data exchange protocols in a section, similarly Chapter 4 and Chapter 5 contain individual sections to discuss related work on information fusion, and configuration automation related work. This arrangement is to facilitate readers to catch the current status of these individual directions and compare the difference of this research with the existing work.

This chapter summarizes current work on the general software frameworks for decentralized data fusion and data processing. The aim is to provide readers with the general idea on the features and design principles of similar middleware or frameworks. To summarize the difference and similarity of this work with the thesis's work would benefit the understanding of the uniqueness of the thesis' work. The heterogeneous network system poses new challenges to middleware or software framework research. The traditional middleware techniques in distributed systems cannot be applied directly to our networked devices since they generally do not need to consider resource constraints, nor controls on wireless communications, etc. The rest of this section focuses on the survey of various existing middleware used for data management based on wireless sensor networks which may possibly be adopted in our scenarios.

2.1 Requirements and Classification of Related Work

This section articulates the challenges or features associated with middleware for data centric applications above the heterogeneous network systems. These features are important for software framework to be adopted on wireless communication based distributed data fusion systems. The section further discusses some related work, and classifies them based on these features.

- **Abstraction Support:** The heterogeneous communication networks consist of large number of different devices. The devices are developed by various vendors and may use different hardware platforms. Hiding the underlying hardware platforms to offer a homogeneous view of the network to support data centric applications is a major challenge for the systems.

Table 2.1: Features and Comparison of Various Middleware Framework

	Abstraction Support	Data Fusion	Resource Constraints	Adaptability	Scalability	Robust Data Exchange	Flexible Configuration
TinyDB	Y	Y	Y	N	Y	N	N
TinyCubus	Y	N	Y	Y	Y	N	Y
Impala	Y	N	Y	Y	Y	Y	N
Agilla	Y	Y	Y	N	Y	N	N
TinyLime	Y	Y	Y	N	Y	N	N
SINA	Y	N	N	N	Y	N	N
Cougar	Y	Y	Y	N	N	N	N
DSWare	Y	N	Y	Y	Y	N	N

- **Data Fusion:** The devices in the network systems are used to collect data from its surroundings. Data collected by various devices have to be merged or synthesized to form high level and easily understandable format or report. Also, communicating this synthesized information to the task issuer or the sink node (i.e. PDA, Laptop, Cell Phone, etc.) is another major challenge.
- **Resource Constraints:** The network system consists of tiny devices with very small memory, computation power, and battery power. Middleware for such devices have to be lightweight to work under limited resource availability.
- **Adaptability:** Middleware for heterogeneous networks must support algorithms that have adaptive performance. Adaptive fidelity algorithms have been developed for this purpose.
- **Scalability:** Middleware for heterogeneous network systems must be scalable enough in terms of number of nodes, number of users, etc. to operate over long periods of time.
- **Robust Data Exchange:** Data collection and data processing requires reliable data exchange to provide data streams from nodes nearby event locations for further processing. Considering the complex environment where the middleware works to collect and process data in distributed way, the data exchange protocol should be robust in all kinds of different scenarios or even under failure models.
- **Flexibility in Application Specific Configuration:** It is necessary for the middleware to support a wide variety of applications across the network. However, due to limited resource availability, the middleware cannot be generalized in this way. The middleware should provide enough possibility and flexibility, while the application specific aspects should be able to be configured and deployed runtime. Middleware with this feature is possible to be adopted broadly.

The following table 2.1 summarizes some existing software framework according to these requirements.

TinyDB [59] is a query processing middleware system for **TinyOS** based sensors. Given a query specifying the data interests, **TinyDB** collects that data from motes in the environment, filters it, aggregates it together, and routes it out to a PC. It is a representing system which can be used to process the data harvested from the wireless sensor nodes. **TinyDB** provides the abstraction support based on TinyOS. The data processing model provides power-efficient in-network query processing mechanisms for collecting data from individual sensor nodes. As an early work it considers energy efficiency of collecting

and fusing data, by reducing number of messages that must be sent. This results in extension of system life time. Besides, **TinyDB** has a good aggregation model. The problem with using it to directly support data-centric applications is that, it does not provide much functionality or complex fusion function support as part of middleware service. Most of the services have to be written in the applications running on top of it. Also the internal data aggregation and extraction structure cannot be adapted easily according to different application requirements. Robustness is not an important consideration for **TinyDB**, it was originally designed to be used in relatively stable environment, the problems caused by disasters or other unexpected reasons are not a consideration for this system.

TinyCubus [60] is a flexible, adaptive cross-layer framework which is implemented on top of TinyOS. The system considers the heterogeneous in terms of the hardware characteristics and application requirements even within a single network. **TinyCubus** considers flexibility and adaptation issues at the beginning of its design. It is divided into three parts: Tiny Cross-Layer Framework, Tiny Configuration Engine and Tiny Data Management Framework. This architecture can be used in different environments. The idea of making the framework flexible and adaptive to more applications while maintaining general functions is similar to the design aim of the thesis. However, the **TinyCubus** framework is not designed specifically to support decentralized data fusion tasks, thus the components it provides in the Tiny Data Management Framework are general. The functions are replication/caching, prefetching/hoarding, and simple aggregation functions. The decentralized data fusion network topology which represents the interconnection of fusion algorithm components is not a consideration of this framework. Besides, since the overhead from cross-layer design are huge, this framework is not suitable in some extreme environment where energy efficiency is a critical requirement. Further, robustness and energy consumption are not major design considerations for **TinyCubus**.

Agilla [25] is designed to develop multiple applications related to fire detection, and tracking, cargo container monitoring, etc. It is a mobile agent middleware using stack-based architecture. This framework enables mobile agents as special processes that can migrate across sensors. In order to save energy, its agent can be moved to bring computation closer to the data rather than transmitting data over unreliable wireless network. This idea to process the data near the data sources as close as possible is important and inspiring in meeting the requirements of energy critical applications. This feature, although it helps in energy saving, on the other hand, allows security vulnerabilities for hackers to inject malicious code into the agents. Besides, the programs are difficult to read and maintain. Thus the flexibility coming from enabling active in-network reprogramming is with a price which cannot easily be ignored.

Impala [53] is a middleware system to help perform long-term migration study of wildlife. **Impala** middleware was designed based on an event-based programming model with code modularity, ease of application adaptability, fault-tolerance, energy efficiency, and long deployment time in focus. It follows a finite state machine based approach taking into consideration various application parameters to handle the adaptability issue. [89]Although **Impala** has data communication support for getting the data back to the base station, it does not have any support for data fusion. Its abstraction model does not take the heterogeneity of the network into consideration and its application domain is rather simplistic.

There are other notable middleware for light-weight embedded devices and systems, such as Cougar [96], DSWare [51], SINA [76], EnviroTrack [1], Mires [83], and Hood [90]. We classify and compare the features of these related frameworks in the table 2.1. Most of the current middleware supporting the resource constraints are lacking support on the robust data exchange and the application specific configurations.

2.2 Challenges on Middleware for Data Processing

This section classifies the challenges in designing and implementing the middleware above heterogeneous dynamic network systems into three main types. The first type of the challenge comes from the conflicts between the contexts of distributed computing and the resource limited embedded devices. Distributed computing should support scalability, reliability, and availability and heterogeneity. This demands a careful design under the context of resource limited devices and adaption of dynamic network topology.

The second type of challenge come from the trade-offs between the degree of application specificity and middleware generality. It is important to integrate application knowledge into the services provided by the middleware because it can significantly improve the resource and energy efficiency and reduce the complexity for application development. However, since middleware is designed to support and optimize a broad range of applications, careful design decisions need to be explored between the degree of application-specific requirements and the generality of the middleware.

The third type of challenge is related with reusable components and flexible configuration of the middleware framework. Most of the current middleware have not considered how to integrate the components into generic middleware architecture to help developers match different requirements. Since many features need to be considered in the design of middleware, it is helpful to adopt reusable services. A generic framework with customizable component-based architecture is desirable. Component-based architecture has good support for dynamic configuration. It can utilize the component interface definition to provide standard service interfaces. The initialization of components can be dynamic, applied only when necessary. The relationship between components and services can be considered in the configuration steps to match the application requirements. However, until now, there is little study on reusable components and the flexible configuration for such middleware. This is one of the focuses of this thesis.

Another problem of current middleware research is that almost all the existing work on middleware has focused on systems of two extremes: either distributed systems or on the sensor networks. The various devices such as different sensor nodes, RFID, camera sensors, mobile sensing nodes, smart phones with sensors, and new generation of mobile sensing I/O devices such as Google glass, construct a complicated application environment for data-centric applications. This creates new challenges to this research, in order to deal with the more complex heterogeneity problems. It is necessary to consider more integration of low-level middleware and higher level pervasive computing middleware, and combine the function of information collection with the processing and use of information.

The framework discussed in this thesis is designed to solve the two problems listed above, with focus on architecture configuration flexibility using component based design to support different design aims in different environment. Also it is based on heterogeneous devices, with consideration on abstract data fusion and robust data exchange functions to serve data-centric applications. The combination of data exchange and data fusion helps solve the problem in dynamic topology and energy efficiency. Furthermore, it considers the heterogeneity of the wireless communication interfaces of heterogeneous network devices to optimize the energy usage. These features have not yet been adopted from existing research work, thus makes this framework novel in middleware research area.

Table 2.2: Problems and Corresponding Design Aims

Problems	Design aims
resource constraints	energy efficiency
various communication interfaces	optimization of communication measures
system dynamics	fault tolerance
mobility and network partition	transparent underlay to applications
unstable wireless communication links	robust data exchange
variety of data centric applications	general purpose platform
variety of fusion functions and interconnections	flexibility in configuration

2.3 Problem Description

The major problem this thesis solves is to support various type of data centric applications with resource constraints under a relatively dynamic environment, through decentralized data collecting and data processing (fusion) in-network. This thesis proposes to use a middleware framework installed on individual node of the network to collaborate in data fusion and data collection. Table 2.2 summarizes the problems and the corresponding design aims that this framework is supposed to achieve.

In order to support data centric applications at the application layer, a middleware which consists of some data fusion and data harvesting components should be set up and configured on the devices with constraints on resource availability. The different nodes are capable of using several different wireless communication interfaces to communicate with each other in different communication ranges. To meet different application specific aims, automatic data processing and data exchange should connect information source and information demander to carry out in-network data processing.

The problem has a broad number of application scenarios, and can be applied to various devices such as mobile devices and mobile Internet, mobile sensing data / mobile information demander /mobile information sources, or on sensor nodes, mobile phones, base stations, and laptops/desktops. The aim is to provide cross platform support for data centric applications. The applications can be based on different data exchange modes (client/server mobile/base station, p2p mobile phone to mobile phone, cluster based models). Specific application scenarios examples can be traditional environment monitoring/data fusion and aggregation, change detection, or query based application: mobile sensing applications, or the novel applications based on smart phone platforms.

From an optimization point of view, there are different cost models for individual applications. The framework should balance the different design aims to meet the specific cost requirements and enable the implementation to be adaptive easily. The main challenges this thesis comes across are application specific requirements such as how the general purpose framework adapts to specific application design requirements flexibly, while dealing with the resource constraints on the devices such as low wireless bandwidth, etc. Thus optimizing the resource usage is also one important topic in this thesis.

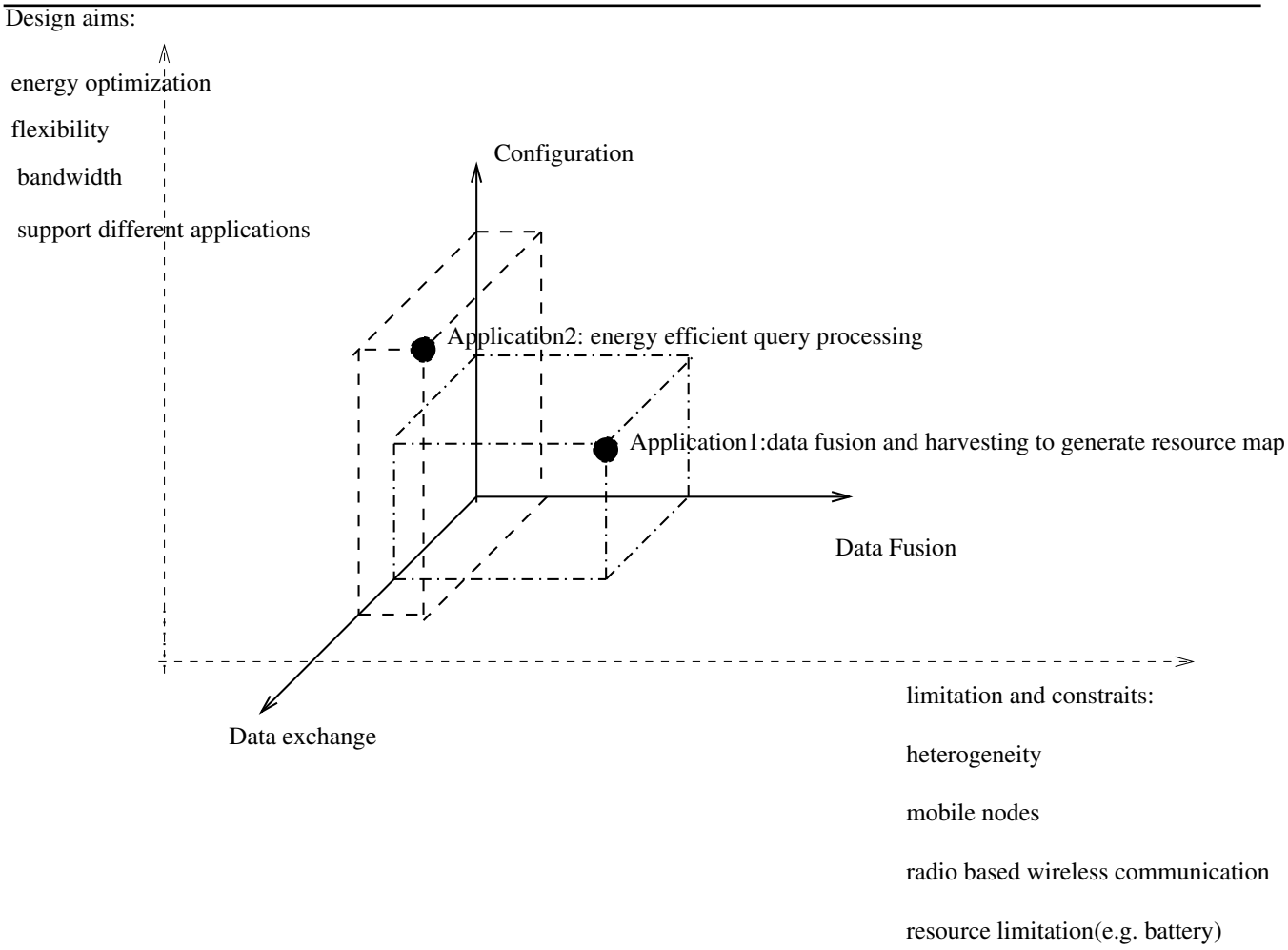


Figure 2.1: Problem Solution Space and Decentralized Data Fusion Framework

2.4 System Assumptions

We model the heterogeneous network system as a graph consisting of nodes and wireless communication links. Within each node, some resources (for instance: communication, computation, storage, caching, limited battery and wireless bandwidth) are available to support the different kinds of data fusion components. These data fusion components process input and output data according to certain applications. The data fusion components use the components of data communication protocols to control the usage of different wireless communication interfaces (Bluetooth^{TM}/WiFi /3G or 4G) to exchange data. The wireless communication interfaces can use different modes in data exchange, such as push/pull, send/receive. Each node also has resource limitations such as energy efficiency, computation ability, communication bandwidth, and storages, etc.

The system works in unstable states where either mobility or environment changes may lead to failures. The types of the failure models we considered are discussed in detail in Chapter 3, section 3.3.

The system model and its relations to the requirements and constrains is shown in fig.2.1. One coordi-

nate is the design aims of the framework. The other coordinate represents the limitations and constraints which comes from the real system or devices capacities. The framework, in order to meet the design aims with the constraints, should explore in the solution space consists of three dimensions of components:

- **Data exchange component:** support application layer robust information exchange protocols, such as flooding and gossip, control also the usage of different wireless radio interfaces such as Bluetooth™, wifi, or 3G/4G.

- **Data fusion component:** in charge of local data processing, output result data streams based on processing the input data streams, example fusion functions such as change detection, query processing, signal processing, etc.

- **Configuration component:** flexible configuration of the data fusion components interconnected by the data exchange components, which influences the system topology and service placement.

The data fusion components on different nodes are connected with each other through data streams specified by data exchange protocols, the interconnection of different components is specified by the configuration components. These three dimensions construct the solution space, where the combination of the solutions on each different dimension forms one solution to a certain problem. Any point inside the solution space is a specific solution to an application. For instance, in the application to generate resource map, the data fusion component uses location estimation fusion function (details in Chapter 4, Section 4.7.1), the data exchange component may use different gossip protocols (details in Chapter 3, Section 3.4) to achieve energy efficiency aim, or time constraints. The configuration component may configure the resource map generation topology to be a fusion tree to improve the aggregation and harvesting performance. Such specific solutions on each dimension can be combined to solve the problem of resource map generation.

In another application of energy efficient query processing, the data fusion component consists of functions in query processing and query optimization. The data exchange component is simple sending/receiving data streams among different data fusion components. The configuration component configures the interconnections of query processing and data sources, decides where the local queries should be processed.

From software engineering's point of view, these three components are naturally separate software components, which are combined as plug-ins to implement the software framework. Such implementation provides flexibility so that the framework can be adapt to system dynamics according to different applications. Some typical use cases are introduced in Chapter 1, section 1.2.1, section 1.2.2, and section 1.2.3.

Chapter 3

Robust Information Exchange for Wireless Communication based Distributed Systems

The data exchange protocols are used to exchange and collect data from wireless communication based distributed systems consisting of heterogeneous devices. We consider the problem of data exchange above heterogeneous devices at the application layer. Because the underlay topology of the physical network consisting of heterogeneous interfaces of wireless communication technologies, it will be helpful to maintain and guarantee the data communication at the overlay on the application layer.

Using data exchange protocols to deploy a data fusion overlay over heterogeneous dynamic distributed systems based on wireless communication poses several challenges:

- Algorithms for constructing data processing overlay in general do not consider the underlying topology of the physical network, as shown in [88]. However, when this system is deployed above wireless heterogeneous networked devices, a mismatch between the overlay and the physical topology leads to an inefficient usage of scarce wireless bandwidth resources.
- Energy efficiency is another problem for resource limited networked embedded systems, such as sensor nodes, mobile phones or smart phones. This can further influence the lifetime of the distributed data fusion system.
- Furthermore, device mobility may frequently change the physical routes that are used by the overlay connection. Intermittent connectivity may cause unstable data streams processed among peers in the overlay network. When we consider the disaster/emergency management scenarios, the damage to the underlay network may in the worst case cause the partition of the network.
- Last but not least, the wireless communication comparing with wired communication is unstable and can cause more failures due to the wireless link. The possible dynamics of the environment are also prone to various failures in hardware/software, and leads to changes of the underlay topology. How to guarantee the **robustness** of information exchange is a huge challenge.

These challenges that this thesis tries to deal with are difficult for traditional data exchange protocols to handle. This chapter tries the possibility of applying gossip protocols above wireless connections to carry out robust information exchange in heterogeneous dynamic network systems.

The following Section 3.1 first classifies the alternative protocols which can be used in information exchange, then summarizes their advantages and disadvantages. Based on the analysis of the research work, Section 3.2 introduces general gossip protocols and their features in fault tolerance. Section 3.3 classifies the possible failure models under emergency/disaster scenarios, then Section 3.4 applies gossip protocols and its adaptive schemes to these failure scenarios to demonstrate the robustness of gossip protocols. The last section concludes and summarizes the entire chapter.

3.1 Introduction on Data Exchange Protocols on Application Layer

Several data exchange protocols are proposed in large scale and dynamic networks such as peer-to-peer or mobile ad-hoc networks. A naive approach to disseminate information and data is flooding. It can be used to exchange both data and queries. Flooding has the advantage of very fast information dissemination speed. However, flooding has some serious problems such as being extremely wasteful with bandwidth and energy: a large (linear) part of the network is covered with irrespective information and there is an enormous number of redundant messages [40]. This will become a fatal weakness for wireless communication based distributed systems, with consideration of the resource restriction on devices.

Aggregation protocols are another type of protocol worth being adopted for the purpose of saving the wireless bandwidth and the energy. Aggregation protocols are often classified in two prevailing categories: gossip-based protocols and tree-based protocols, and hybrid protocols of the two. Unlike tree-based techniques where nodes are organized into a tree, gossip-based protocols do not require a particular structure [44] [20]. Computation of aggregates in tree-based techniques is often done hierarchically in a bottom-up fashion [4]. Finally, hybrid protocols combine a gossip dissemination mechanism with a tree structure [71].

Tree-based protocols use a tree for computing aggregates in a bottom-up fashion. For example in the protocol of SingleTree [4], a node q broadcasts a query to construct a spanning tree on the network. GAP (Generic Aggregation Protocol) [14] builds and maintains a BFS(Breadth-First Search) spanning tree on the overlay network and uses it to incrementally and continuously compute and propagate aggregates. The tree-based protocols have the feature of high initialization building cost in terms of messages and energy, but relatively low cost during data transmission. However, in a dynamic environment, the underlay changes are frequent and unavoidable, due to either mobility of nodes or the failures. The huge cost to build and maintain a tree structure can lead to fast depletion of battery power, and shorten of the system lifetime.

In order to combine the benefits of gossip and tree, some protocols propose a hybrid approach that uses gossip dissemination over a tree structure. MultipleTree [4] is an enhancement of SingleTree that creates k independent spanning trees rooted at the querying node. Astrolable [6] [72] is a distributed information management system that uses gossip to construct an overlay tree for computing aggregates. In DECA [62], nodes are organized into clusters and super clusters.

In wireless sensor networks, the most popular flat network protocol is directed diffusion [37]. This protocol is perfect for one-source one-sink scenario, since it finds the optimized path from source to sink and keeps reinforcing this path. In the multi-source scenario, the definition of best path is changed to the path, which can also stimulate data aggregation among multi-source, instead of the shortest path in

between source and sink. A lot of protocols have been designed for a specific application and do not provide generic mechanisms for data sharing and collecting in wireless communication based mobile ad-hoc networks.

Some other problems or challenges that might hinder the direct application of existing work to the data-centric application of heterogeneous dynamic network system are:

- Very few systems explicitly consider the limited availability of communication resources and the limited energy on the embedded devices, yet this is a serious problem in wireless-communication based devices and platforms.
- Seldom does existing work considers the trade-off and balance in between robustness and redundancy, while these two metrics are both important for this scenario. The solution can be biased on only one of the two extremes. For instance, flooding is robust yet causes extremely huge redundant messages; structure-based aggregation protocols are efficient in avoiding redundant messages, however, are fragile to changes under the dynamics of the system and the environment.
- While aggregation protocols are used to save energy, more complex data fusion and data processing functions are not considered and supported, which limits the range of applications of aggregation protocols.
- Structured data exchange protocols are difficult to adapt to the system and environment dynamics. They suffer huge recovery cost in terms of recovery time and messages used for reforming the structure.

These reasons motivate the more flexible and robust data exchange protocol to be adopted to provide solutions to make up these problems.

3.2 Using Gossip in Robust Information Exchange

[88] introduces the solution of using epidemic algorithms (gossip protocols) on peer-to-peer diagram on mobile ad-hoc networks (MANET). Epidemic algorithms are used to conquer the above problems based on a key observation in MANET that, mobility does not necessarily hinder communication, but may support cost-effective information exchange. Grossglauser and Tse showed that mobility increases the capacity in MANET for delay-tolerant applications [88]. While these previous work focus on the feasibility of the gossip protocols on building service overlay for peer-to-peer diagram above MANET, this thesis focuses on adapting and refining gossip protocols for data-centric fusion applications above wireless communication based resource restricted devices. Especially, the performance of refined gossip protocols under system dynamics and the robustness in different failure scenarios is the focus of this thesis. The contribution is to provide design guidelines in the dimension of data exchange protocols for various distributed fusion applications to achieve specific application aims.

A gossip protocol is a style of computer-to-computer communication protocol inspired by the form of gossip seen in social networks. In gossip protocol, nodes exchange data in a random fashion: a node chooses with some probability a peer to exchange information with. Modern distributed systems often use gossip protocols to solve problems that might be difficult to solve in other ways, either because the

underlying network has an inconvenient structure, is extremely large, or because gossip solutions are the most efficient ones available. In each round of the basic gossip algorithms [8], a random pair of neighboring nodes is chosen to exchange their information. Computer systems typically implement this type of protocol with a form of random "peer selection": with a given frequency, each machine picks another machine at random and shares any hot rumors.

In gossip protocols, the information exchange between nodes can be implemented as one of the following policies: only the node that initiates a gossip, sends local state data to its partner (push); a node-initiator requests state data from its gossip partner (pull); both nodes send their state information to each other (push-pull). Figure 3.1 illustrates the skeleton of a generic push-pull gossip protocol. Each node has a local state *s* and executes two different threads, an active and a passive one. The active thread periodically initiates a state exchange with a random peer *p* by sending a message containing the local state *s*, after that it waits for a response. The passive thread waits for a message sent by an initiator and replies to it with its local state. The random peer selection is based on the set of neighbors as determined by a membership protocol.

```
do once in each consecutive iteration:
        q <-- SelectNeighbors()
        send Sp to q
        Sq <-- receive(q)
        Sp <-- UPDATE(Sp, Sq)
```

(a) active thread

```
do forever:
        Sq <-- receive(*)
        send Sp to sender(Sq)
        Sp <-- UPDATE(Sp, Sq)
```

(b) passive thread

Figure 3.1: Push-pull Gossip

In the information distribution protocols, there are four steps including:

- **Initialization:** selection of information to be selected and initialization of the communication sockets
- **Select peers:** select m(adjacent) communications partners. A purely random peer-selection scheme for gossip is: when agent A decides to execute a gossip round, it picks some peer B uniformly and at random within the network as a whole (or launches a message on a random walk that will terminate at a random agent).
- **Transmission:** exchange of information by either a push or pull method
- **Update:** process received information, and update the newest information to local memory or to a database

In "push gossiping", the new information owner pro-actively sends its new data to its selected neighbors, but does not exchange information with the selected neighbors. While in "pull gossiping", the new information owner wait passively until it receives the requests asking for data. The "push gossiping" and "pull gossiping" can tolerate bad asymmetric message loss.

In dynamic distributed systems connected by wireless communication, during the iteration of gossip, message is in fact broadcasted by radio to different neighbors. There is a probability of overhearing

from some nodes. Thus the speed of information dissemination of gossip above wireless platforms may over-perform the speed above the platforms of wired connections. (This is the best case, supposing other nodes are not using the communication channel and causing any interference or conflict).

This thesis focuses on the performance and application of gossip protocols for information exchange in wireless communication based distributed networked systems, where system dynamics are taken into consideration. Different from existing research, this work considers the resource restriction on some battery-powered devices. The difference between this work and the existing aggregation protocols is that in this work, the gossip protocols are combined with the complex fusion tasks and formed into configuration plans to reduce energy consumption.

Furthermore, this work considers the performance of gossip protocols in different fault scenarios, suggesting the solution space on how to choose different protocols in different application scenarios. The analysis provides some hints on balancing the time performance and energy through controlling gossip fan-out (branch numbers), this can be used to adapt to different requirements of different applications. For example, broadcast is best in terms of time performance at the cost of extremely high consumption of energy and waste in bandwidth. Through choosing the fan-out of gossip protocol, one can control and balance in between different design aims, according to the fact that larger fan-out leads to faster speed, yet higher cost on bandwidth, on communication messages, and on energy consumption.

While considering the difference in between wired and wireless communication, and adapt gossip protocol to wireless fusion applications, gossip protocols are refined to reduce redundancy and to save the cost on wireless communication. This is another part of this thesis.

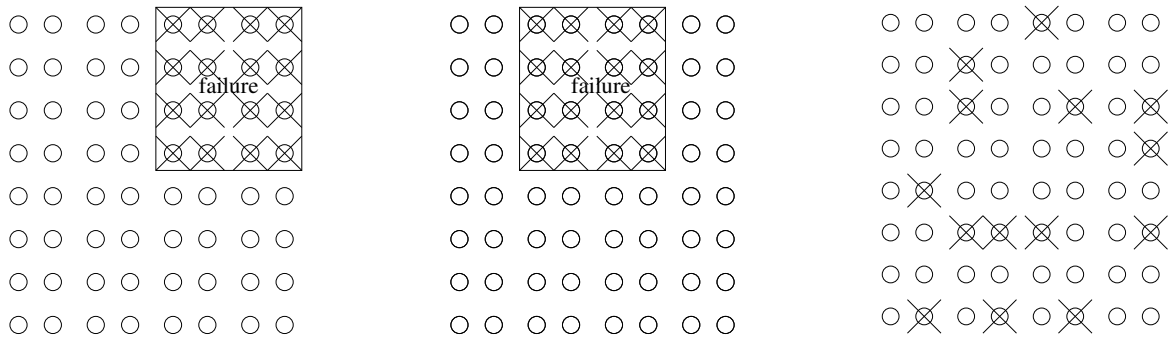
3.3 Failure Models in Heterogeneous Dynamic Systems

The thesis faces a system design problem, where it is important to choose the proper gossip protocols to meet the special design requirements. For instance, when we think about multi-source multi-sink dissemination problem, where the data aggregation should be promoted to reduce energy consumption, and fault tolerance and robustness should be considered in disaster scenario. Gossip was featured as locally simple yet globally powerful in information dissemination, which also provides robustness and flexibility in dealing with faults and failures [18] [23]. We start from considering different kinds of failure models either caused by disaster/emergency or caused by other dynamic scenarios. While [88] focusing on the challenges coming from underlay mobility, this thesis considers the interference to information dissemination caused by failures under emergency/disaster scenarios.

Some disasters or system dynamics might directly cause dysfunction of the hardware devices; some disasters might indirectly interfere with the wireless communications so that the connectivity among peers is not always of high quality. We thus classify these failures into the following models.

3.3.1 Location Related Spatial Failure Model

These failures are directly caused by disasters, in the disasters such as fire, flooding, hurricane, or earthquake, the damages to the sensor network might be restricted into a certain area. For example, figure 3.2 shows two failure models which are caused by disasters. It is impossible to enumerate all



(a) L-Failure Model (b) U-Failure Model (c) Random Failure Model

Figure 3.2: Location Related Spatial Failure Models

kinds of location related failures; however, these two examples can demonstrate the influence of failures' locations to the feature map dissemination time and the accuracy in disaster scenarios.

The disasters cause crashes of local nodes and loss of wireless communications in the certain area. The other part of the system, which does not get influenced by the disasters, will function normally. In such a case, two different interferences take effect on the system: on the one hand, the connectivity of the sensor network is changed which leads to the modification of routing paths; on the other hand, the scale of data dissemination is reduced with the decrease of node number, which further reduces the dissemination time.

The two kinds of influence might interact with one another in the evaluation metrics such as spreading time and energy consumption, depends on the location of dysfunctional failure areas. On the right side is the U-failure, where the destroyed sensor nodes are in the middle of the deployment area. In the L-Failure on the left, the damage is restricted to small areas in the northeast of the network. Later we will compare the influence of the location to the damage of the sensor networks in detail.

3.3.2 Random Failure Model

Contrast to the space related failure caused by disasters, random failure is more common and easier to take place. This can be caused by nodes in lack of power, or failure of relative software and hardware on the nodes. The failure can occur either randomly in time or in space. Some random failures occur temporarily, it includes the situations that nodes function abnormally or wireless communication signal goes down or weak for a while.

We consider the worst case, where failure means completely crash of nodes without considering its recovery, in order to simplify the assumptions. Thus the failures occur randomly in space is of the focus. Different from the space related failure caused by disasters, random failure is more common and easier to take place. Considering the fact that the failed nodes are independent from each other, the distribution of failed nodes conforms to binomial distribution. The thesis adopts a simple assumption: each node in the network system is considered of either working perfectly or failing competely, thus the probability of failure for each node is 50%. Besides, the event that one node fails is independent from the event that another node fails.

3.3.3 Message Omissions

In wireless communications, even the topology of the network does not change, the instability of wireless signals and links leads to message omissions. The damages may come from failures of the hardware or the software of wireless communication, or from the interference in the environment such as weather or temperature. We will later discuss in detail the influence of message omissions to the performance of gossip-based data dissemination, especially in the case where the percentage of message loss increases dramatically. In the assumptions, we assume that every link in average drops the same percentage of messages as the other links, and the message drops are random. Both the data message and protocol messages can be dropped. This is also based on the independence of events. This failure can limit the iterative and repetitive data transmission of gossip, which is used for guaranteeing robustness. The original gossip protocols generate a huge amount of protocol messages and data messages, some messages are out-of-date, useless, while some are updated effective messages. If we do not differentiate these messages, the repetitive information (which is designed to guarantee robustness), they can consume up the limited communication resources. This would lead to congestion or even paralysis of the peer-to-peer system. This common failure model in wireless communications motivates our improvement and refinement of the original gossip protocols in the next section.

3.4 Refining Gossip Protocols for Wireless and Heterogeneous Networked Systems

The robustness of Gossip protocols comes from the random selection of nodes neighbors during iteration of each gossip round. This requires repetitive data copies exchanged among active nodes. In another words, the redundancy and repetitiveness are the reasons to make gossip protocols robust in different scenarios. However, uncontrolled redundancy and repetitiveness leads to wasted wireless bandwidth and extra energy consumption. Though this is not a terribly serious problem in wired communication based services and applications, this is fatal for wireless heterogeneous system consisting of nodes and devices with resource constraints. Thus the flexibility to control and balance redundancy, the consumption on communication bandwidth, and energy is necessary for gossip protocols to be successfully applied to the wireless heterogeneous networked systems.

In order to deal with this problem, it is necessary to rethink and analyze the original gossip protocols. One useful observation is that the key method in gossip protocols is selectPeer(), which generally select the proper peer to exchange information in the next gossip iteration. This method in fact makes decisions on redundancy transmissions and determines the performance and reliability. In earlier work, all nodes had a global view to select a random peer with equal possibilities. This brings an extra communication cost, because the same peers can be contacted in the consecutive iterations, thus redundant messages are transmitted repetitively between nodes. At the same time, nodes which have not gotten updated information will remain stale after several consecutive gossip iterative rounds.

The downside of this design is obvious: on the one hand, the wireless communication bandwidth and the limited battery capacity are wasted on redundancy transmission. On the other hand, the up-to-date data is not forwarded out to the nodes, that actually require it. This delays the information dissemination. It deteriorates the situation further, especially in the scenario where failures of wireless communication

links occur. The double effect of retransmission of data and protocol messages caused by link failures combined with the transmission of redundant messages can cause the resource limited devices to consume up its battery power. The result is that the devices are forced to quit from proving sensing, data processing, and data transmission services from the network.

Figure 3.3 illustrates this general problem by comparing generic gossip protocols' performance under different failure models. The metrics used here for comparison is the number of messages needed for disseminating certain information among a network consists of 500 nodes, and 3 of the nodes are information source. The total message number is averaged on each node, until the dissemination process ends (when all nodes are aware of the updates). In the scenario where there is no failure, it takes around 40 seconds to disseminate the information. The average number of messages consumed is around 57 on each node. Almost all the other failure scenarios cause either the increase in dissemination time (random failure), or cost more messages, or even both (U-failure, 25% message loss, 50% message loss, 75% message loss). This shows how failure scenarios affect the performance of wireless communication based gossip protocols. Especially in the situation of message loss caused by the failure of wireless links, the performance of the original gossip protocols in terms of number of messages consumed is also deteriorated and cannot meet the requirements in controlling the overhead.

For gossip protocols to be deployed on wireless communication based heterogeneous systems, it is necessary to solve this problem. The solution introduced in this thesis is to refine gossip protocols by utilizing local history record for the selective exchange message between neighbor peers in each iterative gossip round. Although in wireless communication, the message is broadcasted out so every node inside the communication range has a possibility to eavesdrop on the message. Only the nodes, selected as target peers, are guaranteed to hear the message even if conflicts occur. The other peers, which happen to eavesdrop on this radio broadcast message, do not have such guarantee. So each round, they may get updated or may not get updated information. All depends on whether conflicts of wireless signal occur caused by hidden station, thus timing to transmit information matters. In evaluating and comparing the performance of different protocols, we can only calculate the worst case, which means only the selected target peers get the information exchange completely. This ignores the nodes, which happen to overhear the updated information. In the following paragraph, the general principle of refining gossip protocols is described in details with illustrated examples. The aim of this design is to save certain amount of unnecessary messages while maintaining the random characteristics of gossip protocols to keep the robustness feature.

The principle of refining gossip protocols can be illustrated in figure 3.4. This shows a simplified example: in each gossip iterative round, one peer can only guarantee information exchange with two of its neighbor peers. Each peer maintains a history, which can record past two iterations' contact peer list. In each of the following round, the results of selectNeighbors() function should mismatch the results of the past two iterations, otherwise, it should re-select new peers until this requirement is met. This step guarantees to select only those neighbors, which have not yet been contacted for information exchange.

This strategy can be used in both the push and pull process: in the push process, where the new information owner pro-actively chosen the nodes which have not yet been contacted to send the information or data. In the pull process, it is also possible that the information demander can send queries to the chosen nodes, which have not been queried before. This strategy avoids repetitive data transmission among already contacted neighbors, thus speeds up the information dissemination among neighbors, which have not yet been contacted. The trade-off here is to utilize local memory to reduce the consumption and waste

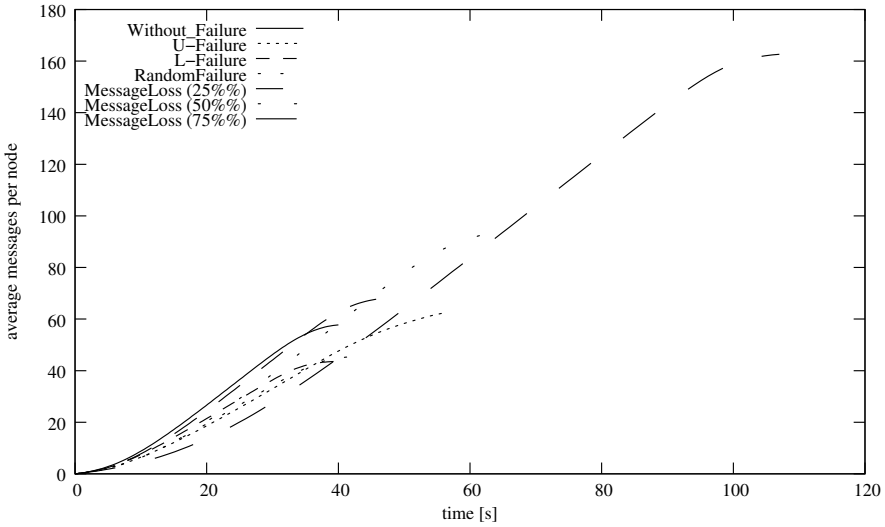


Figure 3.3: Number of Message Consumed by General Pull Gossip Protocol under Different Failure Models

on wireless bandwidth, and further to save up the energy needed for wireless communication. The size of utilized local memory size changes with the window size which decides how many rounds of history record should be saved. This is one of the interesting parameters, which determines the performance of information exchange. The effectiveness of the refined gossip protocols, especially under different failure scenarios, is described in detail in Chapter 6.

3.5 Using Refined Gossip Protocols in Decision Fusion

Refined gossip protocols provide several advantages in terms of dissemination time, number of communication messages, and etc. These features is useful in a broad range of data-centric applications above wireless communication based distributed systems. One use case regarding decentralized change detection in wireless sensor networks is used as example to show the application of refined gossip protocols.

The motivation and the details of this use case can be found in Section 1.2.2. Fig 3.5 illustrates the concept of decentralized decision fusion for the results of local change detections in wireless sensor networks. The reports from various sensor nodes even in the neighborhood might appear to be completely different. Some faults of sensor node itself contribute to the report of invalid values. These results might influence the final decision considering whether or not a warning of fire should be triggered. This section only focuses on the application of refined gossip protocols in the decentralized change detection framework, and how it meets the special design requirements of this application scenario.

One of the major challenges in change detection for wireless sensor networks is the detection accuracy against different fault models under restricted resources (such as memory, communication bandwidth, and battery power) of sensor nodes.

In order to improve the detection accuracy, the decentralized change detection framework adopts two levels of data fusion functions to explore the time correlation and the space correlation of the sensing

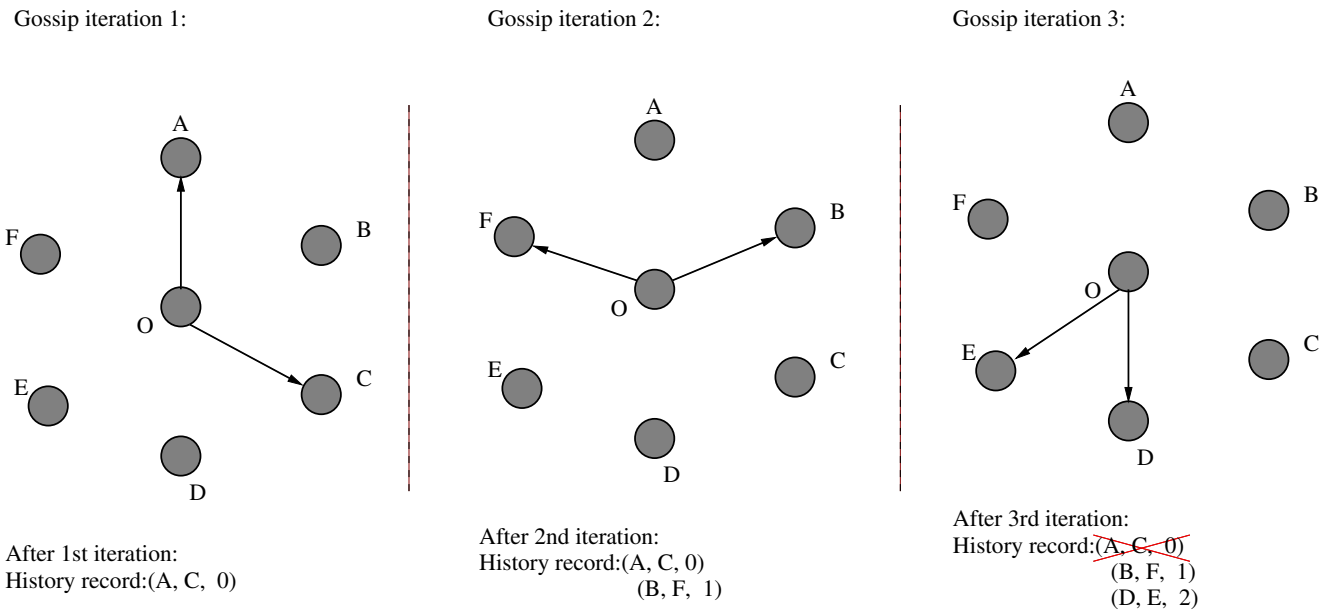


Figure 3.4: Adding History Record

data. Both of these two data fusion functions should be embedded on sensor nodes to provide in-network data processing. This helps save the communication cost to send all sensing data to the centralized sink to process. The local change detection fusion function uses Discrete Fourier Transformation coefficients as synopsis of structures from signal-oriented data streams. While the global decision fusion function should fuse the local change detection results by exploring the spatial correlation among nodes in the neighborhood.

Therefore it is necessary to exchange messages carrying different types of data among peers inside the distributed system. One type of data message is the sensing data (measurements), the other type of data message is the local change detection results, and also the global change detection decision. Energy efficiency is a critical requirement for the data exchange process. This requirement suggests that "push gossip" might be adopted in data exchange. Besides, the detection results should be delivered as soon as possible to certain nodes, so further actions can be taken. This requirement suggests "push-pull gossip" is a good candidate option to be adopted in the decentralized change detection framework. When prioritizing these two design aims, one can conclude to use refined "push-pull gossip" in emergency applications to transmit the detection warnings quickly. If in some other cases of change detection where long-time monitoring of certain area is the purpose, refined "push gossip" can be used to reduce the wireless communications cost on resource limitation devices.

Suppose only one sink node is responsible for taking action, the framework should transmit and fuse the change detection results from the ambient area of events to the sink. This helps to find a consensus among nodes in order to improve global detection accuracy while maintaining reasonable communication cost for the wireless sensor networks. The specific usage of the "push-pull gossip" to exchange data among peers while fusing consensus from several detection results can be seen from the two algorithms: algorithm 4 and algorithm 5. This shows how to integrate the data exchange supported by "push-pull

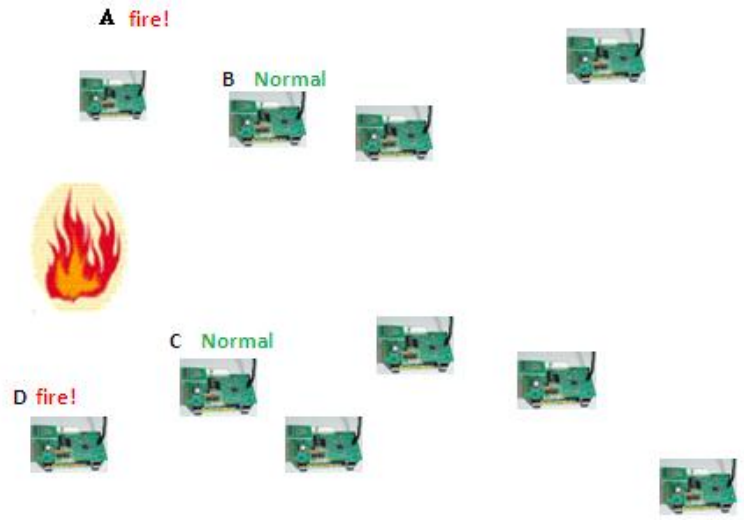


Figure 3.5: Application Example of Decision Fusion

gossip" with the decision fusion process on each sensor node.

Algorithm 1 Decision Fusion Using Gossip to Exchange Information: Active Thread

begin *active thread:*
do once in each consecutive iteration:
q= SelectNeighbors();
$status_p$= $changedetection_p$;
Send($status_p$) to q;
$status_q$= Receive($changedetection_q$);
$fusedDetection_p$=
DecisionFusion($changedetection_p$,$changedetection_q$);
$status_p$= UPDATE($fusedDetection_p$);
end *active thread*

3.6 Summary

In this chapter, we explore the potential of using gossip protocols for the purpose of fault tolerant data exchange. Firstly, we introduce the advantage of utilizing gossip protocols on application layer of heterogeneous dynamic networked system to aggregate and process data. It relies on the flexibility to trade off in between robustness and communication cost on bandwidth and energy. To foster data dissemination for specific devices with limited radio transmission range, and limited battery capacity, this chapter describes a scheme denoted selective data exchange. This improves the gossip protocols to reduce the possibility of transmitting of redundant message among devices. Secondly, in order to meet the specific design requirements for different failure scenarios and to evaluate the performance of different gossip protocols, some failure models are studied in details. Relative evaluation and analysis on

Algorithm 2 Decision Fusion Using Gossip to Exchange Information: Passive Thread

begin *passive thread:*
do forever:
$status_q$= Receive (*);
$status_p$= $changedetection_p$;
Send($status_p$) to sender($status_q$);
$fusedDetection_p$=
DecisionFusion($changedetection_p$,$changedetection_q$);
$status_p$= UPDATE($fusedDetection_p$);
end *passive thread*

these protocols are described in detail in Chapter 6, Section 6.1. The thesis suggests the proper selection of different gossip protocols to meet different design requirements for different failure models, also in Chapter 6.

This chapter introduces one example to demonstrate the applications of using gossip protocols in data dissemination in distributed systems. It uses refined gossip protocols to exchange local decision information, and aggregate the information to generate a global decision regarding warnings. This general application can be used in change detection or accuracy related applications to improve the data quality, which is disseminated in the distributed heterogeneous networked system. As this example shows, data exchange protocols should be combined with data processing components (data fusion functions) into a data processing framework in order to accomplish the data centric applications. Following this chapter, the next chapter will focus on data processing models. The novel model to map data fusion and data exchange into a data fusion graph eases the implementation and deployment of such framework above the heterogeneous dynamic systems.

Chapter 4

Distributed Data Fusion Model – Directed Fusion Graph

Data fusion is one of the three important consisting parts of the data fusion and harvesting framework. It represents the data processing or computation part of this framework. The data fusion components need to be connected with each other through data streams transmitted by data exchange protocols introduced in the above Chapter 3. Thus this chapter describes the data fusion model in the framework. The contribution of this part is that it provides a novel distributed data fusion model. It facilitates the logical representation of the structure of the data processing framework, combining both the elements of data exchange and data fusion. This model further benefits the automatic configuration and deployment of this framework onto real distributed systems.

In the following sections of this chapter, the formal definition and classification of data fusion is first introduced in Section 4.1. It is followed by some examples of typical data fusion functions and the architectural models of the information fusion system. In Section 4.4 the decentralized model denoted decentralized directed fusion graph model is introduced to describe the entire logical structure of the data processing framework. Section 4.5 enables the robust information exchange protocols to be represented in the directed fusion graph. This combines the contents in the last Chapter together. The novelty and features of this work comparing with existing work is stated in section 4.6. The following sections explain how to extract the directed fusion graph in specific applications, by three examples.

The aim to introduce this novel model is not only a theoretical contribution but for practical implementation, configuration and deployment of the framework. After this chapter, how to utilize these logical models in the implementation, configuration and deployment of the framework is discussed in Chapter 5.

4.1 Definition and Classification of Data Fusion

Data fusion or information fusion deals with paradigms and techniques for “fusing” multi-source data and information. It is defined as the synergistic process of associating, correlating, and combining data and environmental factors to derive information and knowledge. [61]

A similar concept emerging in wireless sensor network research is data aggregation. It comprises the collection of raw data from pervasive data sources, the flexible programmable composition of the raw

data into less voluminous refined data, and the timely delivery of the refined data to data consumers [63]. Data aggregation implies the accuracy improvement, and represents the ability to summarize, which means the amount of data is reduced. Therefore, the concept of data aggregation is referred to one instance of information fusion: summarization.

Due to the complexity of data fusion applications, various formats of data fusion exist which differ from one to another. Several methods of classifications are mentioned in [63], to summarize existing data fusion methods from several aspects.

4.1.1 Classification Based on Levels of Abstraction

A common classification method for data fusion is to classify fusion functions and techniques based on levels of abstraction, as mentioned by [57] [15] [38] [63]. In general, data fusion was classified into three levels, according to the abstraction levels, from the lowest level to the highest level: **Data level**, **Feature level** and **Decision level**, as shown in Table 4.1.

Table 4.1: Data Fusion Classification based On Abstraction Level

Fusion Level	Fusion Functions	Techniques
Data Level	Spectral Data mining	Digital Signal Processing
	Data Adaptation	Coordinate Transforms Unit Adjustments
	Estimation of Parameters	Kalman Filtering Batch Estimation
Feature Level	Classification	Pattern Recognition Fuzzy Logic Neural Networks
Decision Level	Decision Action	Expert Systems Artificial Intelligence

- **Data level**: Raw data are provided as inputs, combined into new piece of data that is more accurate (reduced noise) than the individual inputs.
- **Feature level**: Attributes or features of an entity (for instance, shape, texture, and position) are fused to obtain a feature map that may be used for other tasks (for instance, segmentation or detection of an object).
- **Decision level**: It takes decisions or symbolic representations as input and combines them to obtain a more confident and/or a global decision.

The method classifies the fusion functions and techniques, due to their abstraction level. Besides these three basic levels, multilevel fusion is proposed to represent the fusion process encompassing data of

different abstraction levels – when both input and output of fusion can be of any level. With the abstraction level increases from data level to decision level, the complexity of the algorithms and functions also increases. This classification method focuses mainly on the fusion functions and techniques, thus provides a really good cutting point to understand the various typical and general data fusion techniques, and their relations to each other. These fundamental techniques can be applied to a lot of application fields as specific methods to fuse different type of data across specific application disciplines. In Section 4.2, more details of these typical fusion functions are provided to help readers understand the typical and common-use data fusion techniques.

The data level fusion techniques are adopted in the use cases of this thesis as a basic technique to improve the data quality. Data fusion techniques on feature level or above and the multilevel fusion are the main focus of these use cases. This is to say that the combination usage of different levels of fusion techniques and functions are quite common in practical application scenarios. Although, directly adopting the fusion techniques at the decision level, such as expert system or artificial intelligence are sometimes restricted by the capability of the devices.

4.1.2 Data fusion based on Relationship among the Sources

[68] uses another classification method, which categorizes the information fusion as complementary, redundant or cooperative. This classification method focuses only on how the data from various sources interact with each other and their relationships in the fusion process:

- **Complementary**: When information provided by the sources represents different portions of a broader scene, information fusion can be applied to obtain a piece of information that is more complete (broader). An instance of this complementary model is the feature map, which fuses data from individual sensors to form a general feature map that describes the whole sensor field [101] [91] [66] [79]. The use case of resource map listed in Section 1.2.1 is one specific example of this kind of complementary data fusion. The measurement data retrieved from sensor nodes distributed in local areas are complementary to each other in forming the entire map of the field.

- **Redundant**: When more than one data sources provide the same piece of information, these information can be used to improve the accuracy of the information. Besides, this model can also be used to improve the reliability, and confidence of the information. However the redundant information is harmful, in that, it creates more cost for transmission, and maintenance, etc. An instance of the redundant data fusion is the Kalman filter, which is used to get rid of errors modeled as Gaussian noise.

- **Cooperative**: Two or more independent data sources provide data, which can be fused into new information (usually more complex than the original data). Location estimation is one example of cooperative data fusion, where several measured received signal can be combined to estimate the actual location of the static/moving target, based on the relation of the distance with signal strength, and the angel information. This fusion method is broadly used in applications such as change detection in wireless sensor networks.

These three categories proposed by [63] demonstrate the main possible interactions and relationships among data sources. It is also possible for the combination of such relationship to exist in a certain ap-

plication scenarios. For instance, in the use case of the generation of resource map, complementary data fusion is applied to improve the data quality of the initial measurement, while cooperative data fusion is also adopted to locate the actual position of the base station. The advantage of this data fusion classification method is that it originates from the data sources, which helps the application designer to understand the usage of the data. This method does not consider the specific formats of the data fusion functions or techniques which should be adopted, thus leaves a lot of flexibility in selecting the possible specific data fusion functions. The disadvantage is that without specific application scenarios, it is difficult to tell directly from fusion functions, which category they fall into. Therefore, such classification method is more application related, comparing with the classification method based on abstraction level listed in section 4.1.1.

4.1.3 Classification Based on Input and Output

[63] proposes another well-known classification that considers categorizing the fusion process based on the abstraction level of the input and output information. Table 4.2 shows these five categories in this classification method: This classification considers both the input and output data of a fusion function.

Table 4.2: Data Fusion Classification based On Input and Output

Fusion Classification	Input	Output
Data In-Data Out (DAI-DAO)	raw data	raw data
Data in-Feature Out (DAI-FEO)	raw data	features or attributes
Feature In-Feature Out (FEI-FEO)	features	new features
Feature In-Decision Out (FEI-DEO)	features	symbolic representations or decisions
Decision In-Decision Out (DEI-DEO)	decisions	new decisions

Compared with method 4.1.2, it considers the general effect that **one** fusion function generates, and connects the input to the output. This has a finer granularity. The model also does not care about the specific format of the fusion functions; this is similar to the model that this thesis proposes. Due to the strictness of the five categories defined, this classification does not allow the multi-level data fusion, due to its complexity.

The use cases that this thesis exemplifies concern relatively high-level of categories, such as the Data In-Feature Out (DAI-FEO) model in resource map, and Feature In-Decision Out(FEI-DEO) in change detection. Besides, the view to observe and classify the fusion function according to its input and output is inspiring for the data fusion model proposed by this thesis.

4.2 Typical Data fusion functions

The data fusion functions and techniques are basically of two general categories: one is the general techniques and fusion functions which can be applied in broad application fields without too much concern about the specialty of the field knowledge. Although the data are in various formats, these fusion techniques can be used as general mathematics formulas to apply to the data. The specific regulation

regarding the application field does not need to be taken into consideration. In the other category, the field knowledge is needed to be considered and combined with the general data fusion techniques. For instance, when dealing with the transmission of electronic signals, the fusion function has to consider the features and regulations of these signals transmitted in different media. Such fusion functions are difficult to enumerate without specific application scenarios. This thesis tries to demonstrate both those cases. In the following space, typical and general fusion functions and techniques of the first category are introduced. Some examples of the second category are demonstrated by Section 4.7, Section 4.8 and Section 4.9. This is to show how the field knowledge should be combined with typical data fusion techniques to satisfy the requirements of specific application scenarios.

The typical data fusion functions and techniques exemplified in the following spaces fall into the categories of the classification method on level of abstraction, listed by Table 4.1.

4.2.1 Digital Signal Processing

Digital signal processing (DSP) belongs to data level fusion techniques. It is the mathematical manipulation of an information signal used to modify or improve it in some way. It is characterized by the representation of discrete time, discrete frequency, or other discrete domain signals by a sequence of numbers or symbols and the processing of these signals. [21]

DSP usually works among these domains, including: Time and space domains, frequency domain, Z-plane analysis, Wavelet. The first step is usually to convert the signal from an analog to a digital form, by sampling and then digitizing it using an analog-to-digital converter (ADC), which turns the analog signal into a stream of numbers. However, oftentimes, the required output signal is another analog output signal, which requires a digital-to-analog converter (DAC). Even if this process is more complex than analog processing and has a discrete value range, the application of computational power to digital signal processing allows for many advantages over analog processing in many applications, such as error detection and correction in transmission as well as data compression [21]. For instance, moving average filter is widely adopted in digital signal processing solutions due to its simplicity. This filter is optimal for reducing random white noise while retaining a sharp step response. It is used by Nakamura et al. [64] to estimate the data traffic of continuous WSNs. That estimate is further used for routing-failure detection. Yang [94] applies the moving average filter on target locations to reduce errors of tracking applications in WSN. [41] used moving average filters to improve the sensor readings.

In the use case of decentralized change detection, digital signal processing techniques are used to transform the signal from the time domain to the frequency domain by Discrete Fourier Transformation, since in the frequency domain, sudden changes are obvious.

4.2.2 Coordinate Transforms

A Cartesian coordinate system allows position and direction in space to be represented in a very convenient manner. Unfortunately, such a coordinate system also introduces arbitrary elements into our analysis. After all, two independent observers might well choose coordinate systems with different origins, and different orientations of the coordinate axes. In general, a given vector will have different sets of components in these two coordinate systems. However, the direction and magnitude are the same in

both cases. Hence, both sets of components must be related to one another in a very particular fashion. Actually, since vectors are represented by movable line elements in space, it means that the components of a general vector are not affected by a simple shift in the origin of a Cartesian coordinate system. The components are modified when the coordinate axes are rotated. There are basically two types of transformation from/to Polar to/from Cartesian Coordinates. Coordinate transformation is regarded as a fundamental technique used to pre-process coordinates related measurement data, so the data from different sources can cooperate with each other to generate effective results. [63] [13]

4.2.3 Kalman Filtering

Kalman [43] proposed it originally in 1960, later it was extensively studied [56]. It is used to estimate the state of a discrete-time controlled process that is ruled by the state-space model.

The Kalman filter is used to fuse low-level redundant data. It is best used to deal with a system that can be described by a linear model, with noises modeled as Gaussian noise. It can be used in wireless sensor networks, to reach a consensus among sensor nodes [84] [67]. For source localization and tracking, Kalman filter can be adopted to refine location and distance estimates [75] [35], and track different sources [52].

4.2.4 Batch Estimation

Besides Kalman filter, there are other estimation techniques which are summarized by [63], and [100], for instance:

- **Maximum Likelihood (ML)** searches for the value that maximized the likelihood function, that can be obtained from empirical or analytically sensor models. The maximum likelihood estimation is commonly used to solve location discovery problems. In this application, the method is often used to obtain accurate distance estimations that are used to compute the location of nodes or sources.
- **Maximum A Posteriori (MAP)** is based on Bayesian theory. It is used when the parameter x to be discovered is the outcome of a random variable with known PDF p(x). The measurement sequence is characterized by the sensor model. Both ML and MAP try to find the most likely value for the state x. Maximum Likelihood (ML) assumes that x is a fixed yet unknown point of the parameter space, while Maximum A Posteriori (MAP) takes x as the outcome of a random variable with prior PDF known.
- **Least Squares** is a mathematical optimization technique that searches for a function that best fits a set of input measurements. This is achieved by minimizing the sum of the square error between points generated by the function and the input measurements. The least squares method is suitable when the parameter to be estimated is considered fixed. Different from the MAP, this method does not assume any prior probability. The measurements are handled as a deterministic function of the state:

  Least Squares and Maximum Likelihood (ML) are equivalent when the noise is a sequence of outcomes of independent identically distributed random variables with a symmetric zero-mean PDF.

4.2.5 Pattern Recognition

[69] Pattern recognition is the assignment of a label to a given input value. It attempts to assign each input value to one of a given set of classes (for example, determine whether a given email is "spam" or "non-spam").

Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to perform "most likely" matching of the inputs, taking into account their statistical variation.

Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. **Supervised learning** assumes that a set of training data (the training set) has been provided, consisting of a set of instances that have been properly labeled by hand with the correct output. A learning procedure then generates a model that attempts to meet two (sometimes conflicting) objectives: Perform as well as possible on the training data, and generalize as well as possible to new data.

Unsupervised learning, assumes training data that has not been hand-labeled, and attempts to find inherent patterns in the data that can then be used to determine the correct output value for new data instances. A combination of the two that has recently been explored is **semi-supervised learning**, which uses a combination of labeled and unlabeled data (typically a small set of labeled data combined with a large amount of unlabeled data). Note that in cases of unsupervised learning, there may be no training data at all to speak of; in other words, the data to be labeled is the training data.

Clustering algorithms are often used to recognize patterns. It groups the input data into clusters based on inherent similarity measure(for instance, the distance between instances, considered as vectors in a multi-dimensional vector space). This algorithm is used in the LEACH protocol [49] to adaptively cluster the wireless sensor nodes based on the energy availability.

4.2.6 Fuzzy Logic

Fussy logic generalizes probability [3] and therefore is able to deal with approximate reasoning to draw conclusions from imprecise premises. Each quantitative input is calculated by a member function. The fuzzy rules of an inference system produce fuzzy outputs which, in turn, are calculated by a set of output rules. [32] and [33] use fuzzy reasoning for deciding the best cluster-heads in a WSN. Three features were used including: node concentration, energy level, and centrality. These features were turned into linguistic variables and then a rule is obtained. [98] uses fuzzy logic to do efficient routing. They use transmission energy, remaining energy, rate of energy consumption, queue size, distance from the gateway, and current status as input variables; the fuzzy output is the cost.

The fuzzy logic method is used by [17] to generate rules in classifying the normal sensor nodes and dysfunctional sensor nodes, which can later work as guidance to discriminate the dysfunctional sensor nodes in the monitoring and fault discovery system.

4.2.7 Neural Networks

Neural Networks are structures that implement supervised learning mechanisms. This technique can be used by classification and recognition tasks in the information fusion domain. A key feature of neural

networks is the ability of learning from examples of input/output pairs in a supervised fashion. A neural network has two phases: one training phase which takes the history record input and outputs to figure out the relative variables in the structure. The second phase is the estimate and classification phase, which uses the parameters resulted from the first phase to form the structure on the new input, through calculation, get the classification output result for the input. [97] proposes a fusion scheme to fuse edge maps from multi-spectral sensor images acquired from radars, optical sensors and infrared sensors. [17] uses a neural network to classify the sensor nodes into normally functioning ones and failure nodes.

4.2.8 Expert System

An expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning about knowledge, like an expert, and not by following the procedure of a developer as is the case in conventional programming. [24]

An expert system has an unique structure, different from traditional computer programming. It is divided into three parts: one fixed, independent of the expert system–the inference engine; one variable: the knowledge base. To run an expert system, the engine reasons about the knowledge base like a human. And a third part is a dialog interface to communicate with users.

These three parts are all able to be constructed in a sensor network, where sensors may act as data sources, together with the inference engine. While more powerful computational nodes or sinks function as knowledge base, and provide a dialog interface to users of the network. This in fact forms a decentralized expert system.

4.2.9 Artificial Intelligence

Artificial intelligence (AI) is the intelligence of machines or software, and is also a branch of computer science that studies and develops intelligent machines and software. Major AI researchers and textbooks define the field as ”the study and design of intelligent agents”, where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. [2]

Some important problems of AI include reasoning, knowledge, planning, learning, communication, perception and the ability to move and manipulate objects.

Constrained by energy limit, current sensor networks are in lack of the intelligence to carry out most of the activities individually. It is possible for the sensor nodes to participate as a cooperative processing unit to finish some activities. With the increasing capacity of the sensor nodes, some of the nodes are able to carry out more powerful computation tasks, and cooperate with other nodes to achieve the objectives as a whole networked system.

4.2.10 Conclusion on Typical Data Fusion Functions

This section summarizes the typical and commonly used data fusion functions which can be broadly applied on various applications. The typical data fusion functions cannot be enumerated; those listed here are just a subset of the typical fusion functions. Some other data fusion functions are application related. For instance, some signal processing method will be based on the propagation of electronic

signals; some video or audio compressing algorithms also need to take into consideration the features of these multimedia data.

The usage of these typical data fusion functions or their variations is common. The thesis uses a subset of these fusion functions or the variations to demonstrate how these typical fusion functions can be combined or organized in a practical application scenario to achieve certain meaningful results. This thesis tries to illustrate the usage of these fusion functions, especially with special data exchange protocols. These use cases and the fusion functions utilization are introduced in detail in Section 4.7, Section 4.8 and Section 4.9.

The contribution of this thesis, however, is not to invent a novel fusion function or a novel fusion technique. Instead, this thesis looks into the relation in between the data fusion techniques and data exchange protocols, to provide a general **architecture model** of decentralized data fusion, where these specific fusion functions can be integrated as plug-ins of the framework. The architecture is a flexible, configurable middle-ware framework, which supports the application of various data fusion functions combined with robust data exchange protocols. The following sections first introduce the classical architectural model of information fusion systems, and then the novel data fusion model is provided in the later sections of this chapter.

4.3 Architectural Models of Information Fusion System

This section focuses on the architectural models of the information fusion system. The models describe how each of the fusion techniques plays an role in constructing a data fusion system, and how the input and output data of these fusion techniques interconnect among these fusion components. Some of the models also consist of other necessary components besides pure fusion techniques, such as data storage, etc. The components of the architecture can be selected from several specific fusion techniques, to achieve the aim in processing the input data to get the output data.

Three classical models of the information fusion system are introduced in this section, based on the view of the information flow and fusion components required to constructing the system. In general, there are three models: the information-based models, the action-based model and the role-based model. After introducing these existing models, and analyzing their features, a novel model called Directed Fusion Graph is described in the following section, which is based on a different view to rethink the function and architecture of the information fusion system.

4.3.1 Information-based models

Information-based models represent the first generation of models for information fusion system, which focus on the abstraction level of the information handled by the fusion tasks. For example, the JDL model (invented by U.S. **J**oint **D**irectors of **L**aboratories) is composed of five processing levels, an associated database, and an information bus connecting all components. The model connects the data sources to the human computer interface with an information bus, and defines five different levels of data fusion functions which are also interconnected with the information bus. The JDL model represents the first serious effort to provide a detailed model and a common terminology for the fusion domain. However, the terminology adopted in this model is threat-oriented. This is because that the entire model

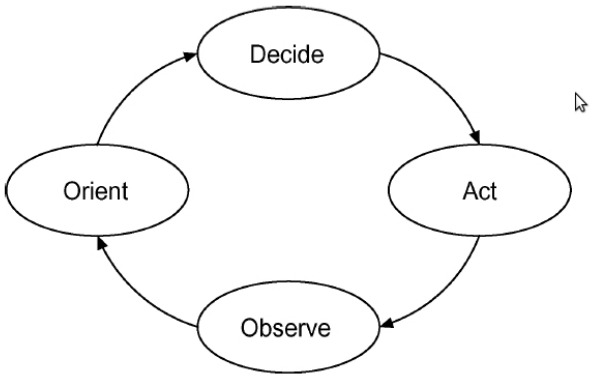


Figure 4.1: The OODA Loop [7], Figure From [63]

represents the system architecture serving military related information fusion applications. The data fusion components, especially Level 3 – the threat refinement component, are of no meaning and not suitable for the other general-purpose information fusion tasks.

Another DFD (**D**ata-**F**usion-**D**ecision) model [15] was created by Dasarathy. It is the first to define the fusion function into Data, Feature, and Decision components, then connects these components from low level Data fusion to high level Decision fusion in order, and interconnects these components with hybrid multilevel fusion. Comparing with the JDL model, this model is able to serve more data fusion applications. This is because this model emphasizes more on the input, output and their interconnections of each fusion component, regardless of the application domain.

The reason that these models are classified as information-based models is that such models emphasize how information flows pass through each fusion components, and how the fusion components interconnect with one another to process the information and achieve the entire fusion aim. These are the initial architectures or models suitable for a single powerful machine. Therefore these models do not specify the network aspects (distributed nature) of the distributed systems (for instance, wireless sensor networks, heterogeneous network system, MANET, etc). Thus these models can work only as an initial guide during the system design phase to specify which methods can be used and how they can be integrated to achieve the requirements of a given data fusion application. They are not suitable to be directly applied for the distributed networked system, which is the focus of this thesis.

4.3.2 Activity-Based Models

Some other models are specified based on the definition of the several necessary activities that must be performed by an information fusion system. Some examples are the OODA loop (**O**bserve-**O**rient-**D**ecide-**A**ct), as shown in fig. 4.1, and the Intelligence Cycle (Collection-Collation-Evaluation-Dissemination) as shown in fig. 4.2. The OODA model is a broad model that allows the specification and visualization of the system tasks in an ample way: it allows the modeling of the main tasks of a sys-

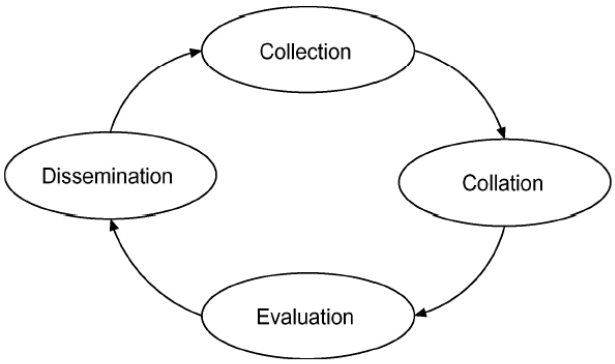


Figure 4.2: The Intelligence Cycle [77], Figure From [63]

tem. The Intelligence Cycle describes the process of developing raw (sensory) information into finished intelligence used in decision-making and action.

The Omnibus model (fig. 4.3) is an enhanced version of the OODA loop specifically for the information fusion domain, which specifies the details that the OODA activities should complete by describing the fusion tasks at each stage. This somehow causes the oversimplification problem of the OODA loop.

The activity-based models state the loop of actions on the data. In general, this kind of model starts from observing (measuring or sensing) untill the decision making and executing. Each activity can be completed by several data fusion functions. The general cycle or loop architecture shows the intrinsic self-contained processes to process data inside the information system. The loop or the cycle shows the equal relation instead of the hierarchical relation among these activities. The advantage of these models is that they are very easy to understand and obvious in defining the general activities that the system must accomplish. Besides they show clearly the sequences or the interconnection orders of the fusion activities. However, these models consist of oversimplified abstract activities, and somehow are in lack of details of the description of the fusion components and the interconnections among different fusion tasks. Thus the general theoretical models are not possible to guide the practical implementation or deployment of the system, due to their characteristics of over abstraction.

4.3.3 Role-Based Models

Object-Oriented Model [47] and Frankel-Bedworth Architecture [27] are two representatives of the Role-based Model. Object-Oriented Model uses the four most important roles (classes) and the association relationships among them to describe the structure of the information fusion system. In each role, the main actions that this role should take are also described. Such classes clearly show their responsibilities in the information fusion system. Such modular-based architecture can directly be used in the

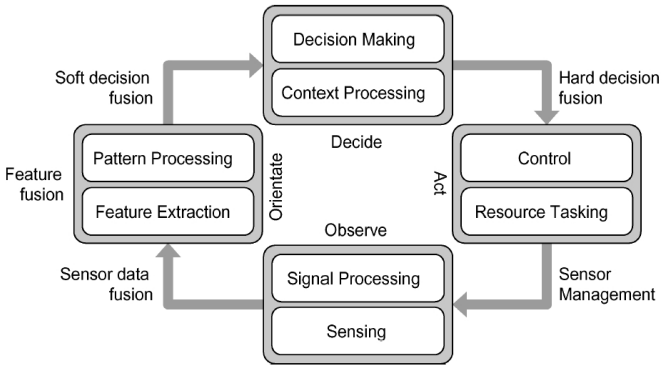


Figure 4.3: The Omnibus Model [5], Figure From [63]

system implementation, in the component design, although they do not directly connect with specific fusion techniques.

In Frankel-Bedworth Architecture, two major roles are introduced in accomplishing the interactions with the environment and the fusion tasks: Estimator and Controller. More detailed actions that these two roles should take are shown connecting each other with information within and between the two roles. This model first introduces the notion of a global process separated from the local process. The global control process (the controller role) rules the local process by controlling and defining its goals and monitoring its performance. The local process is supposed to implement and perform fusion methods to accomplish the system's objectives [63]. Comparing with the object-oriented model, this model reduces the roles that involve in the system, at the expense of extending the responsibilities that each role should take. This model describes in more details the actions of the roles, and the information or the message which is passed through among these actions.

Comparing with the information-based and activity-based models, the role-based models are fine-grained, and specify the actors and their roles in the fusion tasks. The aspects that this type of models looks into can be a good make up for the aspects of the other two models (information-based models and activity-based models) to form a complete understanding of the fusion task and the information fusion system. The problems that all of these three types of models suffer are:

- They are earlier generation of theoretical design, which do not need to take the **distributed nature and the capacity constraints** of the current complex heterogeneous network system (WSNs, etc) into consideration. Thus these models cannot be directly applied to the heterogeneous networked system this thesis focuses on. Because they are abstract and lack the details on practical implementation and deployment of the information fusion system on complex, dynamic distributed networked system.

- Some models stem from **specific** applications, for instance, the JDL model is for military fusion tasks, thus is not **general** enough to be applied to other fusion applications. Some other models are general enough to describe the necessary steps of the general fusion processes, yet they are too theoretical and abstract, so in lack of details to be adapted to specific applications.

- The three types of models all emphasize only on the fusion components, but **in lack of introducing data exchange protocols to describe in detail how data flows are exchanged among these fusion components**. Instead, these models somehow over-simplify the interconnections among those fusion components. The interconnections represent the information or data flow among these fusion components, thus not only what information/data should be transmitted in between two fusion components matter, but also the interconnection methods of the various interconnections (such as push/pull, publish subscribe, etc), even the condition which triggers the information from one component to another makes a difference on the effect of the information/data fusion. Thus the information fusion architecture should integrate with data exchange protocols to describe in more detail about how the data flows among all these data fusion components. All these conditions concerning data flows are in lack of consideration in the above existing three types of architectural models.

- Last but not least, none of the existing models support **dynamic updates of the information fusion architecture runtime**. All the models are **static and fixed**. Once the architectures are deployed, they cannot be easily updated to contain other fusion functions. This causes the data fusion system not to be able to meet some dynamic requirements due to either environment changes or to the information exchange protocols.

Thus it is necessary to research a novel architectural model suitable for the distributed information fusion system above heterogeneous dynamic networked systems. The novel model should **balance the relationship between generality (to be applied to wide-range of data fusion applications) and specialty (to be able to adaptive enough for meeting specific requirements of unique applications)**. This model should **integrate data exchange protocols into data fusion models** to provide supports to the diversity in the data fusion formats caused not only by fusion techniques but also by the dynamics of interconnection data flows among the data fusion components. To be able to **support dynamic architecture updates** is another aim to be achieved by the novel model.

4.4 Directed Fusion Graph Models

The existing research work tries to divide the tasks of the information fusion system and to describe the specific tasks from various aspects (information-based, activity-based or role-based). Seldom these models take serious consideration into the various interconnections options among the fusion components inside the fusion system, nor do they put any effort in dealing with the distributed nature of current complex system which underlay the data fusion applications. Some of the models stem from specific applications thus is not flexible enough to adapt to the general situation. After concluding all these features of the current models, it can be recognized that the difficulty that the data fusion model faces is to balance the relation in between generality and the specialty, and make the model useful to guide practical system designs.

The thesis proposes the Directed Fusion Graph Model, which is novel in modeling distributed data fusion systems, and aim to solve the mentioned problems. In order to deal with the generality of the information applications, the fusion components have to be general enough. Thus this work takes a step back to the original definition of the data fusion, which regards the main task to be “fusing” multi-source

data and information. Instead of directly regulating the aim or specific targets of the fusion components, this model treats the specific fusion components as a black box which connects several input data streams to the output data processing unit. Such abstraction helps solve the generality problem as the model should be able to deal with a broad range of applications instead of only few.

4.4.1 Formal Model for Decentralized Data Fusion

The formal model for decentralized data fusion is useful for guiding the specification, proposal and usage of information fusion. This model is not only helpful for describing the theoretical data processing structure, but also beneficial in practice for the usage of the framework.

A data fusion unit is the smallest component which can complement a fusion task. This work assumes that each fusion unit in the networked system has the ability to fuse data; this unit is a fusion component. Some components can be tasked of sensing to generate data sources. Some other components are capable of executing processing tasks of these data from different data sources. The formal model of data fusion unit can be described as a triple <S, O, f_f >. S is the set of input data streams; O is the set of output data streams, or the mapping result of this fusion function. The fusion function or algorithm f_f is responsible to map the input data streams to the output data streams. To be specific, the three elements represent:

- S is input data streams:
  Let $S = \{S_1, ..., S_M\}$ be a set of data streams from M different data sources, such as sent from M nodes (sensor nodes, mobile sensing nodes, smart phone; or the outputs from M data fusion components) in a networked system. $S_i = \{x_1, x_2, ...\}$ is a time series data stream of measurements or fused results at the i^{th} fusion unit. And each incoming data item x_j arrives in increasing order of j. Time series data streams are suitable for modeling the data produced by a sensing device as data source. The observed measurements are sampled during every time interval.

- O is output data steams:
  Similar to the input data streams of S, output data streams $O = \{O_1, ..., O_N\}$ is a set of data streams. $O_i = \{y_1, y_2, ...\}$ is a time series data stream of fused results at the i^{th} fusion component. And each O_j corresponds to the input data streams at the j^{th} time interval.

- f_f is a fusion function or algorithm:
  At any time interval j^{th}, the fusion function f_f matches the input data stream S to the output data stream O. Formally, $f_f : S \rightarrow O$.

The name of fusion function is a simplified name to represent the computation process on the input data streams. It can be regarded as a black box, which can be filled with proper fusion algorithms or functions, according to different application scenarios. The usage of fusion function on the input data streams is one way of processing the data streams to generate the output data streams. The fusion function can simply aggregate to calculate the output based on the input, or can be classical data fusion techniques and algorithm, which are already introduced in Section 4.2. The fusion function can also be a combination of the typical data fusion techniques and the field knowledge related regulations.

Once this definition of data fusion unit is given, the different units are interconnected with each other by the data streams as input and output. Thus it is possible to map the system consisting of simple data fusion units and data streams into a complex network model.

4.4.2 Combine Data Streams and Fusion Components into Abstract Fusion Graph

The single data fusion unit can be modeled as a "black box", with several input data gates and several output data gates. The data fusion function located at the center of each "black box" processes the input data streams incoming from the input gates, and generates the output results to send out from the outgoing gates. Each fusion component is a complete fusion unit which can be deployed on a certain devices. The continuous processing of the data streams among different data fusion components, requires connecting one component with another component into a certain topology. The logic structure of automatic data fusion for data centric application can be described as a complex fusion graph. The fusion components can be regarded as vertices and the data streams among these fusion components can be regarded as directed edges of the graph.

We can use an example to illustrate the modeling of the data centric applications. Fig. 4.4 shows one application of processing measurement data of received signal strength to discover the radio coverage breaches. Further, this information is used to guide the recovery of the mobile communication networks. The application scenario is introduced in detail in Section 1.2.1. This is achieved through intensive data processing on incoming data streams from the data sources. At the time stamp t_1, the collected measurement data of received signal strength identification at different locations forms a low-quality radio measurement map. After fusing this measurement data, the middle layer map is generated, which reflects the current location of base stations. The top layer resource map reflects the coverage of the base station. At the time stamp t_2, all the three layers of resource map are also generated. Comparing the results at both time stamps, the change on the base station can be detected. The breach in the coverage can be used to guide further recovery actions. This is an application and example of using fusion functions to generate a group of layered **feature maps**.

In order to generate these several layers of feature maps, the fusion components can be deployed on different nodes and process automatically the incoming collected data. Fig. 4.5 shows the connection of relative fusion components to generate Fig. 4.4. On each node, the local measurement data and the exchanged measurement data are fused through **the distance-based conflict resolution function**. The output of this fusion function is the first layer of radio measurement map. This is the input of **the location estimation function**, the locations of base stations are demonstrated at the second layer of base station location map. This data map is again fed into **the fusion function of coverage estimation**. At the end, the upper layer of radio coverage map is generated. These fusion functions combined with gates for data streams are installed as software components on each node. Thus, it forms a data processing network aiming at automatic generation of the resource maps at different time interval.

The above idea of connecting fusion components with directed data streams to satisfy a specific application design aim can be generalized into the form where the fusion functions can be selected according to any application scenarios, such as Fig. 4.6. The interconnections between two fusion components or between data sources and fusion components are specified by directed data streams.

When further extraction made to these data processing flows, a directed fusion graph can be achieved. A directed graph or digraph is a graph, or set of nodes connected by edges, where the edges have a direction associated with them. In formal terms a digraph is a pair G=(V,E), where V is a set of vertices and E is a set of edges. It is also important to specify the special connection directions and the ways that the data streams flow in between the fusion components, such as *push&pull*, *publish&subscribe*, and *send&receive*. When the data connection captions are added into the directed graphs, the special form

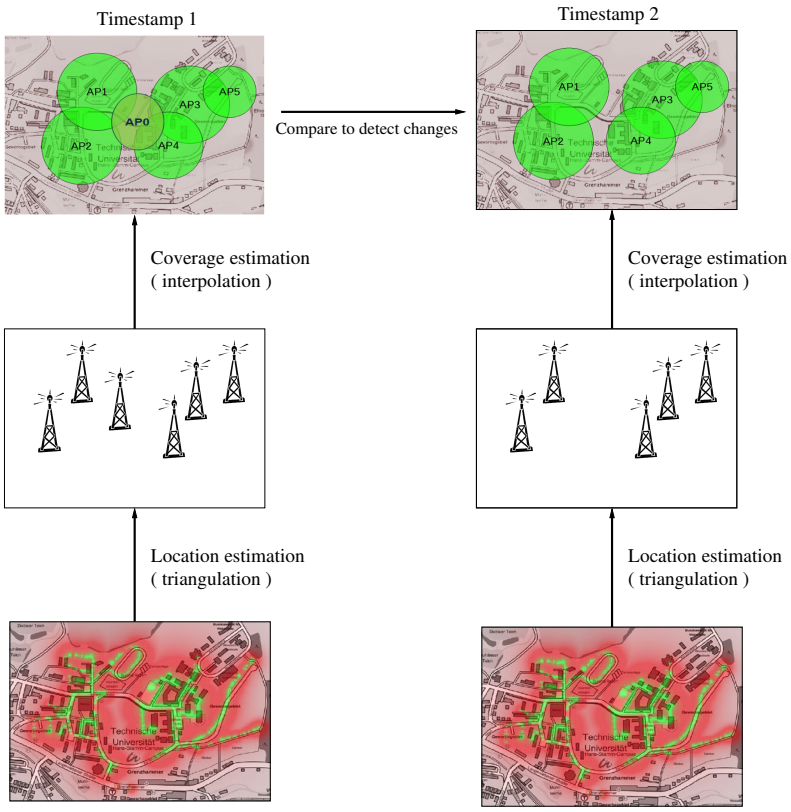


Figure 4.4: Processing Data Streams in Different Time Interval to Generate Layered Resource Map

of the directed graphs can fully represent the logic fusion structure. The resulting directed fusion graph is a novel model for the information (data) fusion.

4.4.3 Centralized vs Decentralized Directed Fusion Graph Model

When the data processing system is described into the form of directed fusion graph, it is easy to deploy such graph onto a single node. Programs generally process the incoming data, and then send the result data streams out. It is well represented in the UNIX operating system which pipes the data between small single-purpose tools.

This section considers a decentralized model to deploy such directed fusion graph onto several nodes instead of a single one. The interconnections among fusion functions can be within one node or among nodes. Fig. 4.7 shows how the centralized model on one node is connected with other fusion functions of the other directed fusion graphs on the other nodes. In this figure, the concrete lines represent the data exchange channels among fusion components within each node, the dotted lines represent the data flows among data fusion components on different nodes. For instance, the measurement data of Node 2 and Node 3 are transmitted to Node 1 (along the data exchange channels represented by dotted line), and fused by the Data fusion algorithm 1, at Node 1. The configuration and deployment of this directed fusion graph is a completely decentralized model. The flexible configuration and deployment of this decentralized model is the problem which next chapter mainly tackles. Although this is an important

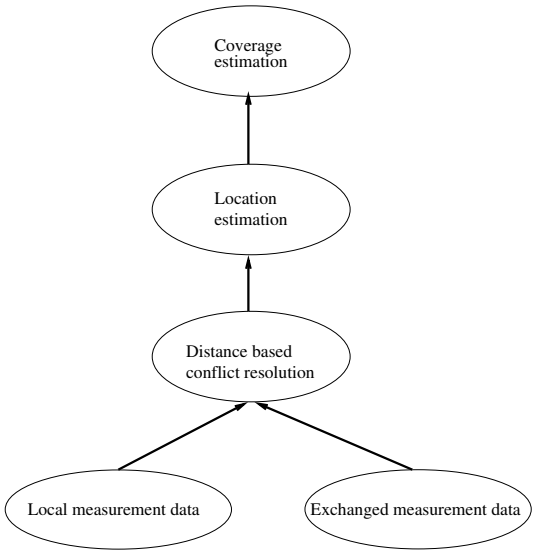


Figure 4.5: Example of Data Flows to Generate Layered Resource Map

problem for distributed fusion systems, very few previous work focuses on solutions of this configuration problem.

4.5 Enable Robust Information Exchange in the Directed Fusion Graph

The robust information exchange protocols represented by the gossip protocol family are shown to be effective and flexible in providing several options to satisfy the design requirements in extreme conditions such as disasters/emergencies. Chapter 3 explores in details the characteristics of applying robust information exchange protocols to adapt to dynamic environment, and the refined algorithms of target selecting to exchange the information among neighbors.

The application of target selective algorithm of gossip protocols brings new challenges to the architectural model of the Directed Fusion Graph: in the iteration of gossip round, the information is exchanged between different gossip partners. The architecture of the decentralized information fusion models is not static, but dynamic.

The results of the computational data fusion component cannot be directly connected to the other corresponding computational data fusion component. Instead, a new gossip component should be added to each device to control the changes in the topologies. This gossip component is not a traditional data fusion component which carries out computation tasks; instead, it only decides the updates of the interconnections among neighbors. This becomes a unique component to enable the dynamics in the interconnections, where no special fusion techniques or algorithms should be adopted at all. The are two phases required to enable gossip protocols:

- **Initialization phase:** the directed fusion graph is generated so that the gossip node has a push-pull data exchange link with all its neighbors. It is a set which contains all possible results of the *selectPeer*() function.

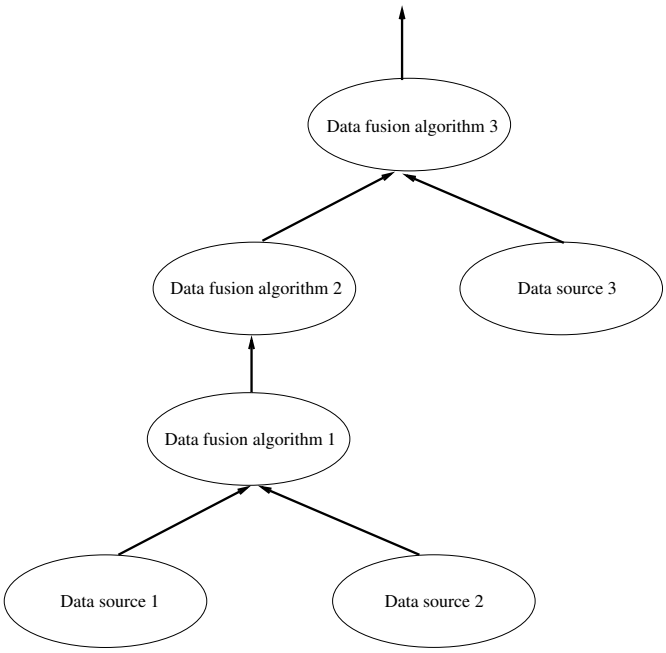


Figure 4.6: General Format of Data Processing Flow with Fusion Functions

- **Iteration phase:** during each iteration phase, only the data exchange link which matches the result of *selectPeer*() function at **that iteration** are temporarily set as valid to use. All the other initialized data exchange links set up during initialization phase are set to be false to use during this iteration. This is done by the extra gossip control components. Besides, the mode of the data streams is also set by this components, to be "push", "pull", or "push-pull".
  In each new iteration, the gossip control component needs to repeat the above set and unset process according to the return results of selectPeer(). The result of these steps decides whether a socket to be ready or not to send/receive data using certain modes. The details of mapping this result to be real ZeroMQ sockets is described in detail in Chapter 5.

These two phases enable a dynamic usage of data exchange links above a relatively stable fusion graph structure after initialization. It saves further effort and cost in re-initialization and maintaining the structure for dynamic requirements from gossip protocols.

4.6 Features of the Directed Fusion Graph

The model of directed fusion graph regards the fusion system as a directed graph which consists of fusion components interconnected with each other through special connections. Thus the fusion components (including special fusion functions and the data gates) are vertices and the connection sequences among the fusion components are the edges correspondingly. As for the specific fusion components, they are treated as a black box which connects incoming and outgoing data gates. As if the configuration and the interconnections is fixed from one fusion component to another component, the data can be

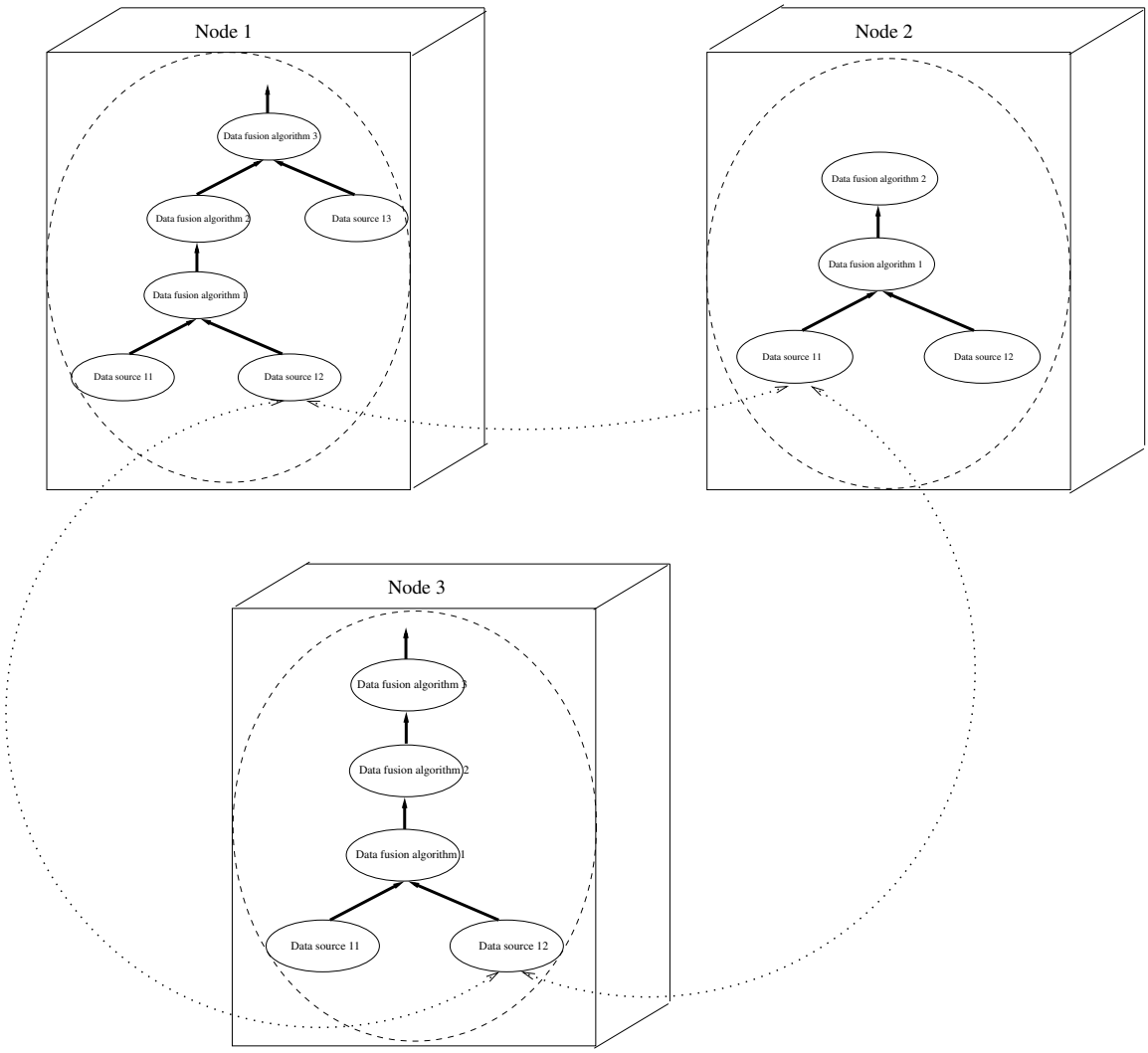


Figure 4.7: Decentralized Data Fusion Model

processed automatically as products passing through streamline. Comparing with existing data fusion models, the directed fusion graph model offers several unique features:

- The decentralized data fusion graph can be utilized **to model the distributed data fusion system**. This model provides an encapsulation design possibility for the fusion system. Designers of specific fusion tasks only need to consider the design specific fusion functions or adoption of specific fusion techniques. The system designers consider the sequence of connecting these fusion functions through the interfaces of data gates.

- The model provides the **flexibility to connect different fusion components**, and breaks the restrictions in the information-based model that different levels of fusion tasks cannot exchange data to execute fusion tasks across levels. For instance, in DFD model [15], Decision level can only get input from Feature Level and Decision Level, but cannot directly get input from the Data level. In Directed Fusion Graph Model, the fusion tasks are achieved with fusion components, and these

components can be interconnected in any order without considering which level the fusion tasks belong to.

- The model provides support **to enable various data exchange protocols with different modes**, instead of a simple passing mechanism. It does not restrict adopting specific fusion techniques to achieve certain goals. This makes the model more flexible to be applied to a wide range of fusion applications. The resulting model can be directly configured and deployed onto a real system without further design effort, as described in the following Chapter 5.

- It is a **general model** that aims to support various data centric fusion applications, instead of targeting on certain specific fusion applications. It extracts the most important factors to construct a fusion system: the different fusion tasks are injected into the vertices and the different connection methods as edges to adapt to the requirements of different applications.

- The model **allows dynamic updates of the fusion architectures**, by introducing special fusion components which control the information/data flows among fusion components on different devices. This allows the whole fusion architecture to be more suitable for dynamic environment.

These characteristics distinguish this model from the other existing fusion models in Section 4.3. In the following sections, several examples illustrate how data fusion components which combine typical fusion techniques and the field knowledge are picked. How these fusion components are added to the general directed fusion graph model, to satisfy the requirements of various applications. The examples start from data fusion in generating layered resource maps, to the decision fusion in decentralized change detection, and ends with a special fusion example of fusing and optimizing multiple queries. In these examples, the data fusion models in distributed fusion graphs, the fusion functions, techniques and the data types are described in detail. The features and classifications of the specific fusion functions are also introduced. How the data are transmitted among different fusion components is the content in the chapter 3, the following sections focus only on the fusion components in the three examples.

4.7 Use case of Radio Resource Map generation

As introduced in Chapter 1, a resource map is a typical feature map which is generated on wireless sensor networks. In wireless sensor networks, every sensor node can maintain a local partial view by measuring and sensing its local and nearby situation within its effective sensing coverage. When these partial views can be fused together, a global view of the entire area where the wireless sensor network is deployed can be generated. This generation process involves some conflicts resolution fusion process to improve the data accuracy, and some cooperation fusion process to estimate features based on local measurements.

In fig. 4.4, there are three different resource maps, based on the different sensing data: the lowest level is radio measurement map, the middle level is base station location map, and the highest level is the radio coverage map. The setcion introduces the fusion process from generating the three layers of resources maps from the lowest radio measurement map till highest radio coverage map.

There are different types of sensing data which can be used to generate various resource maps. This section use the generation of layered radio resource map(fig. 4.4) as one example to illustrate the data

fusion processes. The fusion functions used in the generation process of radio resource map is shown in fig 4.5. The principle used in this example applies also to other feature map generation with different data types.

Two types of fusion functions are mainly used in this process. The first fusion step is carried out to resolve the conflicts and improve the data accuracy among the radio measurements. The feature of this fusion function is that the output data structure is the same as the input data structure, but the data is filtered with improved accuracy or resolution. The distance based interpolation functions are applied to estimate on areas without measurement.

Another type of fusion process used in the radio resource map generation is vertical fusion. This fusion processes are done among several layers of resource maps in figure. 4.5. The input and output data of the fusion functions are of different data structures. The aim is to process from relative abstract measurement information to the more clear information to make decisions.

4.7.1 Example of Fusion Function: Estimating Locations

This section defines the data models for data measurement and aggregation using data fusion based on the received power from different base stations at different receiver positions. We use a free-space path-loss model and do not consider multi-path fading or shadowing, as depicted in equation 4.1:

$$p_R = \frac{p_S \cdot \lambda^2}{16 \cdot \pi^2 \cdot d^\alpha} \tag{4.1}$$

Here, p_R is the rx-power at the receiver and p_S is the tx-power at the sender. λ is the wavelength on which the transmitted signal is modulated. Using the carrier frequency f, λ, is calculated via $\lambda = \frac{c}{f}$ with c indicating the speed of light. The constant α determines, to which extent the environment dampens the signal through the space. The distance between transmitter and receiver is represented by d.

4.7.1.1 Radio Measurement Map

We define the triple $(p \in P, a \in A, x \in X)$ as a single measurement. P is the set of all points in the Cartesian $\mathbb{R}^2$ space, $A := \mathbb{N}$ stands for all possible sender MAC addresses and $X := \mathbb{R}$ contains all possible received power values. Any Measurement Map M is then defined as $M \subset P \times A \times X$. The set of all possible measurement maps is defined as $\mathfrak{M} = \mathcal{P}(P \times A \times X)$. Using $S := \mathbb{N}$, which represents the set of all sensors, it is possible to define individual measurement maps for each sensor. An example using two sensors, 1 and 2 and two base stations 3 and 4 may look like the following:
$M_1 = \{((0,5),3,0.1),((1,10),4,0.7),((10,17),3,0.05)\}$,
$M_2 = \{((6,3),4,1.7),((9,0),3,0.3)\}$.

Taking the radio fusion of location estimation as one example, the input data in the radio fusion problem is in the form of a Radio Measurement Map. It is based on the observations and measurements of the signal strength. Any Measurement Map M is then defined as $M = \{MeasurementID, GPS -$

Table 4.3: Example of Data Stream on Radio Measurement Map(Lowest Level Map)

MEASUREMENTID	GPS-lat	GPS-long	RSSI	BSID	Quality	timeStamp
M_1	50.68047	10.93277	82	B_1	0.8	6049794
M_2	50.680269	10.930667	10	B_1	0.4	6049794
M_3	50.680269	10.930667	95	B_2	0.55	6049794
M_4	50.680721	10.934009	84	B_1	0.8	6049794
M_5	50.680218	10.930629	8	B_1	0.2	6049794
M_6	50.680218	10.930629	96	B_2	0.9	6049794
M_7	50.679923	10.931107	15	B_1	0.4	6049794
M_8	50.679923	10.931107	90	B_2	0.84	6049794
M_9	50.680582	10.931697	20	B_1	0.5	6049794
M_{10}	50.680582	10.931697	80	B_2	0.6	6049794
M_{11}	50.680069	10.93373	94	B_1	0.87	6049792
M_{12}	50.680069	10.93373	20	B_2	0.6	6049792
M_{13}	50.68047	10.93277	80	B_1	0.85	6049792
M_{14}	50.68047	10.93277	30	B_2	0.45	6049792
M_{15}	50.680721,	10.934009	82	B_1	0.75	6049792
M_{16}	50.680721,	10.934009	10	B_2	0.66	6049792
...	...	...	...	...	...	...

Table 4.4: Example of Estimation on Base Station Location Map(Middle Level Map)

BSID	GPS-lat	GPS-long	timeStamp	Estimation-Accuracy
B_1	50.680282	10.933735	1336049794	0.85
B_2	50.680171	10.930957	1336049794	0.75
B_1	50.680276	10.933730	1336049792	0.80
B_2	50.680170	10.930954	1336049792	0.72
...	...	...	...	...

$Latitude, GPS - Longitude, RSSI, BSID, MeasurementQuality, TimeStamp\}$. In this input Measurement Map, for each measurement, MeasurementID is the key for this row of measurement value. GPS-Latitude and GPS-Longitude are the location of the address where the measurement is taken. RSSI is received signal strength identification; BSID is the unique identification for the base station.

At a certain time, the measurement example using two sensors, 1 and 2 and two base stations B_1 and B_2 may look like the following:

$M_1 = \{50.68047, 10.93277, 82, B_1, 0.8, 1336049794\ \}$,
$M_2 = \{50.68047, 10.93277, 30, B_2, 0.7, 1336049794\ \}$,
$M_3 = \{50.680269, 10.930667, 10, B_1, 0.67, 1336049794\ \}$,
$M_4 = \{50.680269, 10.930667, 95, B_2, 0.55, 1336049794\ \}$,

The global measurement map $M_g = \bigcup_{s \in S' \subset S} M_s$, contains the measurements of all sensors. These mea-

surements represent the received signal strength at the position of the measurement sensors, which provides only a vague information of the location of a base station. In order to get the accurate location of the base stations, the fusion process should be used to generate the output, so called "Radio Resource Map". The data type is illustrated in Tab. 4.3.

A more sophisticated map, the output *Radio Coverage Map* as in Fig. 1.1(b) is derived from the radio measurement map and shares some structural similarities with it. The difference is, that $MeasurementQuality$ is replaced by $EstimationQuality$, indicating the quality or accuracy of the estimated base station location. Any element O is defined as $O = \{BSID, GPS - Latitude, GPS - Longitude, TimeStamp, EstimationAccuracy\}$. The resource map may also be defined per-sensor via S or globally, via O_g. Table 4.4 shows the example of a resource map.

The interesting question is now how to transform from M to Q.

4.7.1.2 Base Station Estimation Function

A data fusion function $f_F : \mathfrak{M} \rightarrow \mathfrak{O}$ is required to transform a measurement map into a corresponding base station location map. An example of a simple fusion function is given by Equation 4.2:

$$f_{p_R,Max}(M \in \mathfrak{M}) = \left\{ (p,a,q) \,\middle|\, \begin{array}{c} (p,a,x) \in M \wedge \\ q = max(\pi_X(\sigma_{A=a}(M)) \end{array} \right\} \tag{4.2}$$

Here, we simply select the measurement with the maximal received power and declare it as the location of the base station. This approach is near the base station, but it is limited to the measurement with the smallest distance to the base station. hence, the sensor needs to "hit" the base station for an exact result, which is not sufficient. To overcome this problem, a more sophisticated approach needs to be taken into consideration. It is called the "probabilistic circle intersection". This method uses a radio measurement map $M \in \mathfrak{M}$ in order to compute the distance d for each measurement $m \in M$ using the following equation 4.3:

$$d = \sqrt[\alpha]{\frac{p_S \cdot \lambda^2}{16 \cdot \pi^2 \cdot p_R}} \tag{4.3}$$

We assume, that α, and λ are known for the given environment and for the radio configuration of the transceiver/receiver. Having d, the receiver knows at which approximate distance the corresponding sender is located. We can draw a cycle around each measurement location. Since all points on a single circle are candidates for the base station location, additional intersecting circles are required. The minimum requirement of inter-sectioning circles is 3. Due to problems like inaccuracy of localization, the simple circle intersection as shown with the base station a_1 and the measurements m_1, m_2 and m_3 in figure 4.8 is not sufficient to calculate the base station locations.

Having a higher number of measurements may help to overcome the problem of localization inaccuracy. We use a probabilistic approach to fuse a huge amount of circle intersections in order to get the base station locations. This approach defines a set of circle C with $f_C : \mathbb{R}^2 \rightarrow \mathbb{R}$, representing the measured data. The circle intersections are derived from this data using the function $f_{CI} : C^2 \rightarrow \mathcal{P}(\mathbb{R}^2)$. The

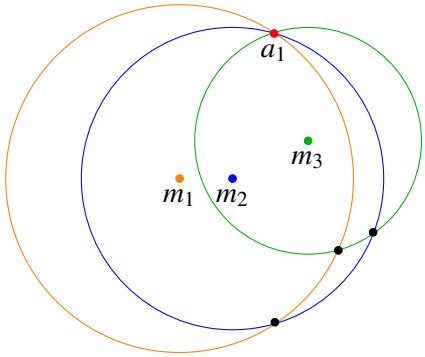


Figure 4.8: Probability Circle Intersection

characterizing function of $f_{CIC} : C^2 \rightarrow \{0, 1, 2, \infty\}$ is defined as the following, with $d = \|p_1 - p_0\|$:

$$f_{CIC}(c_0, c_1) = \begin{cases} 0 & d > r_0 + r_1 \vee d < |r_0 - r_1| \\ 1 & d = |r_0 - r_1| \\ \infty & c_0 = c_1 \\ 2 & \text{otherwise} \end{cases} \tag{4.4}$$

The set of circle intersections, I is directly calculated using the set C and function f_{CI}, as $I = \{p | c_i, c_j \in C \wedge i \neq j \wedge p \in f_{CI}(c_i, c_j)\}$. In the most cases, f_{CIC} will yield 2, for exactly two intersections or 0, for exactly no intersection. The most interesting case for us is the one where f_{CI} yields 2, where one intersection point is likely to form a cluster near the real position of the base station. The center or site of this cluster is then very likely to be the position of the base station. This position is depicted by a red dot in Figure 4.8. The other results of the circle intersection are marked by a black dot in the same figure.

4.7.2 Feature of the Resource Map Fusion Models

The fusion models used in the resource map generation is a computation centric. This is a typical data fusion function, to complete a computation task based on the input data and generate results to output and send through the outgoing gate. The fusion component is distributed on different sensor nodes to collect and process the measurement data, different types of fusion tasks are executed here:

- Complementary data fusion: to enlarge the knowledge range and generate a global knowledge by integrate different pieces of local measurement together.
- Redundant fusion: some measurements from the two sensor nodes in the neighborhood provide the same piece of information, these pieces of measurement in the inter-lapping area can be used to increase the associated confidence.
- Cooperative fusion: the location estimation fusion function executes cooperative fusion task. This fusion function chooses the top three measurement of RSSI from neighborhood measurement and uses the three different measurements to triangulate to estimate the location of the access point.

This fusion model represents a combination of feature level fusion leads to decision level fusion, showing that the directed fusion graph can represent and deal with this traditional DFD fusion model.

4.8 Decision Fusion Example of Change Detection in WSN

[87] introduces the application scenario for decentralized change detection in wireless sensor networks based on DFT based synopsis. A decision fusion component is used to fuse the local change detection results, to achieve consensus among different detection results and to improve the change detection accuracy.

4.8.1 General Fusion Graph Model of Decentralized Change Detection

Let $S = \{S_1, ..., S_M\}$ be a set of data streams of the environmental observations sensed by M nodes in a sensor network. Let $S_i = \{x_1, x_2, ...\}$ be a time series data stream of observations at the i^{th} sensor node and each incoming data item x_j arrives in increasing order of j. Additionally, each data item x_j is generated with some distribution. Time series data streams are suitable for modeling the data produced by a sensor because the observed measures are updated every time unit (milliseconds, seconds, etc).

We assume that each node in the network has the ability to detect the changes of the environment within its vicinity. If a change occurs in the environment that is close to some node in the sensor network, the tasks of the sensor network are to detect and to report the change of environments quickly by using small amount of memory, yet to assure some certain accuracy. Therefore, the problem of change detection in sensor network can be decomposed into two sub-problems as follows.

- *Change Detection in a single data stream*: Due to the unlimited nature of data streams and the limited memory of sensor, the sliding window-based approach is the best choice for the change detection algorithm in the data stream. The changes are observed in a sliding window of N points $\{x_0, x_1, ..x_{N-1}\}$. Let W_1 and W_2 be two corresponding synopsis structures which are constructed from two windows $w_1, w_2 \in R^b$ as basic sliding windows of size b . Now, the task is to test the two following hypotheses:

$$\begin{cases} \widetilde{\mathcal{H}}_0 & d(W_1, W_2) \leq \Omega \\ \widetilde{\mathcal{H}}_1 & d(W_1, W_2) > \Omega \end{cases}$$

  where $d(w_1, w_2)$ denotes a distance function which measures the dissimilarity of two sliding windows and Ω is a distance-based threshold used to make a decision when the change occurs or not.

- *Decision Fusion:* Next, we assume that each node in the sensor network has reached the final decision u_i which is the final decision of the local change detector at node i. The final decision at node i is propagated to the other nodes of the sensor network, based on the received information from node i, these nodes also make local decision themselves. All the decisions are transmitted to the sink node in order to get the global decision.

The local change detection on single data stream is to explore the time correlation of the measurement from a single data source, while the decision fusion process is to explore the spatial correlation of the local detection results from several different sources. The aim of the decision fusion is to improve the detection accuracy, and to offset the local detection failures.

4.8.2 Fusion Function

The data processing in the decentralized change detection frameworks consists of two-folded data processing functions: local change detection function and the global decision fusion function.

4.8.2.1 Local Change Detection Function

Locally, the change detection function is applied on the single data stream using a DFT coefficient as synopsis. As shown in Algorithm 3, the factors affecting the performance and accuracy of a change detector include the sliding window, the basic windows, how to extract synopsis structures from the basic windows, the distance function, and the threshold. These factors will be discussed more detailed below.

Algorithm 3 Change Detection Algorithm Using DFT as Synopsis Structure

Step1: **for** each *sliding window* **do**
    t = 0
    window w1=first b points from time t;
    W1=DFT(w1);
    window w2=next b points in data stream;
    W2=DFT(w2);
end for
Step2: **while not** at end of stream **do**
  for i = 1...N **do**
    if $d(W1, W2) > D*$ **then**
      t = current time
      report change at time "t" ;
    end if
    w2=sliding(w2,1);
    y=lastItem(w2);
    W2=Sliding_DFT(W2,x,y);
  end for
end while

When data streams are modeled as time series, physical windows (time based windows) are a natural choice. Because the sliding window is the most widely used and the most general one in the streaming data context, our work focuses on comparing two sliding windows in order to detect the changes that occur.

We use the sliding window framework proposed by Shasha et al [102] for detecting changes in data streams. In this approach, a sliding window of size N is divided into $\left\lceil \frac{N}{b} \right\rceil$ basic windows of size b.

Changes in the data stream are observed over a sliding window. For a basic window of width b, changes can be observed in a sliding window $[b, b + N - 1]$.

To detect change in a sliding window, we compare a basic window $w1$ used as reference window with another basic window $w2$ moving on the data. Hence, depending on how to choose the reference window, the position correlation between the reference window $w1$ and basic window $w2$, as well as how the windows $w1$ and $w2$ move, we can develop different algorithms for change detection.

In our change detector, we fix the reference basic window while the sliding window moves step by step without considering the change occurs.

As such, the performance of local change detector depends on how to choose the sliding window and the basic windows. Depending on the specific applications, we can select a suitable distance measure. The Euclidean distance function is preferred to other distance functions, because it is preserved under orthonormal transforms such as Fourier transform. Additionally, the Manhattan distance is commonly used in some applications. In this work, we examine our change detector with the Euclidean and the Manhattan distance functions.

A threshold is a value specified by user or automatic procedure in order to distinguish the state 'Changed' or 'Unchanged' of an event. As the balance between sensitivity and robustness of the change detection algorithm is partly determined by the threshold, its selection is a critical step for successful change detection algorithms. The goal of threshold selection is to choose the threshold in such a way that both the probability of false alarm and the probability of mis-detection are minimized.

The choice of the threshold depends on the specific context of each application. Therefore, prior knowledge about the detection problem needed for threshold becomes meaningful. For example, change detection of battery-power level of sensors in sensor network, threshold is fixed and previously given.

Another issue is how to extract DFT coefficients as synopsis structures from two basic sliding windows. The goal here is to exploit the properties of Discrete Fourier Transform to find the DFT coefficients used in change detectors as quickly as possible. This topic is discussed in many books on the digital signal processing [42].

Discrete Fourier Transform is used to extract the features from data streams because the characteristics of DFT allow us to reduce the memory used by local change detector. The reduction in memory comes at the price of the accuracy, but this trade-off is acceptable in the restricted sensor resources. The Euclidean distance is preserved under the Discrete Fourier Transform. Therefore, instead of computing the Euclidean distance on two windows of N samples, the Euclidean distance between two windows w1 and w2 can be computed from the DFT coefficients extracted from two corresponding windows. Additionally, most time series data streams are a real sequence, then $X(i) = X^*(M - i), i = 1, ...M - 1$ where $X*$ is the conjugate of the complex number X. As such, instead of computing N DFT coefficients, DFT algorithm only requires to compute $N/2 + 1$ DFT coefficients. For example, if the width of sliding window size is $N = 1024$, then the number of DFT coefficients is only 513. Furthermore, since the energy of the time series data stream only concentrates on the first few DFT coefficients, we then need to capture K most important coefficients where $K \ll N$. As such, if the width of sliding window is considerably larger, the performance of DFT-based change detector improves significantly in terms of memory consumption. For instance, [103] explains that, it is possible to use a few DFT coefficients ($K = 40$), extracted from a very large window ($N = 4million$), in the compressive sensing process. This conclusion shows the advantage of applying DFT in designing change detectors on the data streams in

wireless sensor networks. To compute DFT coefficients directly, we need $O(N^2)$ units of time, but with Fast Fourier Transform this can be reduced to $O(NlogN)$. To further improve the speed of change detector, algorithm 3 is used to reduce the time required for computing the DFT coefficients. Specifically, instead of computing the DFT coefficients of the basic window w_2, we use an incremental strategy to compute the DFT coefficients of the basic window w_2. Fast incremental processing of new incoming data items arises from the nature of on-line processing of data streams. Incremental processing is based on the evaluation of a high-performance function used for placing new points. This function decides new data items as inputs without comparing the new data items with all the old data items that have been processed.

The distinction between the change detection algorithm without using synopsis and change detection algorithm using DFT as synopsis is that the distance measure in the former is directly computed on the samples of two sliding windows w_1 and w_2 while the distance measurement in the later is computed on the DFT coefficients by procedure *DFT(w).* Therefore, these procedures can be considered as the heart of this DFT-based change detection algorithm. Procedure $Sliding\_DFT(W2, x, y)$ is to compute the DFT coefficients in the window w2 incrementally. $Sliding\_DFT$ computes the new DFT coefficients in the window $w2$ based on the old DFT coefficients, the old item coming out from the window $w2$ named x, and the recently incoming item called y.

Compared with the non-synopsis change algorithm (N samples), the DFT-based change detection algorithm ($K \ll N$ DFT coefficients) reduces the amount of required memory. This local fusion function represents an example of processing data with signal processing function.

4.8.2.2 Global Decision Fusion Function

The local result of change detection with DFT synopsis has to be fused and forwarded to the sink in order to implement a global change detection decision. The local change detection result $\langle Change\ Detection \rangle_i$ on each sensor node $Node_i$ has its own data structure as follows: $\langle Change\ Detection \rangle_i = \langle Estimated\ Event\ Position, Sensor\ Location, Detection\ Result, Detection\ Accuracy \rangle$, where:
$\langle Estimated\ Event\ Position \rangle$: $\langle X_i, Y_i \rangle$,
$\langle Sensor\ Location \rangle$: $\langle x_i, y_i \rangle$,
$\langle Detection\ Result \rangle$: `enum(-1, 1, N/A)`,
$\langle Detection\ Accuracy \rangle$: `float[0,1]`.

Among them, the sensor location is from a GPS or other positioning algorithm for GPS-less sensor networks, such as Triangulation [86].

The estimated event position is assumed within the sensing range of the sensor node. With the GPS knowledge of the sensor location, the sensing range is usually regarded as coverage shaped in cycle, with sensor location as center. In a different environment, the estimated event position may suffer from interference of obstacles. But with a proper technique to aggregate several results from different sensor nodes, it is possible to overcome this and generate a more accurate position of the event. The detection result depends on the distance from the sensor to the event. Some sensor may not be able to detect any changes, which results in N/A (not available). For the events which are within the sensing range of the sensor nodes, they either result in 1 (change detected) or -1 (nothing happens).

The detection accuracy is based on the detection record of the sensor node. When we use the DFT

synopsis based local change detection, the detection accuracy is automatically generated as a result of local detection. The global event detection decision $\langle Event\ Detection \rangle_{global}$, which fuses the local detection results $\langle Change\ Detection \rangle_i$ from each sensor node, has the following structure:

$\langle Event\ Detection \rangle_{global} = \langle Estimated\ Event\ Position, Detection\ Result, Detection\ Accuracy \rangle$
$\langle Estimated\ Event\ Position \rangle$: $\langle X, Y \rangle$
$\langle Detection\ Result \rangle$: 1, -1
$\langle Detection\ Accuracy \rangle$: `float[0,1]`
The question is how to fuse the local change detection $\langle Change\ Detection \rangle_i$ on each individual sensor nodes to generate the global fusion decision $\langle Event\ Detection \rangle_{global}$. In order to answer this question, the decision fusion needs to address three aspects:

- *Decision fusion model* runs on local sensor nodes to fuse results of change detections from different sensor nodes. The decision fusion model runs as plug-ins on each sensor node, which can be implemented according to different requirements. This decision fusion model is application specific.

- *Algorithm to cluster the sensor nodes* near the event location. The decision fusion process should be finished near the event location, in order to reduce communication cost, and it only makes sense to fuse the local change detection results which are within the detection range of supposed event location. In other words, it is necessary to cluster the sensor nodes which are around the supposed event, and to has the detection result either -1, or 1. Furthermore, this cluster algorithm should be triggered by the first sensor node which detects event.

- *Information exchanges and disseminates* from information source to sink. The information exchanged can be a confirmed warning or unconfirmed local change detection results. The transmission path is decided by the location of the source and the sink, where the sink is usually the final decision maker to take certain actions. In some multi-source single-sink scenarios, it is efficient to form a hierarchy such as tree structure to transmit information from the cluster head of information sources to the sink. In multi-source multi-sink scenarios, the problem is changed to disseminate the decision across the sensor network. In this case, protocols which can reduce redundant messages should be adapted. In the sensor actuator network, where the sensor is both the detector and the actuator, the sensors can immediately takes actions. So, the information transmission path from source to sink has the length of 0 hop. Therefore, the information exchange protocol should also be application-specific, and depends on the relative location of source(s) and sink(s).

We cannot enumerate all the combinations of the framework. In the following, we will describe an example of detecting a fire event, where the decision fusion model consists of event location estimation and the warning decision fusion. We consider a multi-source multi-sink information dissemination scenario, where all the sensor nodes should be able to react to a fire detection warning. We will explain relative models, algorithms and protocols in the following sub-sections to illustrate how the framework functions based on local change detection results.

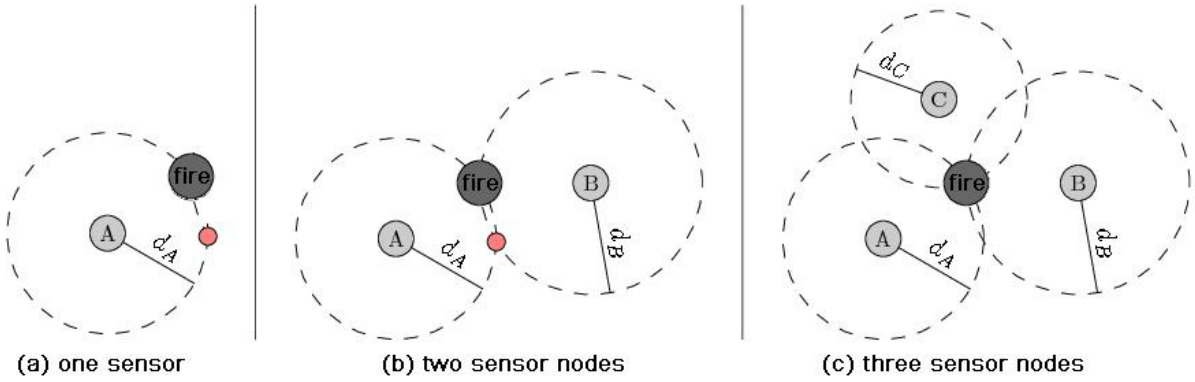


Figure 4.9: Location Estimation with Different Number of Sensor Nodes

4.8.3 Decision Fusion Model

As mentioned above, the decision fusion algorithm should run on each sensor node as a process to compute and aggregate change detection results from different sensor nodes. This decision fusion helps to generate a more accurate report on events, which basically uses quantity to compensate for the lack of quality. The input of the decision fusion function is the local event detection result, while the output should be global change detection result.

4.8.3.1 Location Estimation

Depending on requirements on estimation accuracy and also the available information on nearby sensor nodes, a different location estimation model may be implemented and installed on sensor nodes. Some techniques use triangulation to estimate the event location, while some consider the interference of obstacles and result in a region where the anomaly event occurs [26]. In a scenario where a fire location should be estimated, all the information and local results of change detection on sensor nodes whose sensing range covers the event should be considered and aggregated. We assume the intersection of sensing coverage of the sensor nodes which reported 1 is the fire location, as figure 4.9 shows. With more results from sensor nodes to aggregate the location, it is possible to improve the accuracy of the final location estimation.

4.8.3.2 Warning Decision Fusion

In order to overcome the occasional local failure either in value or in change detection, the sensor nodes should aggregate the local detection result with a decision model. The decision aggregation model works as a plug-in according to different applications. In the scenario of fire detection and warning, we use a history detection accuracy-based weighted method to fuse all the detection results near the reported event location. In this case, each sensor node $Node_i$ maintains a queue for all the detection results $\langle Change$

$Detection\rangle_j$ near the supposed fire location and uses the detection accuracy as weight. Then, the sensor node can generate a global fusion result with the following function:

$$FusedDetection_i = \frac{\sum Accuracy_j * DetectionResult_j}{\sum Accuracy_j}$$

Once the $FusedDetection_i$ is larger than the threshold, in this case larger than 0, a new detected change alarm will be generated from $Node_i$. This is a continuous fusion process: With the $Node_i$ receiving more messages from neighbors, the fused result will become more accurate.

The local detection result of the warning should be disseminated across the network. In order to reduce the redundant messages, we employ gossip protocols. In the basic push-pull-gossip scheme, each node executes both active and passive threads. The active thread is executed once in each consecutive time units at a randomly picked time. The passive thread always receives data and then sends its own data to the neighbors from whom it receives information. In push-gossip scheme, the new information owner pro-actively sends its new data to its selected neighbors, but does not exchange information with the selected neighbors. While in pull-gossip scheme, the new information owner waits passively until it receives the requests asking for data. Chapter 3 discusses different features of these gossip protocols with details. We can choose "push gossip", "pull gossip", "pushpull gossip", and application layer flooding as plug-ins to exchange data among these two different fusion functions.

4.8.3.3 Decision Fusion on Local Change Detection Result with Gossip Protocols

The connection in between local change detection function and global decision fusion function with gossip protocols forms a generalized fusion graph. The graph can also be described using algorithm. Algorithm 4 and algorithm 5 give an example of using algorithm to describe the fusion graph. The scheme is "pushpull gossip", where both active thread and passive thread collect the local change detection results from their neighbors. Next, they apply the decision fusion model on the local change detection results and its own detection result to achieve a fused decision. The decision fusion model can use history-based weights or weights on the distances to the event location.

Algorithm 4 Decision Fusion Using Gossip to Exchange Information: Active Thread

begin *active thread:*
do once in each consecutive iteration:
q= SelectNeighbors();
$status_p$= $changedetection_p$;
Send($status_p$) to q;
$status_q$= Receive($changedetection_q$);
$fusedDetection_p$=
DecisionFusion($changedetection_p$,$changedetection_q$);
$status_p$= UPDATE($fusedDetection_p$);
end *active thread*

Algorithm 5 Decision Fusion Using Gossip to Exchange Information: Passive Thread

begin *passive thread:*
do forever:
  $status_q$= Receive (*);
  $status_p$= $changedetection_p$;
  Send($status_p$) to sender($status_q$);
  $fusedDetection_p$=
    DecisionFusion($changedetection_p$,$changedetection_q$);
  $status_p$= UPDATE($fusedDetection_p$);
end *passive thread*

4.8.4 Feature of the Fusion Model of Decentralized Change Detection

The fusion model of decentralized change detection includes two levels of fusion tasks: one fusion task runs on the local sensor nodes to execute local change detection. This task runs on a series of windowed measurement data, thus can suffer delays. The fusion function at this level is computation oriented. This computation is unique in that it does not process data from different sources; instead, it processes the data from the same sources within a time frame. This is to explore the time correlation of data. At a higher level, the decision fusion function is a typical redundant fusion, to explore the spatial correlation of the data from different sources. The purpose is to improving the accuracy of the decisions.

4.9 Example of Query Fusion

The application scenario of query fusion is introduced in detail in Chapter 1, section 1.2.3. More details of the research can be found from a joint publication with Tianli Mo, Lipyeow Lim, and Misra Archan in [95]. The problem is to process while reducing the total volume of sensor data. The methodology applied is to opportunistically identify and execute the shareable parts of multiple continuous queries on a single "leader" mobile device. This device then distributes the intermediate query execution results to the other devices executing their queries, instead of having each smart phone retrieve its data streams independently. In practice, there are multiple mobile devices selected for executing different shareable portions of the queries, and the role of the "leader" is rotated among the group members for fairness.

From the point view of data fusion, this application requires three-folded data fusion: the first type of data fusion is to fuse the different queries from various mobile phones of the information demander, this results in the maximum common sharing query. The second type of data fusion is to "fuse" the data and the shared query at "leader" devices, this is in fact the data query processing. Finally, the third type of data fusion is to process on each query initiator the non-sharable query part to provide unique answers to individuals.

4.9.1 Input Data Type and Output Data Type

The input data type for the query fusion function is the various queries coming from different mobile devices, at the "leader" devices, fusion function extracts the maximum common part of these queries and

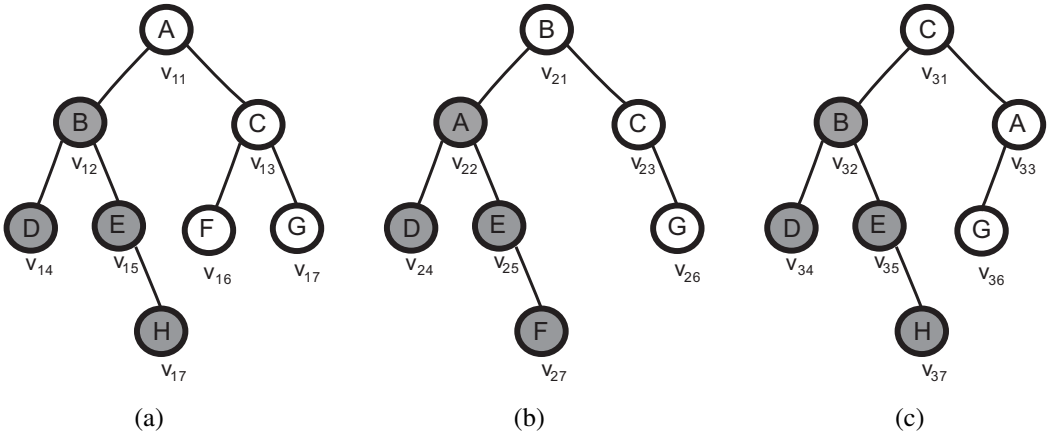


Figure 4.10: Examples of Matching and Non-matching Sub-query Tree. Figure 4.10(a) is a matching sub-query tree of Figure 4.10(c) but not a matching sub-query tree of Figure 4.10(b)

forwards the result to the query executer. Thus the input of the query executer is the maximum common sharable query, and also the mobile sensing data coming from different sources. The output data of this fusion function is the middle result of the queries, which can be used for answering several queries. This middle result is the input data for the last fusion function on the individual mobile node, where the non-sharable query part is fused with the individual sensing data. The output data is generated as the final results to reply to the individual queries.

This special query fusion has the input data type of queries, after the initial process, these queries can be transformed into query trees. These groups of input query trees are fed into the fusion function to extract the maximum common fusion tree as output data type. Fig. 4.10 shows three input query trees extracted from the queries of different smart phones. Some subtree of the query tree may match the other sub-query tree extracted from another query tree. However, even the three sub-query trees marked in gray are of the same structure, they cannot be regarded as the common subtree of these three trees. Because the node label which represents the query operator should also match to become a sharable part. Thus the gray marked in query tree 4.10(a) is a matching sub-query tree of Figure 4.10(c) marked in gray, but does not match the sub-query tree in gray in Figure 4.10(b).

4.9.2 Query Fusion: Finding Query-based Groups

A challenge of this query fusion function is to identify the shareable, common parts of multiple queries. This section describes the algorithm to address this challenge. The input of the algorithm is a collection of queries provided by the collection of phones in a given location-based group. The j-th query provided by phone i is denoted by $q_{i,j}$. The output of the algorithm is a collection of query-based groups and the common sub-queries associated with each of the query-based group. Note that, to avoid redundant query execution and energy overheads, the query-based groups form a partitioning over the set of queries $\{q_{i,j}\}$, for instance, each query should not be in more than one query-based group.

A query $q_{i,j}$ is a tree of relational operators. More formally, a query is a tree (V, E) where V is a set of nodes and E is a set of edges. A leaf node represents a data source and a non-leaf node represents

a relational operator. A directed edge between two nodes indicates the direction of data transfer. Each node $v \in V$ is associated with a name and a parameter: $name(v)$ denotes the name of the operator or the name of the data source, and $param(v)$ the parameter that the operator requires. For data source nodes, the $param(v)$ consists of the information required to retrieve data from that data source. For the relational selection operator σ, the $param(v)$ would be the selection predicate. For the projection operator π, the parameter would be the set of columns to project. Some relational operators such as $\times, \cup, \cap$ do not take any additional parameters, hence their $param(v)$ would be empty.

Two nodes u, v are **name matching** if $name(u)$=$name(v)$. Two query subtrees rooted at u and v are name matching if the two trees are isomorphic (there exists a bisection between the two trees) and each pair of corresponding nodes are name matching. Two nodes u, v are **exact matching** if $name(u)$=$name(v)$ and $param(u)$=$param(v)$. Two query subtrees rooted at u and v are exact matching if the two trees are isomorphic and each pair of corresponding nodes are exact matching. The reason for distinguishing the weaker name matching is that it is often possible to obtain a common sub-query that contains nodes that are only name matching with a relaxation on the parameter associated with the node. An example of such relaxation would be two queries p and q containing a selection operator on the same data source. The selection parameter in p is HR>95 and in q is HR>120. A common sub-query would contain the selection operator with parameter HR>90.

Algorithm 6 Strategy 1: Finding Query-based Groups

Input: A set of queries $Q = \{q_{i,j} : j\text{-th query of phone}i\}$
Output: A set of query-based groups and the associated shared subqueries
begin *queryBasedGroups:*
Find collection of shared subqueries using [48]
Find query-based groups using a greedy set cover algorithm
Remove overlaps between groups
end *queryBasedGroups*

There are several strategies for finding query-based groups. These strategies construct the query fusion functions. The first strategy is outlined in Algorithm 6. We first find a collection of shared sub-query trees given all the queries in the location-based group. Those shared sub-queries are maximal sub-query trees or forests that are shared by two or more queries. The grouping of queries induced by the shared sub-queries may not cover all queries, and may overlap. Since the query-based groups need to cover all the given queries, we find a minimal set cover using a greedy algorithm. As a set cover yields query-based groups that may still overlap, we then ensure that each query is covered by exactly one query-based group.

Finding a collection of shared subquery trees is done using a bottom-up algorithm similar to [48]. Algorithm 8 is one example algorithm to figure out the sharable sub-query trees. An inverted index is constructed that maps leaf nodes (name and parameter) to a list of queries. Starting from the largest list, each pair of queries in the list is examined to find the shared sub-query tree. If that particular shared sub-query tree has been found before, the pair of queries is merged with the existing list for that shared sub-query tree.

The second strategy is outlined in Algorithm 7. We first apply a hierarchical agglomerative clustering algorithm to group the queries using a distance function based on the number of shared data sources

Algorithm 7 Strategy 2: Finding Query-based Groups

Input: A set of queries $Q = \{q_{i,j} : j\text{-th query of phone}i\}$
Output: A set of query-based groups and the associated shared subqueries
begin *queryBasedGroups:*
Find query-based groups using a hierarchical agglomerative clustering algorithm
for all query-based groups
Find the shared subquery forest
end for
end *queryBasedGroups*

(exact matching on leaf nodes) between two queries. The stopping criterion can be based on the number of clusters or the size of clusters (more sophisticated conditions can be used too). Those clusters yield the query-based groups; a tree isomorphism based algorithm is then used to find the shared sub-query tree or forest for each query-based group.

These query processing strategies and algorithms form the general data fusion components of query fusion applications. All these fusion functions for queries should run on the "master" phone, to extract the maximum common sharing part of the queries. As the other fusion functions are to actually process data according to the queries and return the query results. They are relatively common, thus this section does not describe them in detail. Rather, this query fusion part is a very unique form of data fusion.

4.9.3 Five Query Processing strategies

To collaboratively share resources for query processing, mobile devices or smart phones need to be grouped into clusters. The query fusion framework uses a two-level grouping of phones. The first level of grouping clusters phones using the GPS location of the phones. The resultant groups are called *location-based groups*. The exact distance threshold depends on the type of applications to be supported and on the communication link used, for instance, for Bluetooth-based intra-group communications, all members of a group can be no more than 10 meters apart. The second level of grouping clusters the phones within each location-based group using the amount of commonality of the queries in the group. The resultant groups are called *query-based groups*. A location-based group can contain multiple, overlapping query-based groups.

Both location-based and query-based groups are managed by the server. All phones periodically send their location data to the server, which clusters the phones into location-based groups and manages the group membership when phones leave or new phones join the system. Similarly, all phones periodically send their queries to the server and the server will cluster the phone-query pairs into groups and manage the group membership.

- **Naive-P** Fig. 4.11(a). At each query execution, each phone sends requests for remote data to server. The server transmits the required remote data to each phone. Each phone acquires the local sensor data (e.g. accelerometer). After all required data have been received, the phone executes the query and obtains the result.

Algorithm 8 SharableSubQueryTrees(Q)

Input: A set of query trees Q
Output: $\mathcal{T}$ is a set of the maximal common sub-query tree with associated queries list
Let D be the set contains all the leafs name of Q
begin *SharableSubQueryTrees(Q):*
Step1: Scan through Q and construct an inverted index I that maps a leaf node'*name* to a list of query IDs
Step2: Sort I according to the size of the query ID list
Step3: Discard node whose *name* in I that occur in only one query tree
for each *Query ID list* Q_{idx} **do**
for each query tree pair (p_i, p_j) in Q
if done(p_i, p_j)
next;
end if
done$(p_i, p_j) \leftarrow$ true
$Na \leftarrow name(\mathcal{L}eaf(v_{pi})) \cap name(\mathcal{L}eaf(v_{pj}))$
$Tem \leftarrow$ MERGESUBQUERY (Na, p_i, p_j)
for $c \in Tem$
if c does not match any common sub-query tree in $\mathcal{T}$
add c to $\mathcal{T}$
$U_c \leftarrow$NULL
end if
if p_i is not in U_c
add p_i to U_c
end if
if p_j is not in U_c
add p_j to U_c
end if
end for
end for
end for
return $\mathcal{T}$
end *SharableSubQueryTrees(Q)*

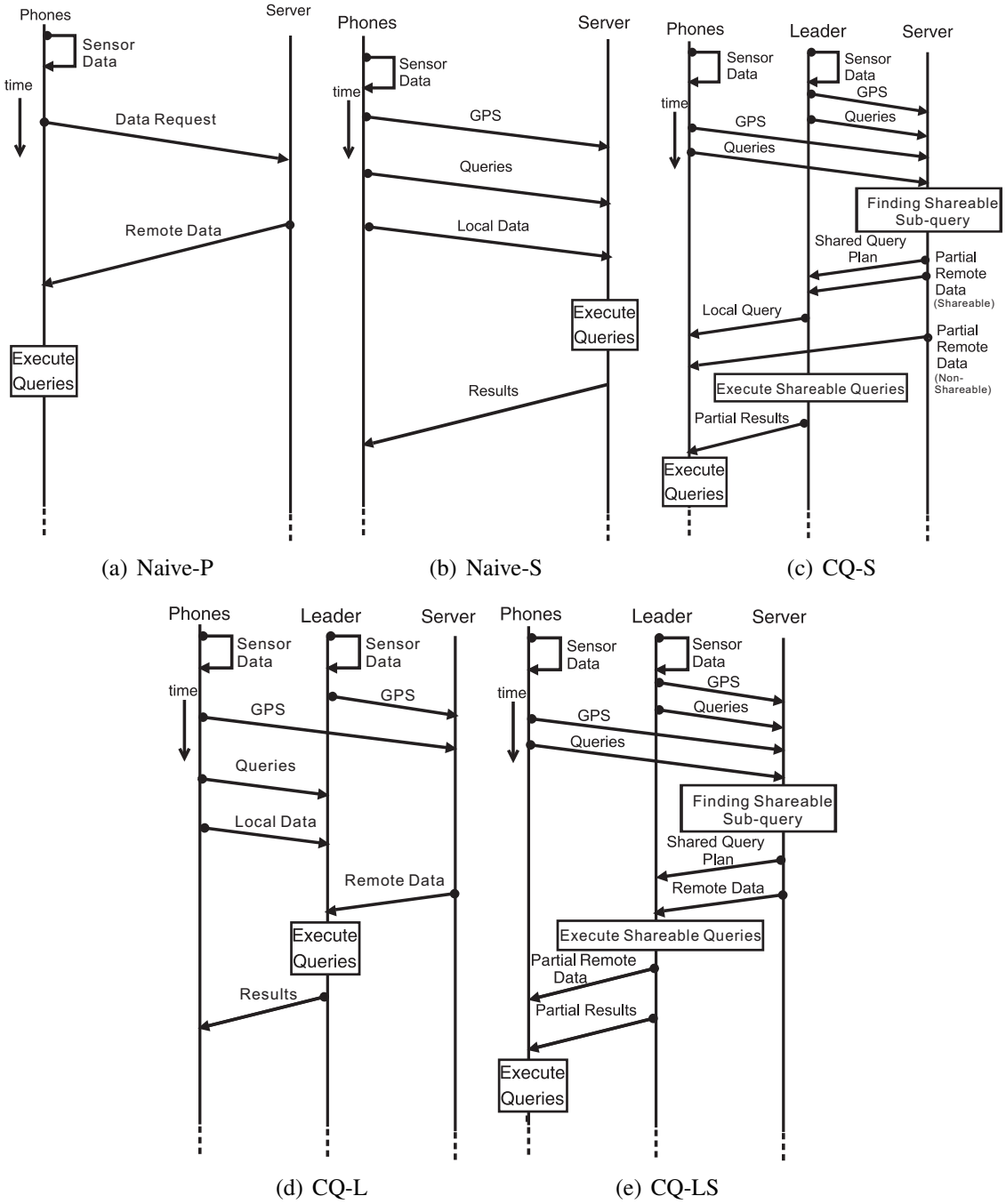
- **Naive-S** Fig. 4.11(b). At each query execution, each phone acquires the local sensor data (e.g. GPS) and sends the local sensor data and the query to the server. The server receives the sensor data and query from the phone, executes the queries using both local sensor data and remote data, and sends the results back to the phone.

Now consider query processing using the query fusion framework. We outline three possible strategies of collaborative query processing, CQ-S, CQ-L and CQ-LS, that differ mainly in the amount of processing handled by the group leader. In all three strategies, groups are formed and managed by the server as described previously. Using the query-based grouping information, the server then coordinates the collaborative query execution via collaborative query execution plans (CQEP). A query-based group must be contained within a location-based group. A phone-query pair is grouped into a query-based group if the query can share resources (such as local sensor data or common query fragments) with the other phone-query pairs in the query-based group. A collaborative query execution plan consists of group membership information, as well as query plan fragments for each device/system, including the group leader and the server. We outline how the collaborative query processing is performed for the three strategies within a query-based group of phones.

- **CQ-S** Fig. 4.11(c). Each member phone receives from the server a fragment of the CQEP for execution. The group leader also receives from the server a fragment of the CQEP called the *shared query plan*. The results of the shared query plan needs to be sent to the member phones in order for the member phones to complete execution of their CQEP fragment. In this strategy, each member phone acquires the remote data required by their CQEP fragment from the server independently. The leader phone and each member phone also acquires local sensor data from their own sensors. The group leader then executes the shared query plan and sends the results to the member phones, which then individually execute their residual CQEP fragments.

- **CQ-L** Fig. 4.11(d). The CQ-L strategy is exemplified by the group leader performing most of the query processing for the member phones. Each member phone sends its query and local sensor data to the leader. The leader acquires all the required remote data from the server. Upon receiving the required sensor data and remote data, the leader executes all the queries for each member phone and sends the results to each member phone.

- **CQ-LS** Fig. 4.11(e). The CQ-LS strategy differs from the CQ-S strategy by having the group leader act as a proxy for all member phones and acquire *all required* remote data from the server, instead of the CQ-S approach where each member phone acquires the remote data independently from the server. CQ-LS aims to exploit the low-powered Bluetooth link among group members for data dissemination. The group leader and each member phone also acquire local sensor data from their own sensors. The group leader then executes the shared query plan and sends the results and the required remote data to each individual member phone, which then individually executes its residual CQEP fragment.



(a) Naive-P (b) Naive-S (c) CQ-S

(d) CQ-L (e) CQ-LS

Figure 4.11: 2 Naive Methods and 3 Methods of Collaborative Query Processing

Chapter 5

Configuration of the Decentralized Data Fusion Framework

Chapter 3 and Chapter 4 describe the data exchange and data processing component in the framework. This chapter focuses on the configuration of directed fusion graph. Configuration is a general problem in distributed system, it determines how the fusion components find each other and how to get started, etc. In some applications, services can be configured on any devices available in the distributed system. However, for decentralized data fusion systems, the configuration decides how the data are routed to the fusion points, and then fused to complete the final fusion task. This further decides what data sets the fusion tasks execute on, and how they are executed. Thus the configuration of data fusion system is also application oriented.

The novelty of this work is that it involves configuration as one building dimension of the distributed fusion framework, and considers the configuration problem during the functional design process of the distributed fusion system. This is different from other research on configuration problems, which consider configuration as a separate post-design task upon implementation on distributed systems.

This chapter starts with section 5.1, given related work, set the foundation for comparison. It then summarizes the features and characteristics of the configuration and deployment mechanism. Section 5.2 discusses the process of mapping and building the Directed Fusion Graph into a networked system. This includes: the formal description of the directed fusion graph with marker language, YAML [93], and mapping this text description onto the system. Section 5.3 discusses the self-configuration and automatic adaption of the data fusion structure, considering nodes' joining in and leaving the system and mobility. The solution distinguishes the roles of leader node and participant node, and discusses how and when to update the YAML configuration file to achieve new data fusion topology under system dynamics.

5.1 Related Work and Characteristics of the Configuration Method of the Data Fusion Systems

Configuration and deployment of a decentralized data fusion system above heterogeneous networked devices capable of sensing and data fusion is not a trivial task. [46] summarizes several of the most general strategies to carry out this task:

- **Predetermined Strategy:** with some premier knowledge of the environment, the system administrator can decide the deployment of the sensor networks. The advantage of predetermined strategy is that it is possible to achieve the optimal solution given an aim to achieve, such as minimizing energy consumption, or messages used in the configuration phase, etc. [19] adopts a grid-based fixed interval to deploy the sensor nodes. The idea is to adjust the grid interval (the distance between adjunctive nodes) to achieve desired accuracy in sensing. [70] noted that predetermined strategy has the advantage of obtaining an optimal deployment solution for desirable coverage while achieving QoS and cost efficiency at the same time. However, the prior condition is hard to satisfy, which requires complete knowledge of the environment where the system is to be deployed.

- **Self-regulated Strategy:** [36] proposes an approach to deploy sensor nodes automatically in an unknown environment, where the aim is to reach the maximum coverage through repelling each sensor node by each other and obstacles. [12] presents a scheme to deploy sensor nodes sequentially in steps by introducing path exposure as target metric. The methodology is to properly choose the number of sensor in each step, so that the desired detection accuracy can be obtained with a relatively low cost in deployment. Comparing with the predetermined strategy, this self-regulated strategy is scalable and saves a lot of human efforts. This comes with a price through: the computational cost and communicational cost required to make decisions regarding deployment on the nodes themselves are extremely high, and the components or middle-ware to support the information collection and decision making processes are unavoidable.

- **Randomly Undermined Strategy** is more realistic for large-scale WSN application, where sensor nodes are generally spread uniformly in a given area [85]. The strategy has the advantage of low cost in deployment. The disadvantage is that the coverage of the resulting network is not uniform, and it does require a configuration phase to set up the initial routes of data towards the sink. [70], based on this work, proposed a practical virtual force algorithm to reposition the sensors in order to enlarge coverage to the desired aim, such as balancing the requirements in high- or low-detection accuracy with certain energy constraints.

- **Biased Distribution Strategy** deploys the nodes according to specified detection requirements and the environment. [92] illustrates an example of biased placement, where the density of sensor nodes nearby the windows is higher than inside the rooms. This strategy considers an application's requirements, thus is preferred in the situation where these prior requirements and conditions regarding the environment is available.

Considering the existence of heterogeneous nodes and mobile nodes in the scenario of this thesis, to completely adopt the predetermined strategy is impossible since the prior knowledge about the nodes, and their resource availability is unavailable, and the location of the some of the nodes is not fixed. Thus it is necessary to use a self-regulated strategy. The thesis tries to initialize the configuration and deployment with a predetermined strategy, to configure the network with several basic leaders according to the prior knowledge of the network. After this initialization, some selected leader nodes should take the responsibility for self-regulated configuration and deployment in the later phase in the organization of other nodes in facing dynamics of the environment and system itself.

The self-configuration and deployment of the directed fusion graph is a task, which requires the support from low layer self-configuration and deployment techniques, especially the work on self-configuration

at MAC layer or topology management. Thus this work has been informed and influenced by a variety of other research efforts.

Pottie and Kaiser [81] define techniques that wireless nodes used to discover their neighbors and acquire synchronism. This is the basic bootstrapping capability. ASCENT [10] based on this work, addresses the next level of automatic configuration that will be needed to realize envisioned sensor networks, especially on how to form the multi-hop topology [22]. The work is based on the ability to send and receive packets; they adopt the techniques from MAC layer protocols to solve the problem of distributed topology formation.

ASCENT consists of several phases. When a node first initializes, it enters a listening-only phase called neighbor discovery phase, where each node obtains an estimate of the number of neighbors actively transmitting messages based on local measurements. Upon completion of this phase, nodes enter the join decision phase, where they decide whether to join the multi-hop diffusion sensor network. During this phase, a node may temporarily join the network for a certain period of time to test whether it contributes to improved connectivity. If a node decides to join the network for a longer time, it enters an active phase and starts sending routing control and data messages. If a node decides not to join the network, it enters the adaptive phase, where it turns itself off for a period of time, or reduces its transmission range.

Another work, which is the basis of this work, is [82]. It focuses on low-level synchronization necessary for network self-assembly. The work solves the problem of self-configuration and synchronization in sensor networks at the single cluster level. The difference in between [82] and ASCENT is that the later aims to form the multi-hop topology efficiently, while the former is on lower level synchronization based on TDMA scheme.

[50] presented a scheme where mobile nodes modify their trajectory to transmit messages in the context of disconnected ad-hoc networks. This work and the thesis share the notion of adaptation of the basic topology for efficient delivery of messages. This work may complement ours in the case of mobile nodes deployment and in the presence of network partitions.

[28] presented an adaptive local network formation/routing algorithm that facilitates cooperative signal processing. An election algorithm is used to select a central node among a small group of nodes that cooperate in information processing. That work is an effort which integrates the application layer data processing aim together with the data routing protocols aiming at energy efficiency through the adaptive network formation. The problem definition and selection is similar to this thesis, which tries to support the application of signal processing with proper configuration and deployment strategies. While [28] were designed to support to one application in signal processing, the thesis aims to provide a general support for data fusion applications, in a dynamic environment.

The thesis is based on this prior work, focuses on the application layer configuration automation, to be specific, the configuration and deployment of the directed fusion graph onto the devices. The configuration is usually updated by the detection of the new neighbors, and triggered by low layer topology discovery algorithms. Effectively combining the configuration research work in the low layer, especially the bootstrap process and the topology updates under the mobile scenario, helps the thesis focus on the problem of configuration automation at the application layer. The middle-ware to support the automatic configuration of the fusion systems on each device has the following features:

- Firstly, the fusion middle-ware supports different fusion applications to combine different fusion

tasks **flexibly** to meet the requirements. Different data fusion applications are supported, by connecting different plug-ins of fusion function through generated ZeroMQ sockets. The different fusion functions and the interconnection modes of these fusion functions satisfy the specific requirements of applications.

- Secondly, this middle-ware supports **scalability** of the fusion system onto more devices. To scale the system such as adding fusion functions is of linear cost, according to the operation of adding text lines in the YAML files. It is a relatively lightweight supporting architecture; this provides the possibility to be installed on some resource restricted devices, such as mobile phones, etc.

- Thirdly, since the configuration YAML file is available on each node, which can contain one or more data processing (fusion) components. The interconnection of these components is specified by the YAML file. The connection gates of the data flows are generated dynamically with each node starting the configuration process by loading and compiling the YAML file. This self-configuration process can be triggered anytime when the configuration file changes or gets updated, by comparing the modification time of the available configuration plan and the local plan. This feature makes it easy for the system to dynamically change its interconnection topologies. It is also possible to be extended to support the **dynamic adaption** of the system structure according to certain optimized plans in special environments.

All the above features are beneficial to the self-configuration and self-deployment of a distributed fusion systems consists of wireless communication based, resource restrict devices, under dynamic environment.

5.2 Building Data Fusion Network based on Directed Fusion Graph

The directed Fusion Graph is a novel model of describing the actual logic structure of the fusion systems. It brings about the problem of how to transfer the system description, initialized as a graph, into a practical decentralized data fusion system. The graph description is a way only easy for system designers or administrators to understand, yet not directly available for devices to perceive for deployment. In order to achieve the "translation" aim, it is necessary to process the directed fusion graph through several steps until it can be processed by devices. This section describes this process. After the introduction to the entire process of such translation process in section 5.2.1, the following sections narrate each step of the process.

5.2.1 Mapping the Directed Fusion Graph on Distributed Networked System

Figure. 5.1 shows the entire process of translating a directed fusion graph to the decentralized data fusion system. The process starts from one piece or group of Directed Fusion Graphs coming from the designer of the system. A group of Directed Fusion Graphs represents a decentralized model, where several components are connected. The difference in between the single-node model and the distributed model is that there are interconnection edges (dotted edges) instead of inner edges (concrete edges) inside the graphs. The difference in between these two models is illustrated in Section 5.2.2.

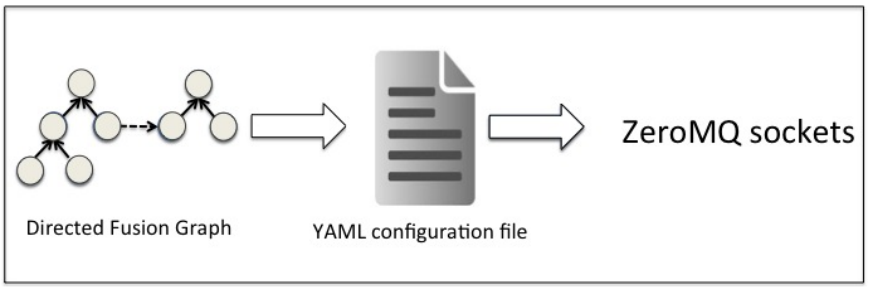


Figure 5.1: General Process of Translating a Directed Fusion Graph

Since the graph format cannot be directly processed by the devices, there is a second step to transfer the entire group of directed fusion graphs into an easy-to-read description in text format for both machines and human (YAML). In this step, the most important parts of the graph, both the nodes and the edges which interconnect the nodes are translated into text representation. The details regarding this step is described in Section 5.2.3.

In the third step, the YAML based text description of the fusion graph is further translated by the embedded library of the devices to generate the distributed fusion components and system. The focus of this step is on replacing the interconnection edges between two components with specific sockets of certain types, which can be used for data exchange to pass data from the output gates of one component to the input gates of another component. The nodes in the Directed Fusion Graph, at the same time are replaced with corresponding libraries of fusion algorithms. The translation of both the vertexes and the edges of the graph to the practical fusion functions and data exchange sockets completes the process, resulting in a practical decentralized data fusion system configured according to the Directed Fusion Graph. Section 5.2.4 focuses on the discussion of this step.

5.2.2 Centralized VS Decentralized Directed Fusion Graph Model

When the data processing system is described into the form of directed fusion graph, it is easy to deploy such graph onto a single node. As programs generally take in data, process it, and then feed it back out. It is well represented in the UNIX operating system, which pipes the data between small single-purpose tools. Programs in a data flow language start with an input, perhaps the command line parameters, and illustrate how that data is used and modified. The data is now explicit, often illustrated physically on the screen as a line or pipe showing where the information flows. Thus to deploy and configure fusion functions on one single node is easy to implement through function calls without worry about the intermediate results.

This section considers a decentralized model to deploy such directed fusion graph onto several nodes instead of only one. The interconnections among fusion functions can be within one node or among several nodes. The interconnection of fusion functions within one node can be achieved by function calls and cached in local memory. The flexible configuration and deployment of a decentralized model

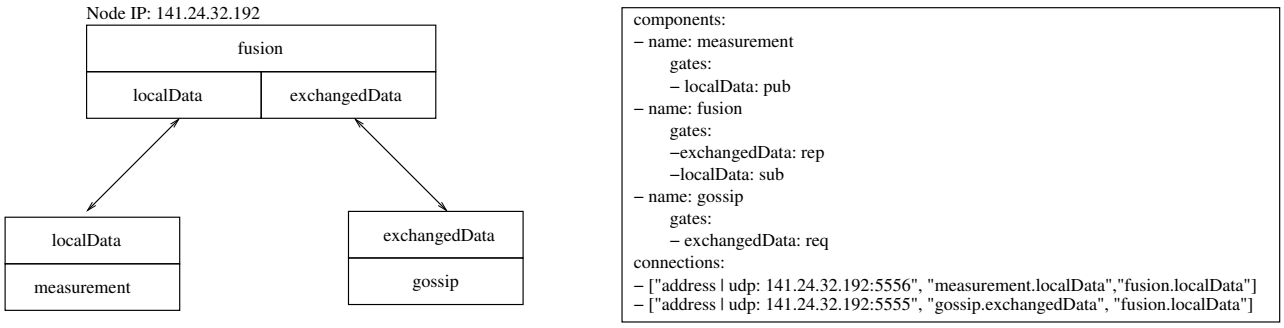


Figure 5.2: In Network Configuration Fusion Functions

is the problem that this thesis mainly tackles. Very little previous work focuses on providing solutions with flexibility and resource constraint, when considering this configuration problem.

5.2.3 Text Specification for Directed Fusion Graphs

Once we have the directed fusion graph describe the logic architecture of the system, the next step is to transform this graph into a form which can be easily understood by machines and devices, as well as human (system administrators). This section use YAML to achieve this goal. This format is then interpreted by the machines to generate actual architectures to deploy on the devices.

To easily process the directed fusion graph, it can be rewritten into a text description as a deployment plan. This plan should be easy to understand by people, and be processed by the nodes. This chapter uses YAML to record the deployment plan for the system. YAML is a human-readable data serialization format that takes concepts from scripting languages, and the ideas from XML, together with the data format of electronic mail (RFC 2822). It is available for several programming languages. YAML is to distinguish its purpose as data-oriented, rather than document markup.

YAML lacks the notion of tag attributes that are found in XML. For data structure serialization, tag attributes are, arguably, a feature of questionable utility since the separation of data and meta-data adds complexity when represented by the natural data structures (associative arrays, lists) in common languages [93]. Instead, YAML has extensible type declarations (including class types for objects). YAML itself does not have XML language-defined document schema descriptors that allow, for example, a document to self validate. The reason for YAML to be adopted is that the supporting libraries and interpreter are relatively light-weight for devices to carry.

Figure. 5.2 is an example of translating fusion graph into YAML description. The description separates the fusion components and the connections. The format is straightforward, easy to compare with the graph. The inter-node configuration specifies, at which socket and on which IP address, the sockets for data exchange among two components should be set up. These data sockets are called from different fusion components to exchange data according the specified modes. It is not important to specify in this YAML description where the fusion components are deployed on which devices. These components can use the specified address of the ZMQ sockets from the YAML specification to form the actual connection edges.

Table 5.1: Classification of Tasks in the Directed Fusion Graph

Tasks or functions	Definition	Requirement on resources		
		Bandwidth	Computation	Energy
Sensing and Measurement	Generate raw data sets	low	low	low
Data aggregation and filtering	Initial processing to improve data quality	low	average	average
Feature or Decision Fusion	Fusing backbone to generate average or final results	high	high	high
DFG active management	Monitoring topology changes updating DFG architecture	high	average	high
DFG passive updating	Receive new DFG recompile to generate new architecture	low	average	average

5.2.4 Dynamical Deployment of the YAML Description

The YAML based fusion system specification is another form of the directed fusion graph, another blueprint of the structure of the fusion system. Once the devices in the distributed networked system can compile to understand the structure, they can generate relative ZeroMQ data connection sockets to implement the system architecture. This is an automatic configuration method for the distributed fusion system. Comparing with the traditional manual configuration method, this method provides a flexible self-configuration possibility. In order to achieve this goal, a certain unified architecture is necessary to be installed on each device to provide interfaces to accept the configuration plan and to translate the plan into actual data exchange sockets at application layer.

5.3 Self-configuration and Deployment of the Data Fusion Network under Dynamics

After mapping the directed fusion graph onto devices, another problem is the deployment of these nodes carrying the data fusion graph into the field. In other words, which node should carry which part of the fusion functions? It's worth further consideration. This is related with the features of the fusion tasks and the capacity of heterogeneous nodes in the network system. Several distinct tasks in the Directed Fusion Graph need to be accomplished, these tasks and their requirements on the resources of the devices are illustrated in Table 5.1:

In the table 5.1, the first three tasks are the classical fusion related tasks, including raw data generation, data aggregation and filtering applications, and complex data fusion techniques. The other two tasks are related with dynamic topology management, which enables the dynamic updating of directed fusion graph (DFG) according to the changes coming from the systems and the environment:

- **DFG active management** is the task which collects the information coming from the low layer, regarding changes on the neighbors. The changes include neighbors joining, leaving, moving to an-

other location, etc. The changes are detected by the techniques discussed in Pottie and Kaiser[20]. After detecting these changes, the tasks that should be executed are updating the current text description of the DFG, by adding or deleting certain nodes and their related connections in the YAML file. This is through a rule engine defined according to these changes, and then actions are taken. Thus changes are directly written to the YAML file. The next step is to flooding the new YAML file representing new DFG architecture to its current neighbors.

- **DFG passive updating** is the task which passively listens to figure out if there is new version of the DFG sent by valid neighbors. After receiving such new version of the DFG, the task should run an auto compilation to generate new ZeroMQ sockets based on the new architecture.

The differentiation of these fusion related tasks, and the requirements on the capacity of the devices show that it is necessary to identify at least two roles for nodes, based on the capacity of the nodes:

- **Leader nodes** should be capable of the tasks which have high requirements on the availability of resources. They are capable of executing the main tasks of data fusion, routing on the backbone of the fusion graph, and are in charge of DFG active management, etc. These nodes should be authorized to update the existing YAML description of the DFG and to distribute it over nearby nodes.

- **Participant nodes** are capable of the remaining tasks to reduce the resource consumption of the entire data fusion system. They should listen for the update of the current topology in the DFG and dynamically compile and execute these changes.

The roles of the nodes can be either fixed or rotated in turn according to the resource availability of these nodes, the selection of leaders can adopt existing self-organization algorithms such as LEACH [34] [49]. Although the stability of the leader nodes would benefit the entire data fusion architecture, it is dependent on the type of the network and the heterogeneous nodes in the networked system. For instance, in mobile communication networks, it is possible to initialize base station as leader nodes, while in wireless sensor networks, the leader nodes can be initialized according to locations, and further updates are expected based on energy availability.

The deployment of the DFG, and the dynamic update of the data fusion topology and the corresponding DFG under system or environment dynamics follows the two diagrams distinctly, which are illustrated in figure 5.3 and 5.4.

Figure 5.3 describes the leader process. There are several phases or steps for the entire configuration process:

- **Phase1: initialization phase** The leader nodes are initially given a YAML file representing the architecture of the data fusion system, with consideration of only backbone nodes. The initial YAML file is designed and generated by the system administrators, the purpose of this file is to assign several leader nodes based on their location, this can be generated by any clustering algorithm based on two-dimension distance. The initial backbone nodes play the leader roles until the next round of leader selection and role switch according to LEACH is triggered. The initial fusion components and the interconnection of these components are defined in the YAML file.

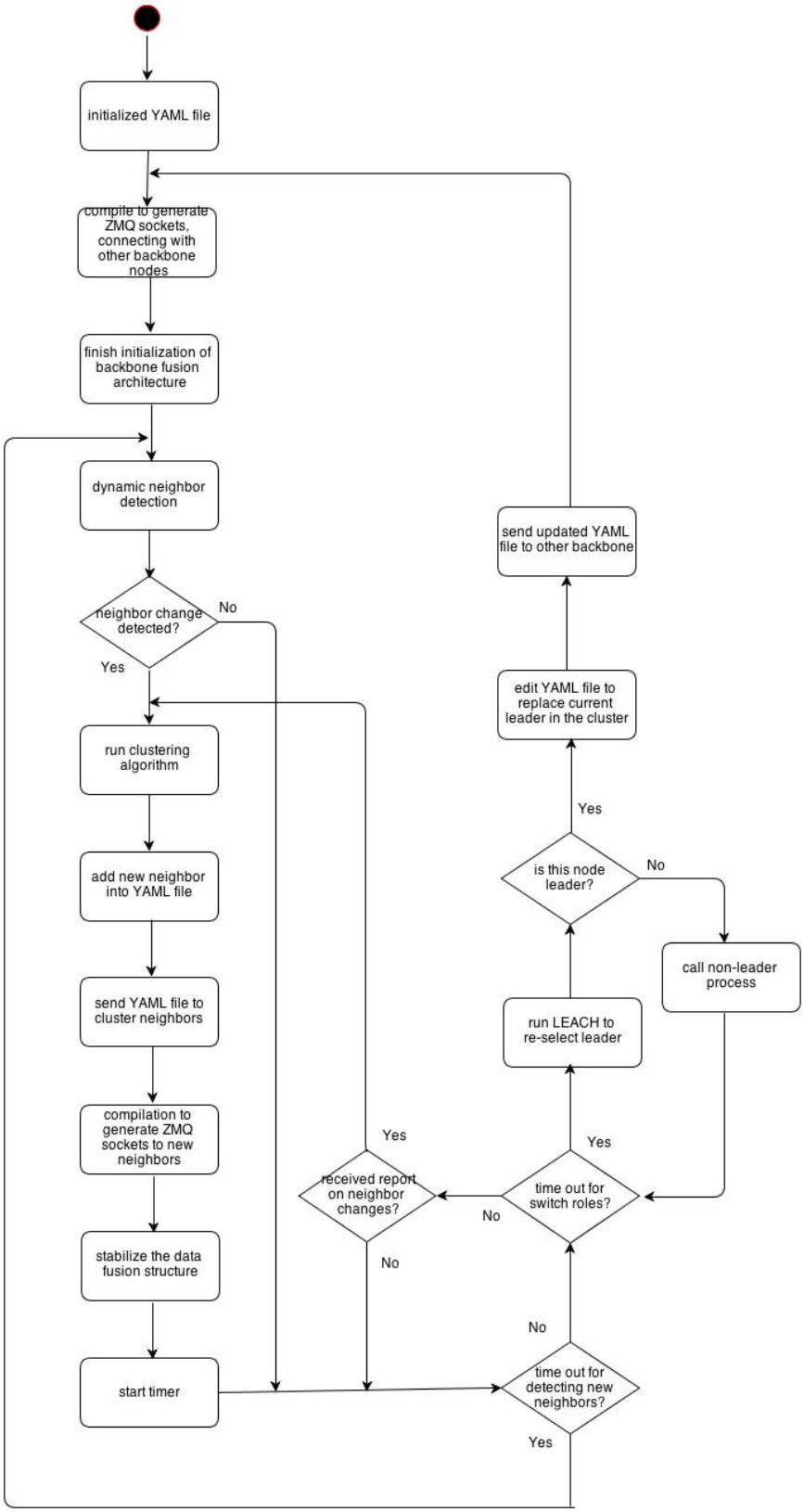


Figure 5.3: Leader Process

The initial YAML file is compiled to generate ZMQ sockets to connect with the other backbone nodes, which during the same phase call the fusion functions in the libraries to generate the initial distributed data fusion structure.

- **Phase 2: neighbor addition** The initial YAML file is a general blue print which does not include all the nodes and devices of the network system. The other nodes in the area are discovered by the local leader nodes through the neighbor discovery algorithm [81] and added into the YAML file. The cluster leader is capable of updating the YAML file with write access. After updating the YAML file, the cluster leader would propagate the new YAML file to its local area, to let the participant nodes learn the current configuration. The participant nodes, after receiving that new YAML file, can compile and generate corresponding ZMQ sockets to connect with each other. At the end of this process, the directed fusion graph is deployed and stabilized with current nodes and devices. This step is also necessary for nodes moving from one cluster to another cluster to be triggered.

- **Phase 3: roles switch** When the LEACH algorithm reports the selection of the new leaders out, if the current leader does not change to be a participant node, it continues its current role in the data fusion system. If it has been decided to be a participant node, it is switched to execute the participant, non-leader process.

Besides the initialization phase, the neighbor addition and the role switch process loops until the node exits from the networked system. The intervals for role switching and new neighbor detection have an effect on the energy consumption of the battery. This is a necessary cost for the system configuration to deal with the mobility and system dynamics.

The non-leader process is described in figure 5.4. There are several phases or steps for the entire configuration process, comparing with the leader nodes; the non-leader node does not need to consider updating the configuration file, but only needs to report the of new neighbors to the cluster leader. As participant node, it requires only passive detection of updates of the configuration file. Whenever it received a new version, it needs to close the current ZMQ sockets and compile to generate new sockets. After the node has been switched to be leader node, it starts to execute the process in figure 5.3.

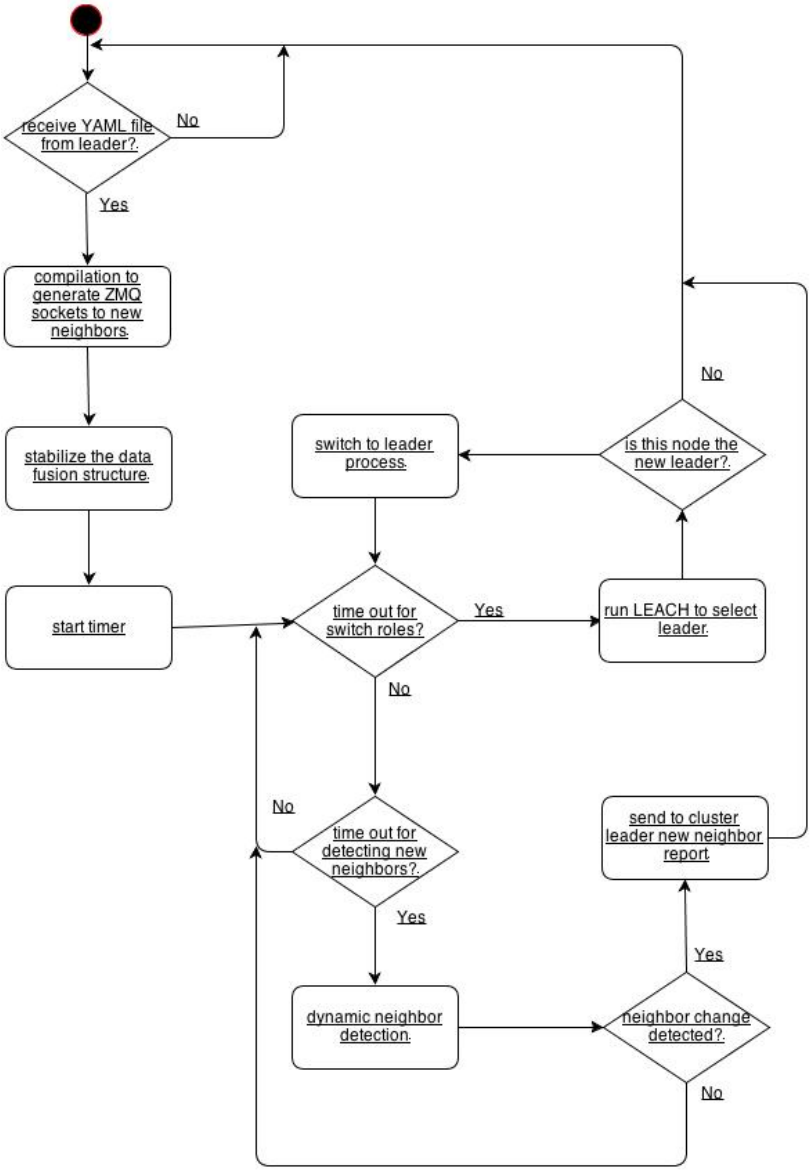


Figure 5.4: Non-leader Process

Chapter 6

Evaluation and Implementation of the Decentralized Data Fusion and Data Harvesting Framework

This chapter evaluates the performance of the data fusion and data harvesting framework from different perspectives. Firstly, in Section 6.1, the robust information exchange protocols used in the data fusion and data harvesting framework is examined for different failure scenarios. Different types of information exchange protocols are compared under various failure models to reach a general conclusion to support the engineering design. Following this section, a framework targeted to serve decentralized change detection is evaluated, and some results concerning the detection accuracy are provided. Since data accuracy after fusion is closely related to specific applications, it is better to evaluate the performance of data accuracy under specific application scenarios. The last section discusses another application use case of the framework which works on query fusions. Evaluation results on this special query fusion framework are provided in details in the last section.

6.1 Simulation Evaluation on Information Exchange Protocols in Failure Scenarios

This section focuses on the evaluation of the data exchange protocols of a framework for network systems consisting of large number of nodes. The mechanism used in this part is simulation, this helps examining the performance of the information exchange protocols under stress. The evaluation of information exchange protocols cannot be left alone without the information fusion functions, since the aim of information exchange is to facilitate data fusion. The simulation aims to explore the interferences of the failure scenarios, in terms of information dissemination speed, accuracy of the fused data and number of messages transmitted.

In order to focus on the interference of the failure scenarios to wireless communication based distributed system consisting of large number of nodes, the simulation makes several hypotheses:

- All the nodes of the data fusion system are of the same type and configuration. Although this assumption is not conformed to heterogeneity, this is reasonable in the case of wireless sensor

networks. The same type of nodes is deployed in one area for the purpose of monitoring. This assumption leaves the difference from the low level hardware or software apart from the application level data fusion and exchange tasks. Thus, the performance of these tasks plays a major role in the reaction to the different failure scenarios.

- In the simulation for location related failures, it assumes that the energy for each node is enough, this assumption weeds out the interference of topology changes caused by nodes out of battery. For example, in the location related failure models, such assumption helps exploring the interference caused by the different locations of failures. Thus, the simulation assumes that besides the failure nodes, all the other nodes have an unlimited energy supply until the end of the simulation.

- The failure models adopted in the simulation are the crash-failure model, which means that a node halts but is working properly until it halts. The simulation does not take the more complex models such as omission failure, timing failure, response failure or Byzantine failure into consideration. This simplifies the simulation to focus on essential phenomena that the failure of nodes brings about. It filters out some random phenomena which might not be caused by the data exchange protocols and fusion structures. Thus it is easy to compare the performance in different scenarios.

- Last but not least, the comparison is mainly on the application layer data fusion and data exchange protocols. Thus the common low layer wireless communication protocol stack is necessary to set up the foundation for the comparison at application layer. The simulation picks the 802.11 MAC protocol and UDP. Although there are quite a lot of other options and even cross layer designed low layer protocol stack especially for wireless sensor networks, the adoption of these two most common protocols is widely used and easily adapted to other nodes such as mobile or station devices.

Based on the above assumptions, the following space describes the evaluation for the gossip protocols in wireless communication based distributed fusion systems. The evaluations metrics are firstly introduced, then the description off the simulation environment and the specific evaluation results are presented. The detailed simulation results are discussed in the following section. The essential aim of the experiment is to provide a comprehensive engineering suggestion on the adoption of proper types of protocols to exchange data under different failure situations. This is shown in the last part of this section for evaluating the data exchange protocols.

6.1.1 Evaluation Metrics

The peer-to-peer mode itself of these distributed fusion systems might be interfered by the system or environment dynamics. Different failures might be caused by the disaster or emergency situations: node failure, unreliable wireless communication links, incorrect value readings, and mistaken warning report, etc. Threaten by all these different failures; the performance of the information dissemination is evaluated towards the following metrics:

- Accuracy (Correctness): The system itself is not stable, how to guarantee the information is correctly sent out and disseminated within the networked system is a problem. The usual notion of correctness can be defined as the value of the computed fusion results being exactly equal to the

true value. Laukik [11] proposes a realistic notion of correctness under faults for aggregation protocols in sensor networks: robust correctness. If only one of the nodes becomes faulty and the aggregate is the sum of all the values (one per node) then, under this notion of correctness, the result is allowed to be between the value of the sum with and without the faulty node. We use the relative difference in between the fusion result and the actual value to define the accuracy in the following simulations.

- Robustness: The robustness that defines the capacity of a system to operate correctly and to ensure accuracy despite external factors such as node or link failures. This section evaluates the performance of the data exchange protocols under different failure scenarios, and compares it with the performance under normal scenarios. The aim is to reach the conclusion of which protocol is more robust and why it should be adopted in certain failure models.

- Energy efficiency: the energy consumed in the distributed fusion system consists of the computation and the communication costs. **Computation cost:** the maximum computation cost among all the nodes in the network, for a single node, the computation cost is the number of steps taken by the process that is executed on the node. **Communication cost:** the sum of sizes of messages sent between any node pairs during fusion process takes place. Generally speaking, the wireless communication cost is usually several orders of magnitude of the computation cost. The design of energy efficient protocols is to reduce the volume of data transmitted, at the expense increase calculations. Thus data aggregation, data fusion and coding, etc are regarded as effective method in reducing energy consumption.

- Information dissemination speed: it is the necessary time between the initialization of the fusion and the time when all nodes (or querying nodes) hold the fusion results (for instance, the elapsed time for both communication and computation).

Besides all the above metrics which are measurable during the simulation evaluation, there is one more requirement on flexibility. The fusion functions and the data types transmitted are different. The information dissemination protocols should meet these special requirements from the different applications and adapt to the environment dynamics.

The aim of the evaluation is to compare the performance difference for gossip protocols, refined gossip protocols, and even flooding in terms of these metrics, under different failure scenarios. The comparison results can be used to make proper design decisions for wireless communication based decentralized fusion system.

6.1.2 Simulation Setup

The simulation tool used is OmNet++ 4.0, which provides a library for the 802.11 MAC layer included in the INET framework. We use the 802.11 MAC and UDP protocols to construct the underlay. This setting is beneficial for the performance evaluation focused on the data dissemination protocols on the application layer.

In the initialization step, the distributed fusion system is in a matrix layout with 900 nodes, the internode distance is 150 meters. The sensing range of each node is 250 meters, which means the nodes

Table 6.1: Simulation Parameters Setting Up

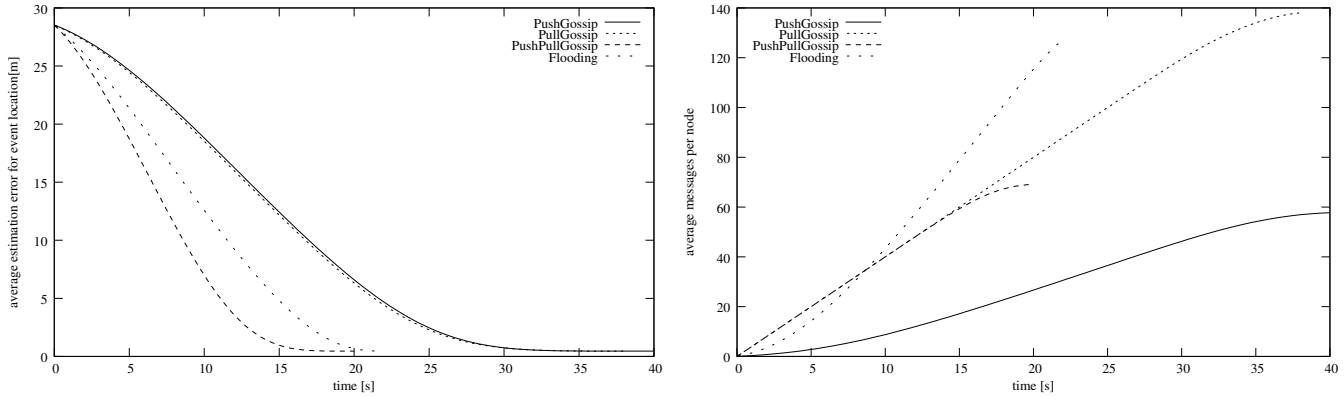
Parameter name	Parameter value
mac.maxQueueSize	14
mac.bitrate	2Mbps
radio.bitrate	2Mbps
radio.sensitivity	-75mW
radio.snirThreshold	4dB
channelControl.playgroundSizeX	800m
channelControl.playgroundSizeY	800m
channelcontrol.carrierFrequency	2.4GHz
channelcontrol.pMax	1mW
channelcontrol.alpha	2
channelcontrol.numChannels	1
sensor.DataFusionType	MaxQualityPosition
sensor.MeasurementType	PositionMeasurement

in the central area have 8 neighbors, nodes at the edge or at the corner have 5 or 3 peer neighbors. There are three information sources (event locations) located among the sensor nodes - their positions are randomly initialized, and they are within the sensing range of some sensors nearby. The gossip fan-out(branch number) is set to be from 2 to 8. In case of fan-out being equal to 2, 2 neighbors are selected to carry out information exchange in the iteration.

For the details of the simulation scenario, MAC and physical layer parameters are set up according to Table 6.1. Since gossip protocols or flooding protocol run on the application layer, the low layer parameters should be set up identically to ease the comparison of different protocols. This guarantees the same underlay communication infrastructure, and provides a fair platform to compare only the performance of protocols which run on the application layer. At the transport layer, UDP is used. We assume that initially each node obtains only a partial view, which is reachable with the radio broadcast range. Through further data exchange, the information on further reachable nodes is also piggybacked with the data message. So the knowledge about more neighbors is enlarged as the time goes by and the process of information exchange carries out.

6.1.3 Simulation Analysis

In the following space, the performance of the gossip protocols is analyzed under the scenarios introduced above. We start by demonstrating the advantages of gossip protocols to flooding protocol; this shows the general performance improvement for wireless communication based information exchange. The thesis then moves to discuss the influence of different failure models to the protocols, with the focus on data accuracy, the number of messages consumed, and the convergence time under these failure scenarios. This is supposed to give reader a general impression on the influence of disasters or failures to the system. Following this discussion regarding failure scenarios, the thesis further analyzes the performance gain of using refined gossip protocols comparing with normal gossip protocols under failure scenarios. Lastly, the thesis summarizes by providing a table to compare different gossip protocols, and suggests the proper protocols for different application scenarios. Each subsection describes the detailed



(a) Comparision of Data Accuracy (b) Comparison of Message Number

Figure 6.1: Gossip Protocols versus Flooding in Normal Scenarios without Failures

evaluation results from the simulations.

6.1.4 Improvement of Gossip Protocols Comparing with Flooding

Figure 6.1 shows the comparison results of gossip protocols with flooding under normal scenario. Figure 6.1(a) shows the data accuracy of the different dissemination protocols. Since this evaluation metric is always related to certain fusion functions, we use a general location estimation algorithm to be the fusion task. The algorithm estimates the locations of the three information sources and compares these locations with the actual locations to find out the difference. The error estimation is averaged for the three errors as stated in the vertical coordinate. The horizontal coordinate states the simulation time.

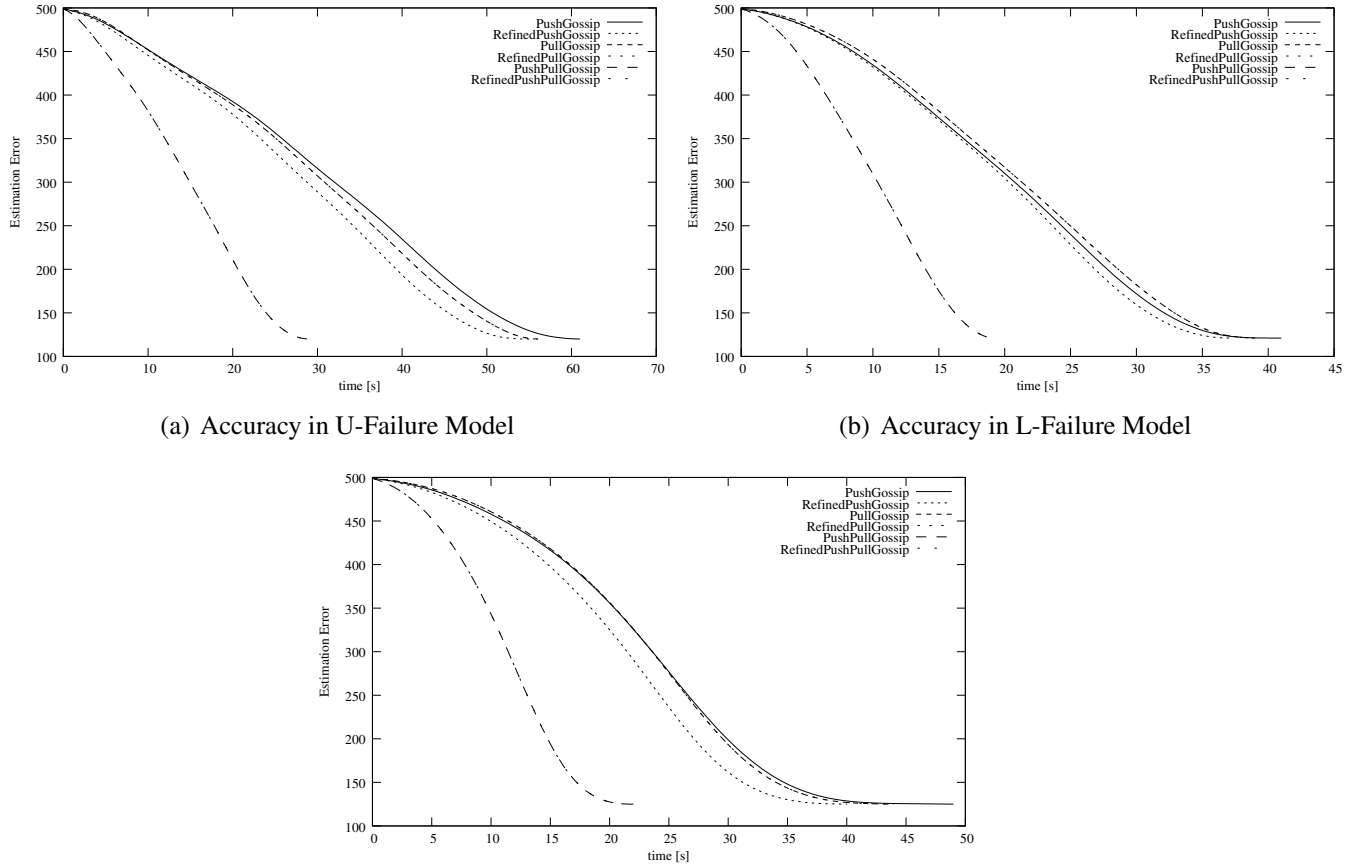
The location estimation fusion function is a general application to show the trend of convergence speed and relative data accuracy. In the left figure 6.1(a), the PushPullGossip protocol (fanout=2) converges faster towards the minimum estimation error than the broadcast (flooding) protocol. Yet PushPullGossip consumes less data messages comparing with flooding. This comparison demonstrates the advantage of PushPullGossip comparing with flooding. The result is the general foundation of using gossip protocols instead of flooding in the wireless communication based distributed fusion systems.

6.1.5 Accuracy of Gossip Protocols under Different Failure Models

Figure 6.2 shows the accuracy of different gossip protocols under spatial failure models, and figure 6.3 shows the accuracy of gossip protocols under the fault models of unreliable wireless links (with different message drop rates).

In figure 6.2, because some nodes fail, it is difficult to reach the zero estimation error. Since there are three event locations, it is for sure that some nodes near the event location might fail to collect the sensing data. The nodes failures affect the accuracy of estimation. Thus in all three models, the final estimation errors are not zero. Only the time to reach this minimum error is different.

The U-failure model blocks the main information dissemination route, which is the diagonal of the square. Instead, the information should be disseminated along the three edges. This causes the convergence time of U-failure much longer than that in the L-failure model. In the L-failure model, a quarter of



(a) Accuracy in U-Failure Model

(b) Accuracy in L-Failure Model

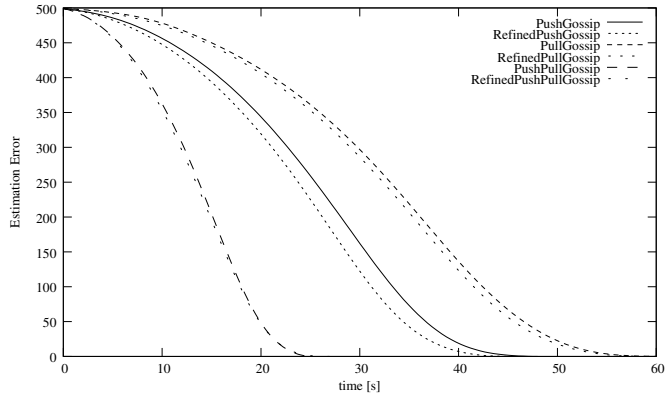
(c) Accuracy in Random Failure Model

Figure 6.2: Accuracy of Different Gossip Protocols in Spatial Failure Models

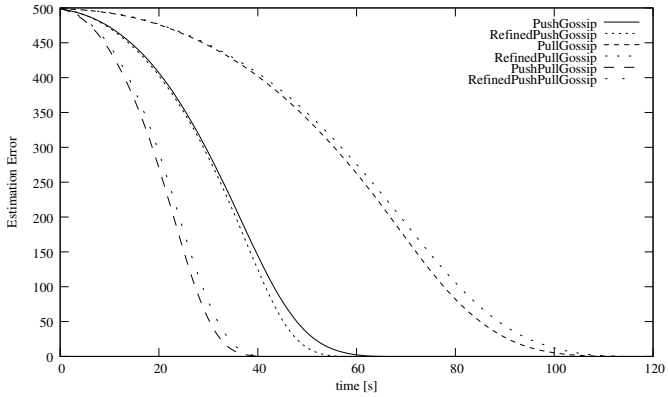
the nodes on the upper right corner failed. Yet it does not extend the path of information dissemination. The random failure model does not belong to these two extremes, thus is the second in the dissemination time.

In figure 6.3, there is a clear trend for the estimation error to converge to be zero with the time passing by. This is because there is no damage to the information source, only some packets get lost during the communication. Given enough time, all the three models can at the end reach the zero estimation error. Among all the results, the “push-pull gossip” is the fastest, “push gossip” follows and the “pull gossip” is the slowest. This trend is not clear in the spatial failure models. “push-pull gossip”with and without the target selection algorithm are overlapping with each other as the fastest ways to disseminate the location estimation information. From figure 6.3, the failure models of message loss caused by wireless link problem do not lose accuracy, given enough time to disseminate information inside the system. While the spatial failure models cased by node failures might sacrifice the final accuracy, due to the reason that some nodes near the information sources may fail and thus not able to provide their measurements to improve the location estimation. In the case where 75% messages are dropped, it takes a much longer time for the estimation error to reach minimum.

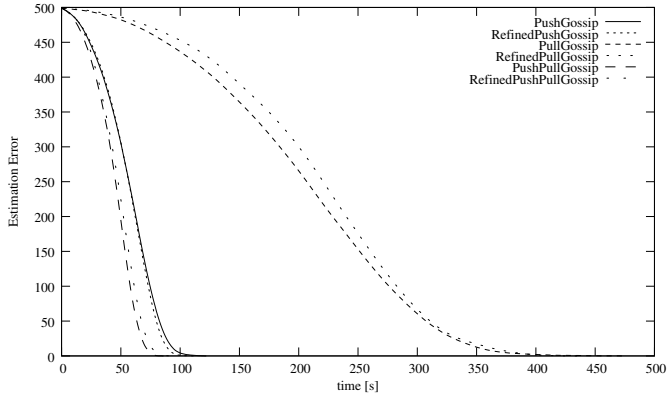
In summary, the message loss caused by link failure is especially difficult to deal with using original gossip protocols, which requires much longer time to disseminate and reach the minimum error (or



(a) Accuracy in 25% Failure Model



(b) Accuracy in 50% Failure Model



(c) Accuracy in 75% Failure Model

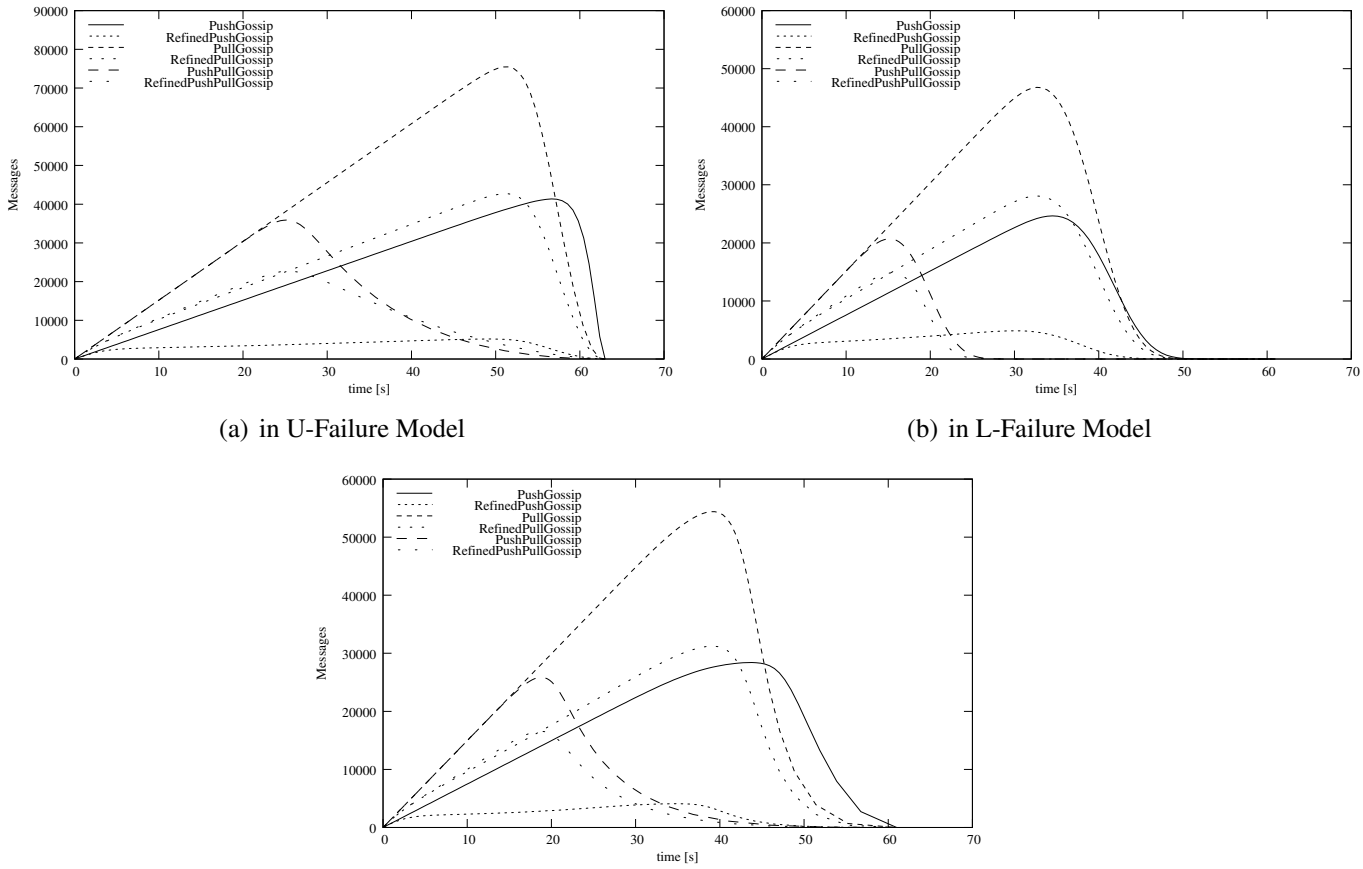
Figure 6.3: Accuracy of Different Gossip Protocols in Failure Models of Wireless Links

maximum data accuracy). Using refined gossip protocols with history records, the dissemination of data messages is more effective. Because it reduces redundant message transmission and avoids further damaging to the lower layer communication networks; while speeding up the information dissemination among the whole system. Among the different refined gossip protocols, "refined push-pull gossip" is superior at reaching the most accurate aggregation results in shortest time.

6.1.6 Communication Consumption in Terms of Message Number in Different Failure Models

Figure 6.4 shows the number of messages consumed in the dissemination process among different gossip protocols under spatial failure models. In these three failure models, the "refined push gossip" is the most message saving protocol. It spends only 10% to 20% of the messages of the normal "push gossip", which consumes the second less messages. The other two refined gossip protocols also over-perform corresponding original gossip protocols. This clearly demonstrates the superiority of refined schemes over the original schemes.

The same result appears even more obvious in the failure models of wireless links. Figure 6.5 is the number of messages consumed in the information dissemination process under the failure models of

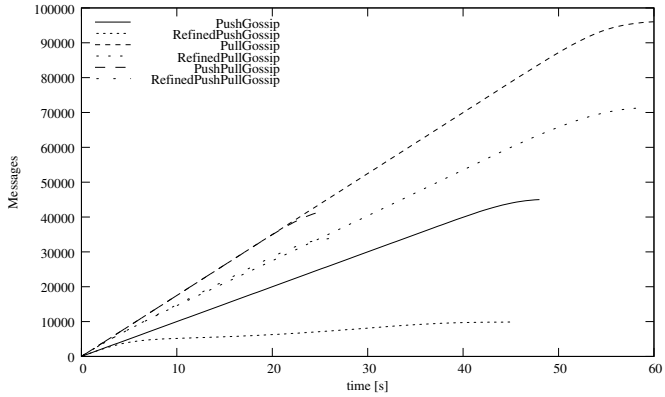


(a) in U-Failure Model

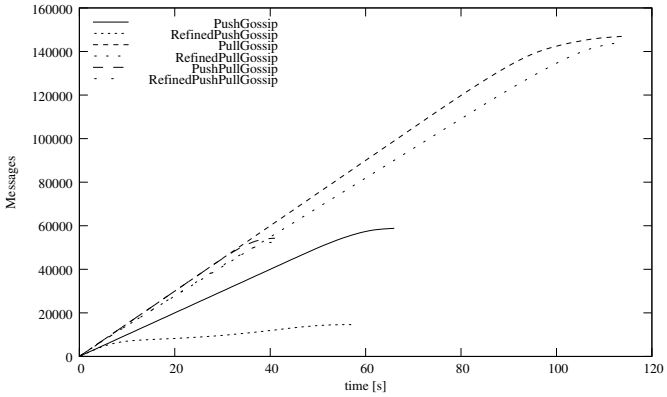
(b) in L-Failure Model

(c) in Random Failure Model

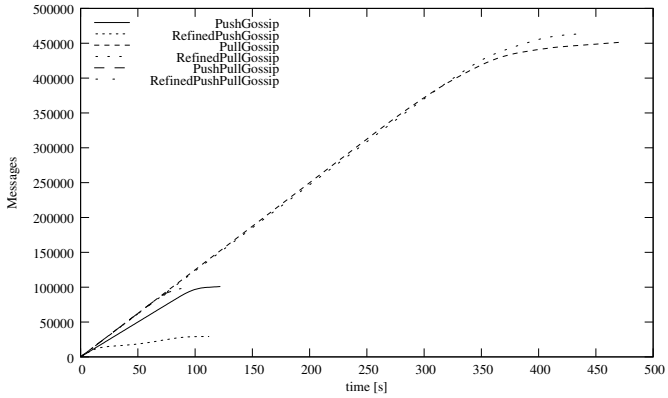
Figure 6.4: Number of Messages for Different Gossip Protocols in Spatial Failure Models



(a) in 25% Failure Model



(b) in 50% Failure Model



(c) in 75% Failure Model

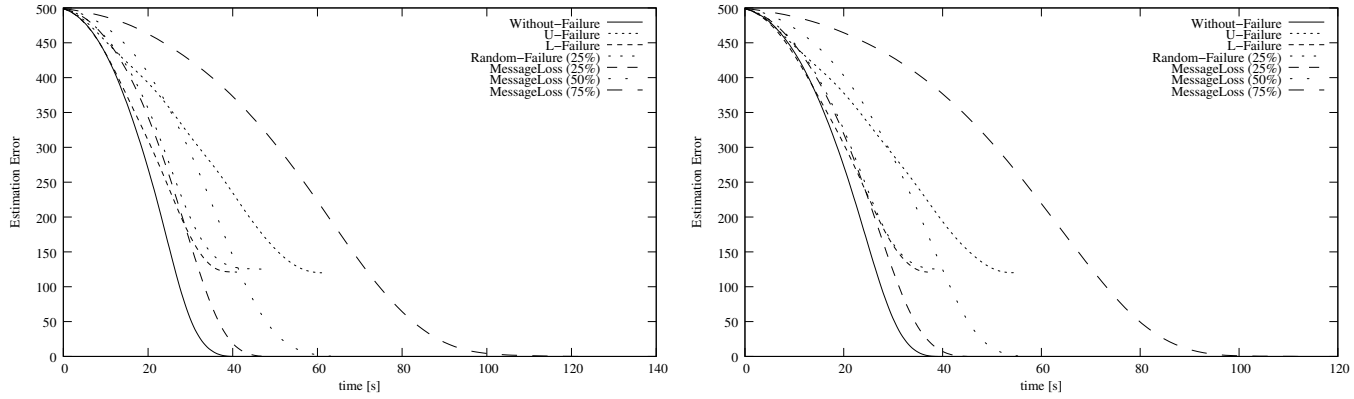
Figure 6.5: Number of Messages for Different Gossip Protocols in Failure Models of Wireless Links

unreliable wireless links. The "refined push gossip" is still the most efficient in messages consumed. In these three failure models, the "refined push gossip" spends only 15% to 30%of the messages of the normal "push gossip". The unreliable wireless links cost more message transmissions than the spatial failure models. This is because messages dropped need to be retransmitted. Under all the failure models, the "refined push gossip" is especially efficient in reducing communication message number. It should be adopted under this failure scenario.

6.1.7 Advantage of Refined Gossip Protocols in terms of Convergence Effect

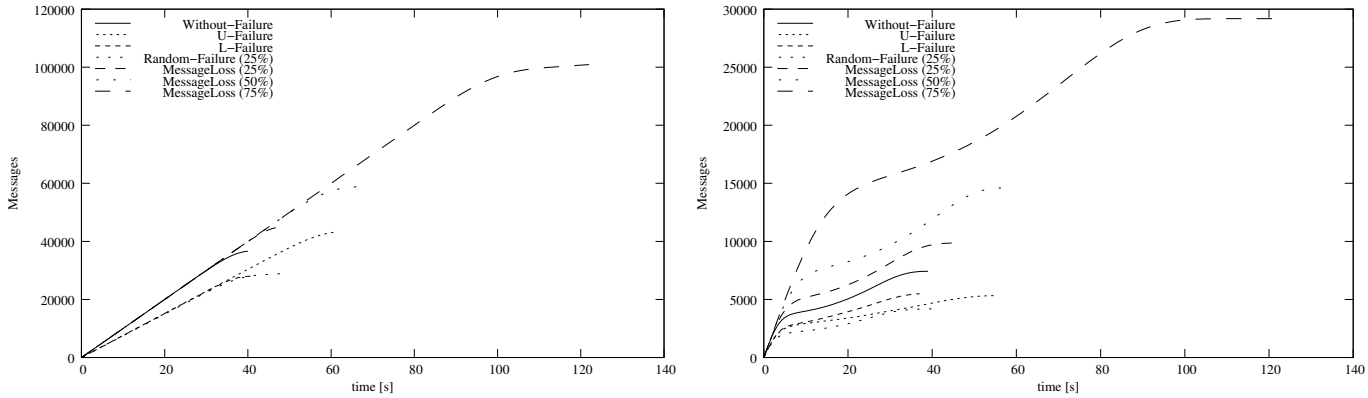
For the three special types of gossip protocols: "push gossip", "pull gossip" and "push-pull gossip", there are three corresponding refined gossip protocols. This section analyzes the difference in terms of Convergence Effect between gossip protocols and the refined gossip protocols. Their performance under either normal scenarios without any failure or various failure models is illustrated in the following:

- **Push Gossip versus Refined Push Gossip Protocols (figure 6.6 and figure 6.7)**
  figure 6.6 shows the processes on the peers of the system using the two different protocols ("push gossip", and "refined push gossip") to disseminate and fuse the location estimation results under



(a) Push Gossip (b) Refined Push Gossip

Figure 6.6: Convergent Accuracy of Push Gossip vs. Refined Push Gossip

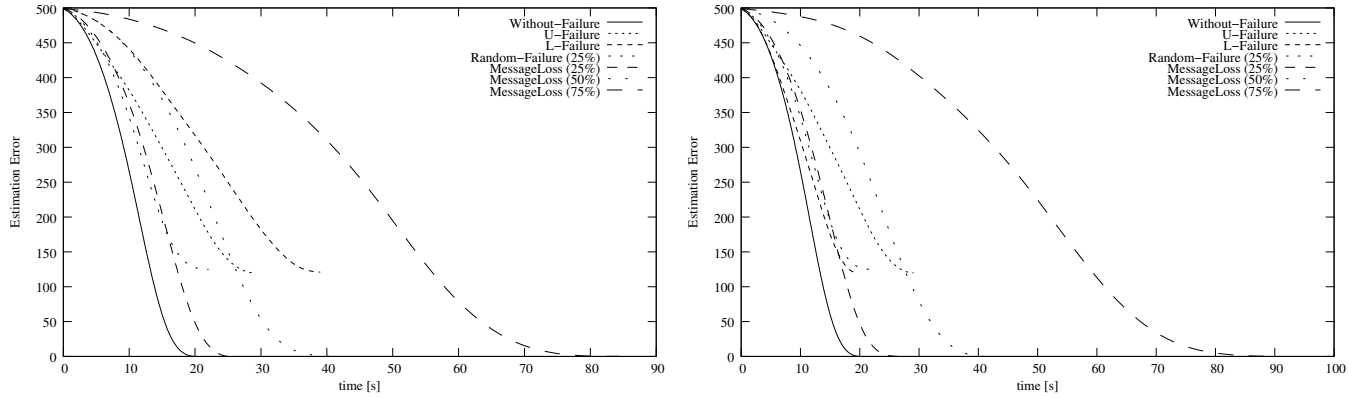


(a) Push Gossip (b) Refined Push Gossip

Figure 6.7: Convergent Message Number of Push Gossip vs. Refined Push Gossip

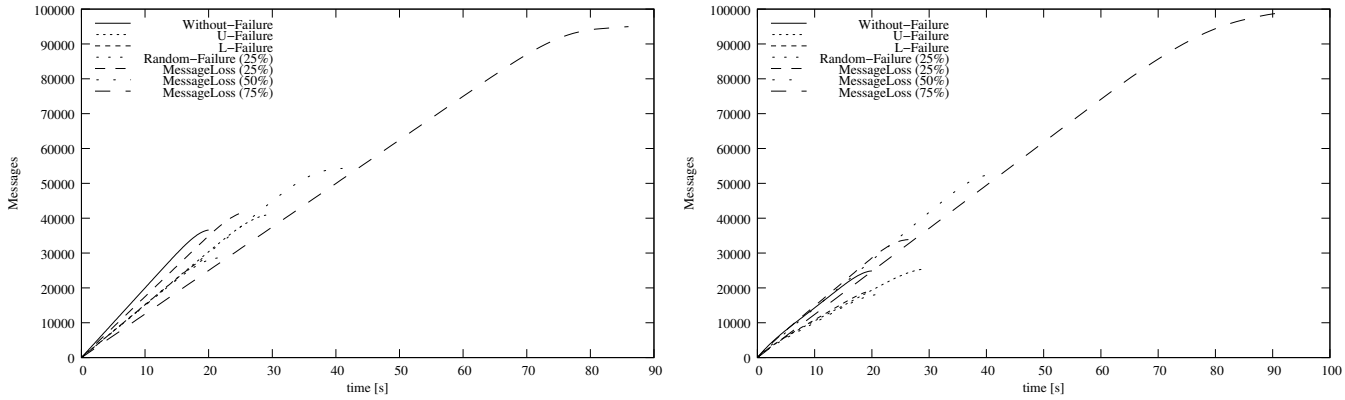
different scenarios. The “refined push gossip” implements the improved local target selection at the information owner (sender) side. The processes last from the first sending of the information sources till the average minimum estimation error is achieved. Comparing the two processes of the convergences of the estimation error, the effect of the refined gossip is that it increases the slopes of the curves under all different failure scenarios. This speeds up convergence towards the minimum average estimation error. The “refined push gossip” is general 5 to 20 seconds faster in reaching the minimum estimation error. Figure 6.7 shows the same process but in terms of corresponding message consumption of “push gossip”, and refined “push gossip”. The refined “push gossip”, not only reduces the sum of message number during the simulation process, but also relaxes the increasing trends of the message numbers. This comparison clearly shows “refined push gossip” improves the convergence process comparing with original “push gossip”.

- **PushPull Gossip vs. Refined PushPull Gossip Protocols (figure 6.8 and figure 6.9)**
  In the case of “push-pull gossip”, the target selection exchange was applied to both the push and the pull processes. Figure 6.9 shows the effect of the refined algorithm in that it helps reducing the total number of messages while speeding up the convergence. Due to the double effect of the target selection implementing at both push and pull processes, the time spent in the dissemination



(a) Push-Pull Gossip (b) Refined Push-Pull Gossip

Figure 6.8: Convergent Accuracy of Push-Pull Gossip vs. Refined Push-Pull Gossip



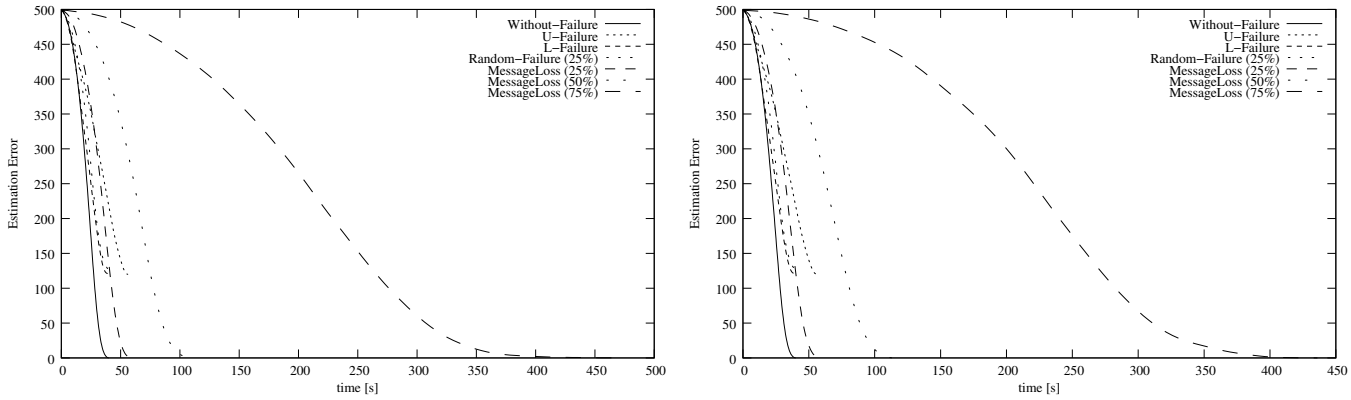
(a) Push-Pull Gossip (b) Refined Push-Pull Gossip

Figure 6.9: Convergent Message Number of Push-Pull Gossip vs. Refined Push-Pull Gossip

process is reduced, comparing with the original protocol. Similar effects can also be viewed from the comparison of "pull gossip" and refined"pull gossip" (figure 6.10 and figure 6.11).

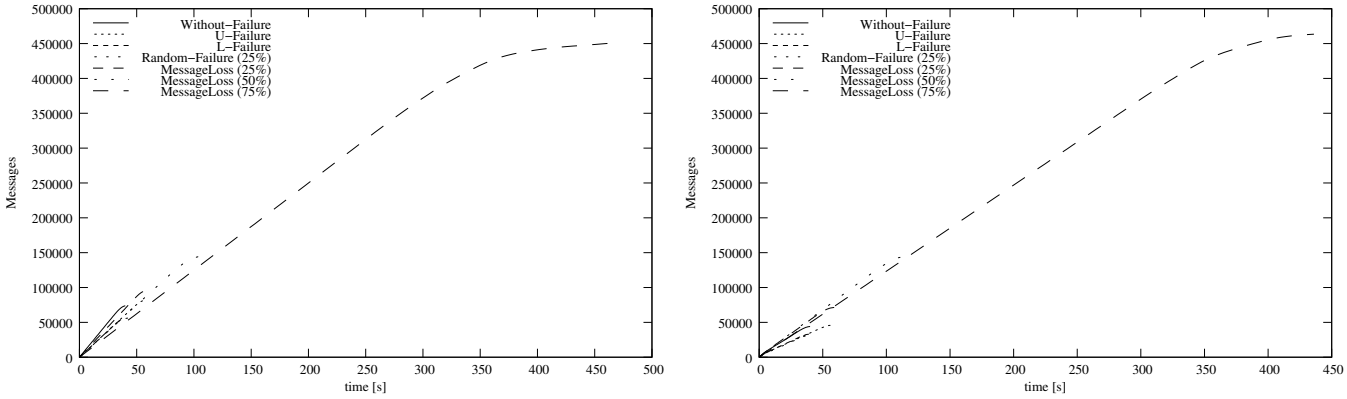
6.1.8 Comparison of Refined Gossip Protocols and Gossip Protocols in Different Scenarios

Figure 6.12 shows the time performance of using different gossip protocols to fuse and disseminate data in normal and in failure scenarios. The time lasts from initial sensing till the average estimation error of the locations reaches minimum on each node of the system. It requires less time to disseminate the fused estimation results among the entire distributed fusion system under the L-failure scenario than under U-failure. The reason is L-failure does not change the maximum information dissemination path length which is the length of the diagonal of the square matrix (with length of $30 * \sqrt{2}$). Yet the maximum information dissemination path length in U-failure is around 3 times $30 = 90$. This shows that the location of failures on the key paths will influence the time performance of aggregation and dissemination. It also shows that the target selective exchange in various refined gossip protocols helps reduce the data dissemination time among the network.



(a) Pull Gossip (b) Refined Pull Gossip

Figure 6.10: Convergent Accuracy of Pull Gossip vs. Refined Pull Gossip



(a) Pull Gossip (b) Refined Pull Gossip

Figure 6.11: Convergent Message Number of Pull Gossip vs. Refined Pull Gossip

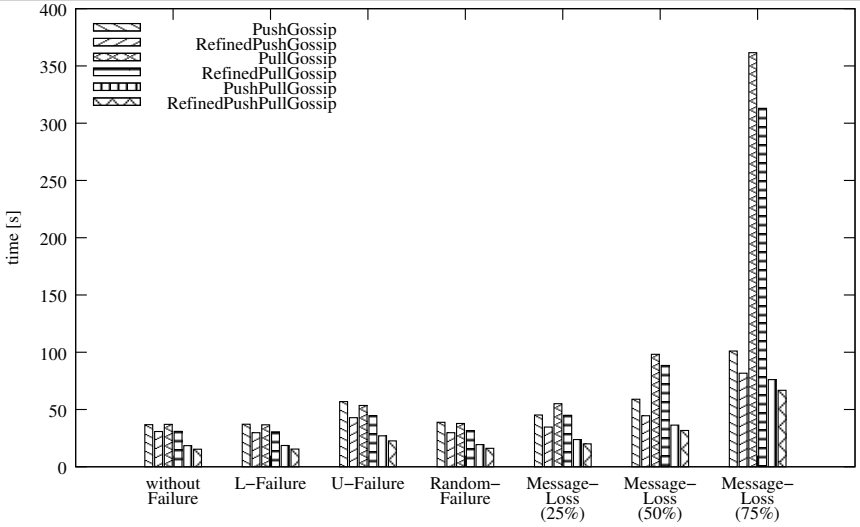


Figure 6.12: Dissemination Time

Figure 6.13 shows the total number of messages consumed for data dissemination in different failure models. The MessageLoss(75%) resulted from unreliable wireless links is the failure mode which increases huge volume of data messages, follows the MessageLoss(50%) and MessageLoss(25%). The several spatial failures (L-Failure, U-Failure, and Random-Failure) do not consume significantly more messages comparing with the normal scenario. It is because that the message drops resulting from unreliable wireless communication links in turn futher deteriorate the communication with extra requirement on retransmitted messages.

In all the failure modes, **"refined push gossip" with target selection** algorithm is the most efficient in the consumption in message consumption. **Refined "push-pull gossip" which uses the target selection** algorithm follows to be the second efficient in messages. This result shows the high effectiveness of refining gossip protocols in reducing the number of wireless messages.

To sum up, the different gossip protocols have different features and apply to different application scenarios. The refined gossip protocols with target selection algorithms adjust the gossip protocols on the wireless platforms by greatly reducing the number of messages transmitted. This makes it feasible to apply gossip protocols on resource restricted wireless communication based devices. Among different gossip protocols, the **"refined push gossip"** is superior in controlling the communication cost, thus enable long system operation time. This can be used in the applications such as long time monitoring using wireless sensor networks which has resource limitation on the sensor nodes. The **"refined push-pull gossip"** is good in data dissemination speed, and can be used in some emergency scenarios to detect changes and report warnings.

The table 6.2 summarizes the features of the different protocols. It can be used to guide the design of the distributed fusion systems to support specific applications. This table shows the evaluation of the features for all the different data dissemination protocols. The mark of + represents the positive score, while the mark of − represents the negative score. For instance, in comparing the performance of the communication cost, refined push gossip is the best, following are push gossip, refined push-pull gossip and the push-pull gossip, while flooding is the worst in this metrics.

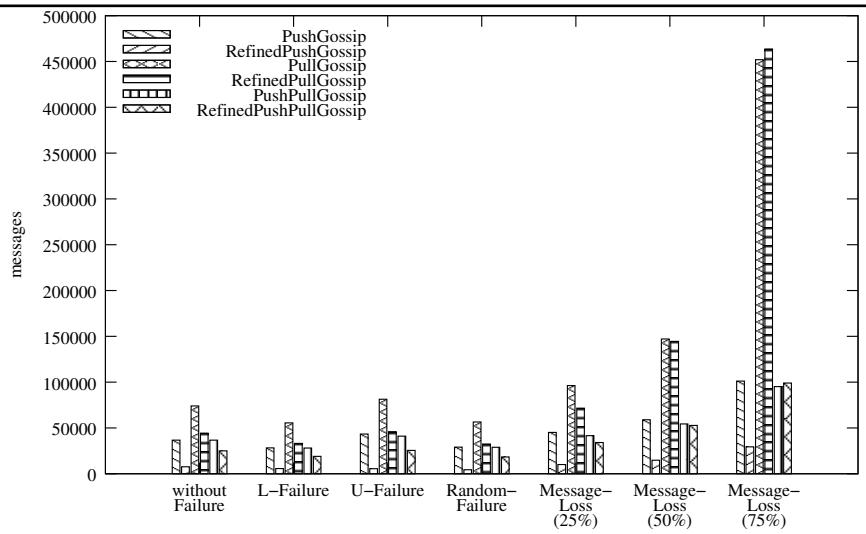


Figure 6.13: Message Number

Table 6.2: Features of Various Data Dissemination Protocols

	push gossip	refined push gossip	pull gossip	refined pull gossip	push-pull gossip	refined push-pull gossip	flooding
communication cost	+	++	-	-	+	+	- -
energy efficiency	+	++	-	-	+	+	- -
dissemination speed	+	+	- -	-	+	++	+++
query based app	-	-	+	+	-	-	-
data source driven	+	+	-	-	-	-	-
data demander driven	-	-	+	+	+	+	-
local memory consumption	-	+	-	+	-	+	-

6.2 Evaluation of Decentralized Change Detection Framework

As introduced in Chapter 1, decentralized change detection framework is a special use case of the distributed data fusion and data harvesting framework. It is built above wireless sensor networks. The framework deploys local change detection function and global detection decision fusion function (as fusion components), on each sensor node to implement the special fusion applications. This section evaluates the performance of this special fusion application framework. This is a joint research work with Dang-Hoan Tran [87].

While Section 6.1 focuses on the evaluation of data exchange protocols, this section focuses on the data fusion dimension of the framework. Accuracy is an important yet application related evaluation metrics for data fusion dimension, which can only be evaluated according to specific application scenarios. In this specific fusion application, detection accuracy is one of the most important evaluation metrics for change detection. Especially, the effect and comparison of different fusion functions and the decision fusion functions is of interest.

This section divided the evaluation of the decentralized change detection framework into two parts: the first part evaluates the performance of the DFT synopsis based local change detection algorithm. For this part there is an experiment evaluation. In the second part, a simulation approach is used to evaluate the

Table 6.3: Effect of Every Basic Window Width on the Accuracy Parameters of Change Detector

Width	Threshold	FA	Precision	Recall	Sensitivity
1	4	0	100	100	6.18
2	5	0.14	56.52	100	3.95
4	7	0.27	34.44	77.50	1.385
8	15	0.27	32.95	72.50	1.226
16	18	0.5	21.17	72.50	0.613
16	20	0.29	25.30	52.50	0.628

performance of the distributed change detection framework to detect the occurrence of special events. The focus of the evaluation of local change detection was accuracy. For the distributed decision fusion, the focus was to evaluate the global estimation error(also the accuracy) of the event location based on the decision fusion model. Furthermore, we investigated the time to disseminate the decision from the information sources to sinks, as well as the communication cost in terms of the number of data messages. The reason to use simulation was to check the performance of the framework in a relatively large scale with sensor network consists of huge number of sensor nodes.

6.2.1 Evaluation on Local Change Detection Algorithm

The change detector was written in Wavescript [30, 31, 58], a high-level, functional, and stream-processing language used to develop distributed, high-speed applications. The experiments were run on a PC with a 2.60GHz 2x Pentium (R) Dual-Core CPU and 4GB memory, Linux platform.

We thoroughly studied the change detection algorithm in practice with synthetic and real data sets. To make it easy to observe the results without losing the accuracy of our experiments, the results of changes are observed in a sliding window of size 256.

Our change detectors were evaluated in many aspects. The first group of experiments tested the effect of basic window sizes on the accuracy of change detection. The second group of experiments analyzed the effect of threshold selection on the accuracy. Finally, the third group of experiments compared the performance of change detectors using different distance functions.

6.2.1.1 Effectiveness of Basic Window

To assess the effect of basic window width on the accuracy of change detector, the following tasks were executed. For each window width (1,2,4,8,16), we computed the performance parameters by specifying a given raw threshold (temperature 25.9°C) and changing the distance-based threshold in a certain range. After that, we chose a threshold considered as a "nearly optimal" threshold.

As shown in Table 6.3, increasing the width of basic window results in decreasing in both precision and recall. The sensitivity of change detector decreases with an increasing basic window width because the estimation error of the distance function increases when the window width increases. There are two interesting cases in our results. First, when experimenting with a window width of 1, an interesting result obtained was that the rate of hits $Pr(H) = 1$ and the rate of false alarms $Pr(FA) = 0$ (with distance-based

Table 6.4: Comparison of the Effectiveness of the Euclidean Distance and the Manhattan Distance

	Euclidean				Manhattan			
Width	FA	P	R	S	FA	P	R	S
1	0	100	100	6.18	0	100	100	6.18
2	0.14	56.52	100	4.17	0.14	55.71	100	4.17
4	0.27	34.44	77.5	1.385	0.27	34.83	77.5	1.385
8	0.22	38.16	72.5	1.385	0.22	38.16	72.5	1.385
16	0.29	25.3	52.5	0.628	0.29	17.72	35	0.139

threshold 4). We had such an ideal result because when the window width equals to 1, the Euclidean distance reduces to $d = |x_i - y_i|$. Therefore, the distance-based change detector reduces to the raw threshold-based change detector which is used to detect the truly changed points and unchanged points. Second, in some cases, it is difficult to choose the most suitable thresholds if we only use two parameters precision and recall. For example, with a window width of 16, if we were only interested in the highest rate of hits (0.725), we would choose the threshold (18) corresponding to this rate of hits. Unfortunately, this threshold is not acceptable, because the rate of false alarms (0.5) is too high. A different approach to choose a suitable threshold is that we used the sensitivity, a parameter combining both the rate of hits and the rate of false alarms. Therefore, the better threshold is 20. Our results agree with previously published work confirming that the window of small size is sensitive with abrupt change. Additionally, we realized that selecting of window model is affected by not only two parameters of change detector (precision and recall), but also on other parameters of change detector (probability of hits, probability of false alarms, probability of misses, and sensitivity). From a practical point of view, our work can suggest that it should be better to exploit multiple parameters and the context of specific applications in order to develop an effective change detector.

6.2.1.2 Effectiveness of Distance Measure

To compare the Euclidean distance and the Manhattan distance used in our change detector, we used the Neyman-Pearson criterion as follow. First, for each basic window with given width, we computed the accuracy parameters corresponding to the distance-based thresholds. Second, the rate of false alarm was set to $Pr(FA) \leq 0.3$, the distance-based threshold and the accuracy parameters corresponding to the biggest rate of hits were chosen. After that we determined the accuracy parameters of the Manhattan distance-based change detector corresponding to the previously specified rate of false alarms of the Euclidean distance-based change detector. Table 6.4 shows the results of this group of experiments.

With a basic window of small width, there is little difference between the sensitivity, the Precision, and the Recall of the Euclidean-based change detector and the Manhattan-based one. But with a wider window (16), the sensitivity (0.628), Precision (25.3%), and Recall (52.5%) of the Euclidean distance-based change detector are greater than those of the Manhattan distance-based one (Sensitivity: 0.139, Precision: 17.72%, Recall: 35%). This results agrees with previous work on the evaluation of distance functions in the change detection in data streams presented by Bud et al. [9]. But in addition to evaluating the precision and recall, we considered the sensitivity of change detector.

Much previous work in change detection only focus on analysis of effects on probability of hits and

probability of false alarms or precision and recall without considering the effectiveness of threshold selection on sensitivity of change detection. The smaller the threshold is, the greater the probability of false alarms. However, if the threshold is too big, the probability of hits is small, or in other words, the probability of missed changes is large. Contrary to the intuitive belief, larger precision and recall does not really mean better results for change detection algorithms [54]. If the threshold increases, the precision increase, too, but the recall decreases. The rate of false alarms is proportional to the threshold. Therefore, to select an appropriate threshold, we must consider multiple parameters at the same time instead of considering individual parameters. This experiment makes the following choices: the suitable threshold in this case is 5, because its sensitivity (2.727), precision (95.65%), and recall (56.41%) are the best choice. Depending on specific application, the threshold can be chosen suitably. In a scenario of wildfire warning system, the probability of misses should be as small as possible, but in stock market related applications, both probability of false alarms and probability of misses should be limited.

6.2.2 Evaluation of Distributed Change Detection Frameworks

6.2.2.1 Simulation Scenarios

We created a simulation scenario where a network system consists of nodes with sensing ability that can be used to detect new changes. In a wireless sensor network where 500 nodes were deployed, we created four event locations and switched them on at 0.1 second after the simulation start, to simulate the events. The locations of the four events were randomly initialized. The aim of this simulation was to see how fast this change can be detected and learned by the other sensor nodes, and further how accurate the location of event could be estimated by all the nodes, as well as the general communication cost for disseminating the fire warning.

To simulate this scenario, we used OmNet++ 4.0 and its INET framework to provide an 802.11 MAC layer. So the protocols for comparison were all implemented on the application layer. For instance, the flooding we used here was not a physical layer radio broadcast, but data message flooding.

6.2.2.2 Simulation Analysis

We compared four different kinds of protocols in exchanging the local change detection data message, to show the flexibility of this framework in different scenarios.

Figure 6.14 showed the change of average estimation error on all the nodes with simulation time increases. At the beginning of the simulation, only nodes nearby the event directly detected the changes, while most of the other nodes had a N/A state. With the information exchange protocol transmitting the detected information among sensor nodes, and with the triangulation location algorithm executing on improved data source, the event was learned by more and more nodes. This reduced the estimated location error. After 5 seconds of decision fusion runs, 30% sensor nodes using PushPullGossip learned the event and have knowledge on the event location with error smaller than 1 meter. Among the four data transmission protocols, PushPullGossip was the fastest to reach the lowest estimation error, because it exchanged data using both push and pull approaches. Flooding followed PushPull with high communication cost in terms of data message numbers as shown in figure 6.15. PushGossip and PullGossip spent twice as longer as PushPullGossip for all sensor nodes to learn the event location.

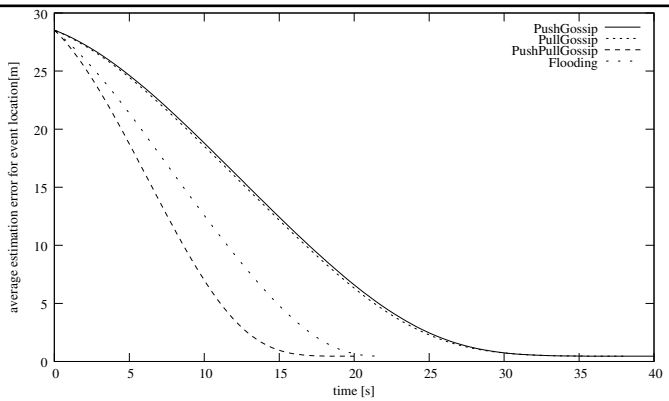


Figure 6.14: Location Estimation Error

Figure 6.16 and figure 6.17 showed estimation error for the event location and the message number in the scenario where 25% nodes failed. The distribution of failed nodes conformed to the binomial distribution, and the failure occurred randomly in space.

Comparing with a normal case where no nodes failed, the difference was that the final estimation error was obvious, around 7 meters. In this case, some nodes, which were around the event location and can directly detect the fire, failed. But in general the framework still functioned similarly to the normal situation in terms of time performance. This meant that the events were detected and learned by all the nodes but with a relatively bigger range of where the fires occurred.

In comparing communication costs, figure 6.15 and figure 6.17 showed that PushGossip was the most efficient in both normal and random failure scenarios.

We used this special example to detect events to demonstrate the broad utility of data fusion and data harvesting framework, since decentralized change detection framework is a specialized application and use case build upon the general data fusion and data harvesting framework. The simulation results showed that the distributed detection decision fusion framework can effectively fuse local event detection results to generate a more accurate event location. Besides, it could transmit and disseminate the information from the information source to the sink (in this case, a multi-source multi-sink scenario). In general, we wanted to demonstrate that the framework was flexible in that one can choose different communication protocols to achieve different design aims (such as low communication cost, or good time performance, or fault tolerance), and also different decision fusion models on sensor nodes according to different applications.

6.2.3 Summarization of the Results

In regards to the specialty of the fusion functions in this fusion application: the change detectors are closely related to the meta-algorithm for change detection proposed by Kiefer et al. [45]. To detect change in data streams, they compare the distributional distance between two sliding windows with a given threshold. Their approach requires no prior assumptions on the nature of the data distribution. They can compare two sliding windows of different sizes because the distance used by their algorithm is the Kolmogorov-Smirnov statistical distance. However, this algorithm is not well suited for sensor network

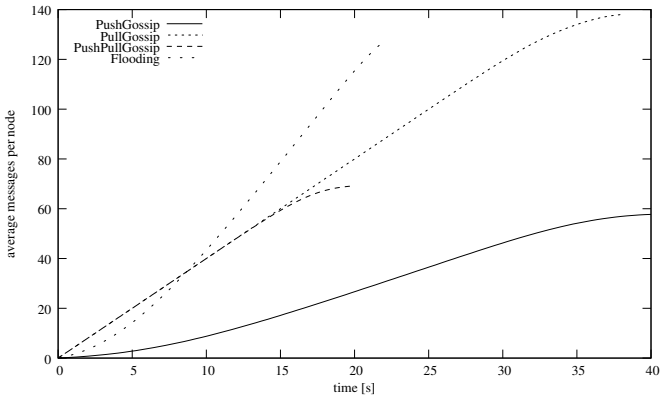


Figure 6.15: Message Number

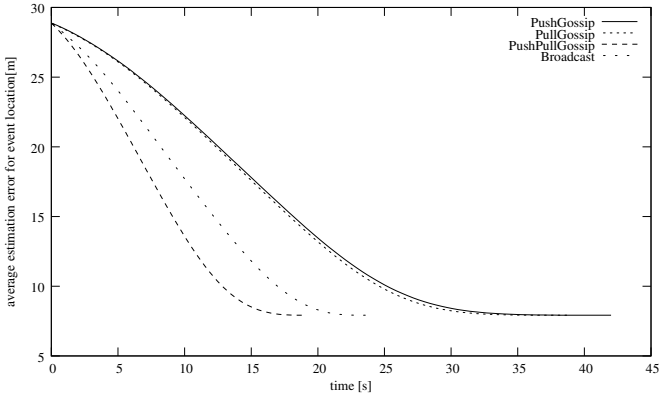


Figure 6.16: Location Estimation Error in 25% Nodes Failure Scenario

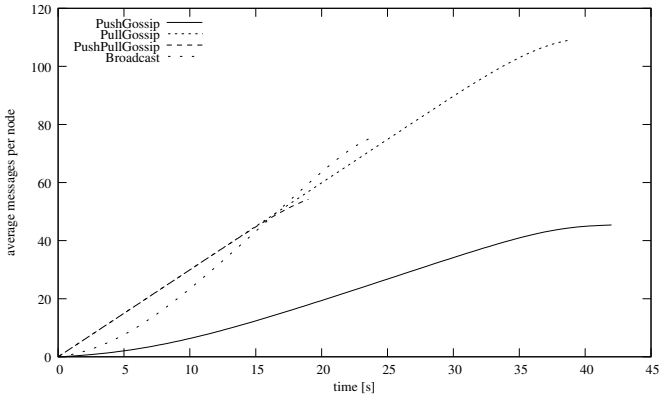


Figure 6.17: Message Number in 25% Nodes Failure Scenario

applications because computing the Kolmogorov-Smirnov has high computation complexity and some other important limitations [65]. For example, the K-S distance applies only to continuous distributions. Perhaps, the most serious limitation of the K-S distribution is that it typically must be determined by simulation. Therefore, we have used in our work simple geometric and algebraic functions such as the Euclidean and the Manhattan distance as distance function. Another drawback of their sliding window model is that it causes difficulties in evaluating the accuracy of change detector.

In order to find a suitable solution to the change detection problem in resource-restricted settings of sensor networks, we have adopted the principle of work of Kiefer et al. In our change detectors, the two-window-paradigm is exploited, but we have improved this model by using the sliding window model first introduced by Zhu et al. [102].

To summarize, in this section the decentralized data fusion and data exchange framework is applied for decentralized change detection in wireless sensor networks. Such framework is a special application use case of the data fusion and data harvesting framework. This framework consists of both a local change detection models based on DFT synopsis, and a global decision fusion model for determining the event location. The local change detection model and the global decision fusion model both can be regarded as specialized fusion components locates on each sensor node.

For local change detection, the sliding window model is exploited to detect local changes in sensor data streams. We used signal segments to reduce the memory footprint of the change detector. The locations of events (i.e. changes in the environment) are estimated using a global decision model based on gossiping. By choosing different models of local change detection, global decision fusion, and data exchange protocols, different detection framework are generated and different design goals can be achieved.

The experimental evaluation demonstrated that for the local change detection fusion component, the Euclidean distance based change detector is better than the Manhattan distance-based change detectors, especially in terms of detection accuracy. For the decentralized decision fusion framework, the decision fusion models and functions efficiently improve the accuracy of event location estimation, as time increases. Specifically, push-gossip can be implemented for power efficiency design aims, while push-pull-gossip is a good choice in reducing decision, which is the key factor in emergency applications. These results conformed to the general results on data exchange protocols from Section 6.1.

6.3 Application Use Case: Query Fusion Framework

The decentralized data fusion and data harvesting framework can even be used to support some special fusion applications. In this section, the query fusion framework is introduced, this section is based on the joint research work with Tianli Mo, Lipyeow Lim and et al on [95].

In this framework, the input to the fusion components or fusion functions are some queries and data sets. The output of the fusion functions can be either maximum common queries or some data sets as final or middle results. The query fusion function fuses the different incoming queries and extracts a maximum similarity part from these queries.

Since the fusion technique here is unique, the evaluation focus is still on the dimension of the data fusion, to analyze the performance of these fusion components. Section 6.2 already evaluates this dimension of the framework from the perspective of data accuracy resulting from decentralized data fusion

and harvesting framework. The difference is that: this section shows the efficiency in terms of energy as a result of executing the several query fusion functions to achieve the same goal. This demonstrates the influence of the data fusion techniques itself on the energy consumption of the framework.

This section describes the results from our simulation-based experiments to quantify the performance gains of using query fusion (in terms of the reduction in energy overheads) of the proposed collaborative query processing framework. Our experiments are conducted using a Python-based simulator which accepts as input groups of phones, phones' queries and the data streams required by the queries. The query evaluation at each phone is performed periodically at some interval of time. Each simulated query may require some amount of remote data from the server and some amount of data from the local sensors. Hence we investigate the performance of the framework under varying amount of local sensor data, varying amount of remote server data, and varying the evaluation interval. In particular, we compare the energy performance for the five different query processing strategies introduced in the following: Naive-P, Naive-S, CQ-S, CQ-L and CQ-LS. These five strategies are introduced in the Section 4.9.3.

6.3.1 Evaluation Metrics

The results are aggregated as an average performance per execution interval over the entire simulation time.

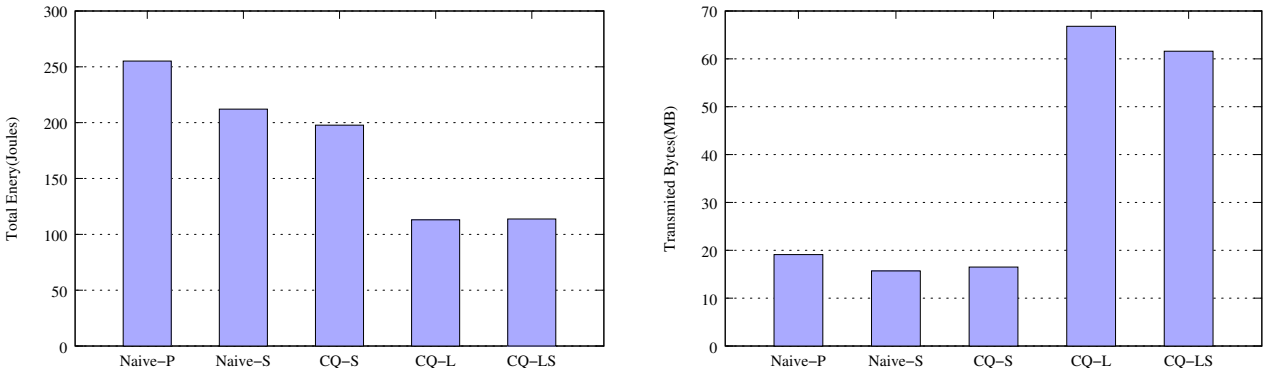
For the streaming local sensor data required by the query at each evaluation interval, we generated the value of each attribute in the streams using the normal distribution $N(\mu, \sigma)$ (with appropriate truncation to avoid underflow below 0 or overflow above 100%). The data size of the streaming local sensor data is determined using a parameter called the ratio between local sensor data to remote server data. This ratio varies on each phone, and is randomly selected between 0.25 and 4 in increments of 0.25.

We use the following metrics in our experiments:

- **Energy consumption:** we evaluate the energy consumed on each phone as well as the sum of the energy consumption on all phones.
- **Number of bytes transmitted:** since energy consumed on data transmission is several orders of magnitude larger than the energy consumed on computations, we focus on the quantity of data transmitted among phones (using Bluetooth), servers and phones (using 3G).
- **System operational lifetime:** the lifetime of the system usually depends on the performance bottleneck of one device, thus we work on finding the bottleneck device (such as group leader)'s time to battery depletion.
- **Group size:** the group size is a significant attribute influencing the query sharing. In addition, group size is restricted by the communication range of wireless techniques.

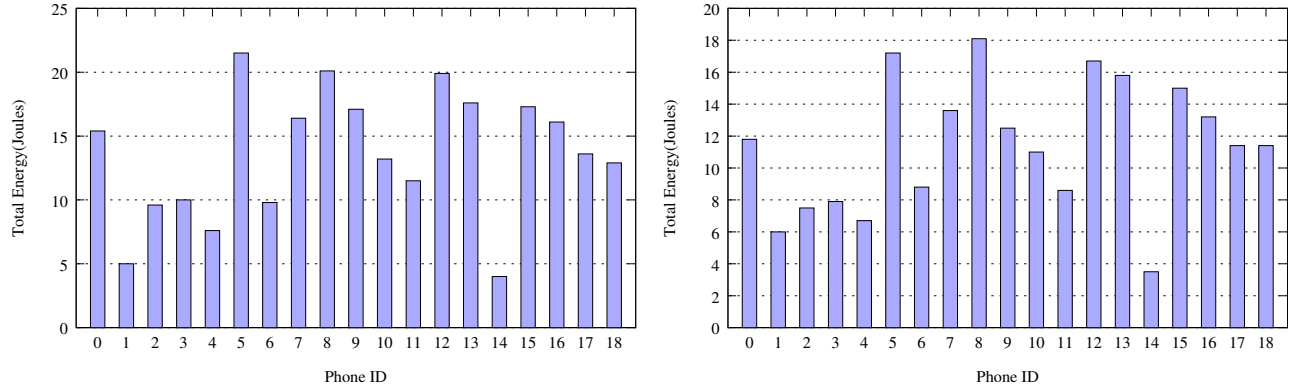
6.3.2 Energy Consumption

This experiment addresses the questions: "To what extent can the collaborative query scheme help reduce energy consumption in the system?" and "What does the energy consumption profile look like on each phone?" Since the energy efficiency is the main goal for the framework, the performance of this metric (energy consumption) decides whether this chief design goal is met or not.

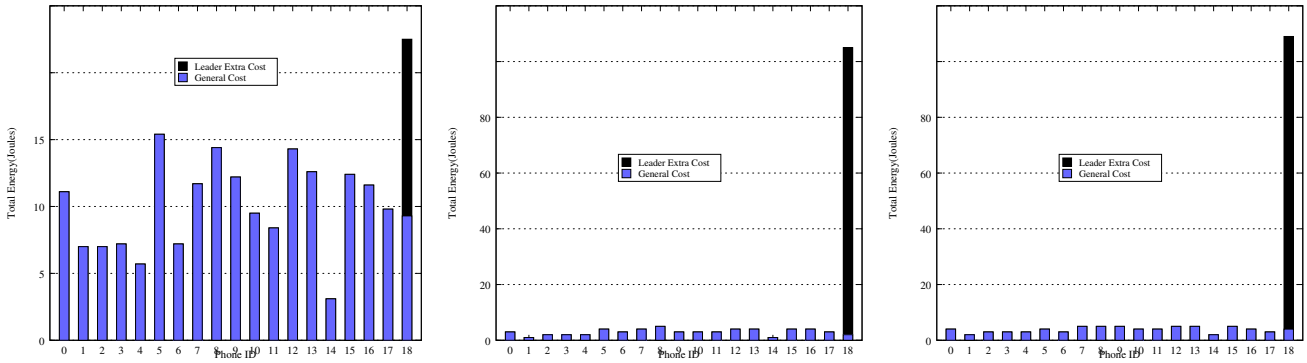


(a) Total Energy Cost (b) Total Bytes Transmitted

Figure 6.18: The Energy Cost and the Number Bytes Transmitted for the Five Methods



(a) Naive-P (b) Naive-S



(c) CQ-S (d) CQ-L (e) CQ-LS

Figure 6.19: Distribution of the Energy Consumption over All the Phones

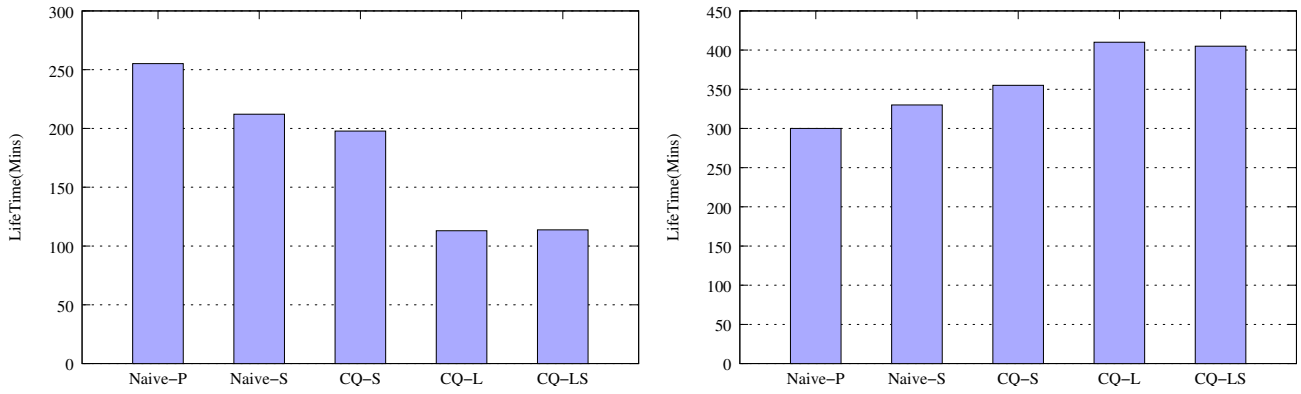
Figure 6.18(a) gives the total energy consumed for the five different methods and shows the advantage in energy consumption of the methods proposed in our framework. The phones we used in the scenario consists of 18 smart phones in one group sending out queries to be answered. The interval to process queries within one query-reply round is 20 seconds. During each interval, at the server side, 10KB data are generated. In each query-reply interval, the Naive-P and Naive-S methods consumed, among the 18 smart phones in total, over 215 Joules to answer the queries. In the 3 sharing schemes (CQ-S, CQ-L and CQ-LS), the 18 phones are grouped into one location-based group and one query-based group. These enable both sharing strategy among group members and transmission optimization technologies. CQ-L method prevents every member phone from acquiring remote data with 3G, resulting in energy consumption to be half of the naive methods. In CQ-LS, the leader takes the responsibility to process queries for member phones as proxy, results in similar energy reduction effect as CQ-L.

Figure 6.18(b) shows the total data transmitted averaged during a round of query-reply. Naive-S uses the centralized server to process the query, thus is usually the lower bound of the data needed to be transmitted. CQ-S adopts sharing strategy on the server, thus requires a bit of overhead in coordinating the sharing processes. CQ-L and CQ-LS both requires large amount of data transmitted among leader and member to coordinate the sharing process. Comparing with Figure 6.18(a), the sharing methodologies although they cause overhead in coordinating sharing by using cheaper technologies (Bluetooth), they are effective in reducing transmitting original data sources and partial query results directly in 3G. This effect actually reduces the final energy consumption in total.

Figure 6.19 goes into details of the profile of the energy consumption on each phone in the five methods. The sharing schemes proposed in our framework in general decrease the energy consumption on each member phone, with the price of causing extra overhead on the leader mobile phone to coordinate sharing process. This brings out a new problem of adding an obvious bottleneck of leader node into the system, which might influence the lifetime of the entire system. Therefore, dynamically choosing the leader is necessary to extend the lifetime of system.

6.3.3 System Lifetime

Second to the energy consumption of the system, another interesting and important question is “how long can the system work in the collaborative query sharing scheme?” If the system’s lifetime is greatly reduced, it might affect of the willingness of the owners of smart phones to utilize the collaborative sharing scheme. The system’s lifetime is decided by the weakest yet important device. So this experiment tries to find out the shortest lifetime of an important device in the system. In general, one smart phone (for instance, iPhone’s battery) holds 10000 Joules to 20000 Joules energy. This capacity and the query processing methods decide the lifetime of one smart phone. In order to solve the problem of fast depletion of battery on leader node in our sharing framework, we introduce a dynamic leader method. In general, each node within group is chosen as leader in turn according to their current available battery level. This method effectively extends the life time of the system, as shown in figure 6.20, especially CQ-L achieves an extension of around 57% over Naive-S.



(a) Fix Leader (b) Dynamic Leader

Figure 6.20: Life Time of System

6.3.4 Influence of Group Sizes

In different applications, the distances between smart phones influence the distance based group size. The similarity of the queries on mobile phones will influence the size of the query groups. "How does the scale of the group influence the effect of query sharing" is examined in this experiment. The answer to this question helps understand the scenarios to use our framework.

Figure 6.21 shows the influence of the size of the location based groups to the total energy consumption in five different scenarios. In the experiment, the total number of phones are 36. The query sets are the same in the three grouping scenarios. In general, the larger the group is the more effective the query sharing strategy could be. This can be understood intuitively, a larger group leads to greater sharing potential. Figure 6.22 shows the influence of the size of query based groups to the total energy consumption in five scenarios. A total of 16 phones are grouped into query based groups of sizes 2, 4 and 8 individually. The conclusion is that larger sharable group leads to more sharing part and less differential part, thus is more energy efficient.

6.3.5 Local Data vs Remote Data

This experiment aims to answer the question "How does the ratio of local data to remote data affects the adoption of the query sharing schemes?" This ratio is one of the bases to choose proper sharing schemes in different applications. Since it is a cooperation of the query node and the information source to jointly complete queries' execution, the decision of adopting a specific sharing or even naive solutions is largely dependent on where and how the data streams meet. This is another chief factor decided by specific applications. The following three changes might trigger the changes of the ratio of local and remote data:

- Changes of the frequency of the sampling of the data streams locally or remotely on the server.
- Changes of the locations where for the data sources (local or remote).
- Changes on the queries may cause the completion of the queries using different ratios of local and remote data.

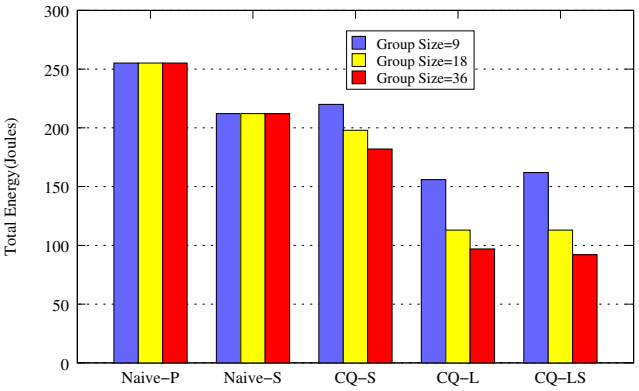


Figure 6.21: Number of Group Size

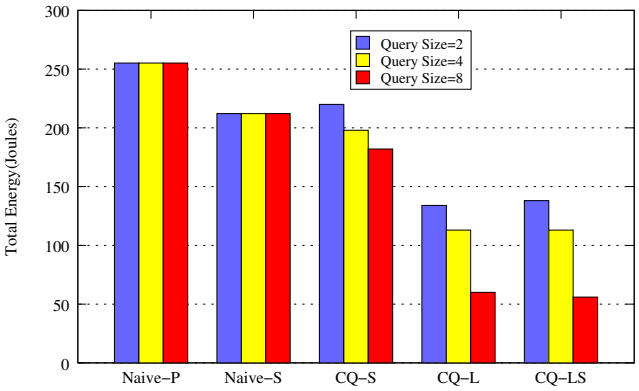


Figure 6.22: Number of Query Group Size

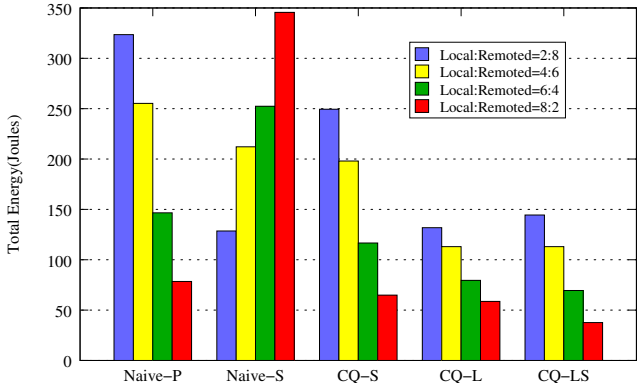


Figure 6.23: Ratio of Local Data vs. Remote Data

Figure 6.23 shows the influence of the ratio of local and remote data on the energy consumption. Although the collaborative sharing techniques in general have less energy consumption than the naive methods. In some special scenario, the ratio of local and remote data is 2:8, which means that a large amount of data is actually available on the server, Naive-S is more effective than the shared strategies. In conclusion, the adoption of a proper method to answer queries depends on many factors and the optimization process should consider generating a reasonable optimized collaborative query execution plan for different scenarios.

6.3.6 Summarize the Results and Evaluation of Query Fusion

As a special type of data fusion, query fusion combines the advantage of using different communication interfaces and the techniques in multiple query optimization to reduce the energy consumption and extend the system lifetime of the distributed query processing system built upon heterogeneous, resource-restricted devices.

The framework is used for energy-efficient continuous evaluation of multiple complex queries over mobile sensing data streams. The framework uses two optimization techniques for sharing query execution and sensor data streams among multiple mobile nodes. The key to the query optimization framework is the automated identification of the similarity of the queries among different mobile users, and the execution of shareable fragments of multiple queries on a common 'master' mobile node. The framework further reduces energy overheads by using low-energy wireless interfaces (such as Bluetooth) to exchange data and query state directly between nearby smart phones. Our results on synthetic traces indicate that, compared to existing purely centralized or decentralized solutions without any query fusion techniques, the hybrid framework can result in 60% reduction in the energy overheads, and 40% to 65% increase in system operational life time (if the 'leadership' role is rotated dynamically).

6.4 Implementation of the Data Fusion and Data Harvesting Software Framework

This section discusses the implementation of the software framework which integrates the three dimension components of data fusion, data exchange and configuration parts. The software framework can be a middleware platform installed on all kinds of devices to support the data centric applications based on data fusion and data harvesting. The following sections describe the steps of the general architecture, the implementation and mapping of the directed fusion graph to the real software framework.

6.4.1 Architecture of the Distributed Data Fusion software framework

In order to integrate the three components of data fusion, data exchange and self-configuration as software plug-ins, each node in the heterogeneous distributed system should embed the three-layer architecture shown in fig. 6.24. At the lowest layer are the ZeroMQ library and the Data Exchange Library. The Data Exchange Library translates the YAML system specification into the local ZeroMQ data gates mainly using the YAML's loader. Besides, this library provides the interfaces of data gates for the fusion

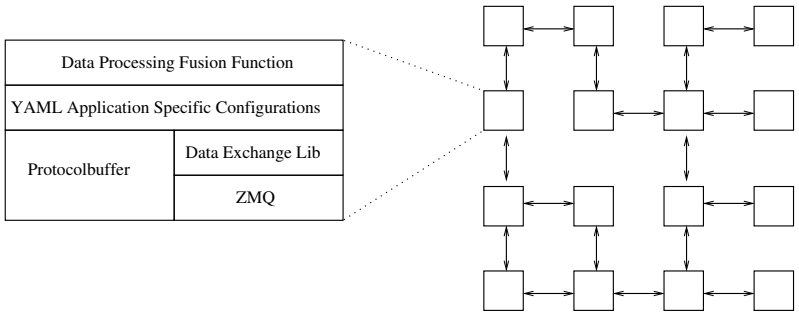


Figure 6.24: Software Architecture for ZMQ based Configuration

tasks to call and transport data, so that each component can retrieve the ZeroMQ socket address by giving the name of the component and the name of the gates within the component. These core functions are implemented in python language. To enable the fusion algorithms written in other languages, for each different language, the library should provide the interfaces written in that specific language. For example, there should be C/C++ based library to support the fusion functions written in C/C++, etc.

At the middle layer is the YAML specification for the entire system. From this specification each component is able to "understand" the logical structure of the entire system and its own placement and connections to the other components. This specification can be edited or rewritten dynamically to represent the changes to the structure of the system.

At the highest layer, the Data Processing (Fusion) Functions process the input data streams with certain data fusion algorithms to generate the results data streams, through the specified gates and connections.

ZeroMQ is the open source socket library that acts as a concurrency framework, the socket library supports applications for clustered products and supercomputing and it is faster than TCP. It carries messages across IPC, TCP and multicast. The connection modes that ZeroMQ supports including connect N-to-N via fan-out, pubsub, pipeline, request-reply, asynchronized I/O for scalable multicore message-passing applications [99]. In this work, the application layer data gates are generated using ZeroMQ sockets, these sockets connect the other sockets embedded in the other fusion components through network, while the sockets also input and output data from the fusion functions within the fusion component.

Fig. 6.25 shows the details about how three nodes interconnect with one another through the sockets. Each node can host several fusion functions, which are specified in the configuration plan documented in YAML. The interconnections among these fusion functions are ZeroMQ sockets which are generated run-time according to the configuration plan, with the support of the data exchange library. ZeroMQ connects the data fusion software components with sockets which support the data exchange among internal data fusion components and the implementation of flooding or gossip protocols which are between nodes. The YAML configuration defines the data exchange (sockets connections) among the nodes. And the data exchange components either statically or dynamically use these sockets connections for data exchange in each round.

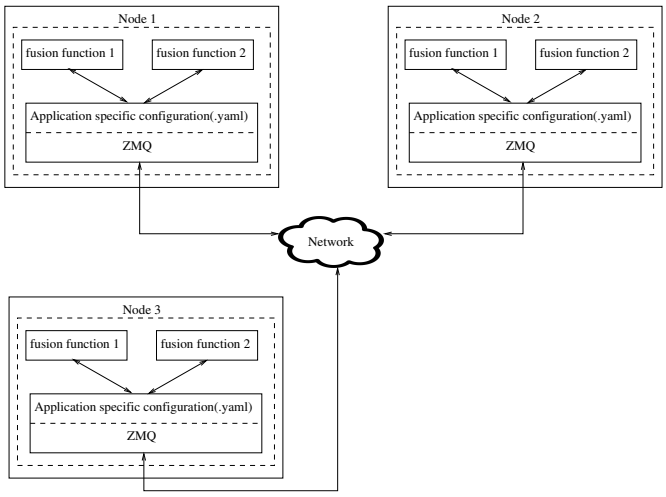


Figure 6.25: Distributed Deployment of Fusion Functions on Distributed Networked System

6.4.2 Implementation of the Software Framework for Resource Map Generation

Implementing data fusion functions relies on different applications. This section shows the implementation of the software framework used for resource map generation. The use case itself is explained in detail in Section 1.2.1. A prototype was build with clusters to show the configuration process. Although the configuration is not directly implemented on wireless communication based nodes, this is the initial implementation of the data fusion and harvesting system based on the flexible software architecture.

This use case shows that three devices cooperates with one other in measurement data collection, exchange through gossip protocols and then fused at one of the machine to generate the estimated base station location. Three data sources are a minimum number to enable location estimation algorithm, through the triangulation method. All three machines carry the ZeroMQ library together with the application libraries of fusion components, all these libraries account to around 3 Megabytes.

In this scenario, device with IP: 141.24.32.195 and device with IP: 141.24.32.173 generates measurement data in the Measurement component. The measurement data through the gate of localData, by gossip protocols, was sent to the device with IP: 141.24.32.192. At the device of IP: 141.24.32.192, the external data through gossip protocols arrives at the gate of externalData of the Gossip component, and go out through the gate of exchangedData. The data later arrives at the fusion component of Location Estimation through its gate of exchangedData. The other input of the fusion function of the Location Estimation is the local measurement data; the inputs of data get combined and fused inside the Location Estimation component.

Figure 6.26 shows the configuration plan - the directed fusion graph - to support this application. The corresponding YAML file describing this is as figure 6.27.

The gossip components, during configuration process should set up the data exchange sockets among all the neighbors. This is similar to the fusion graph for the flooding architecture. The control and usage of these sockets are managed by the gossip protocol, in each iteration, only selected ZeroMQ socket(s) is(are) used to exchange data between neighbors. The corresponding algorithms for using push-pull gos-

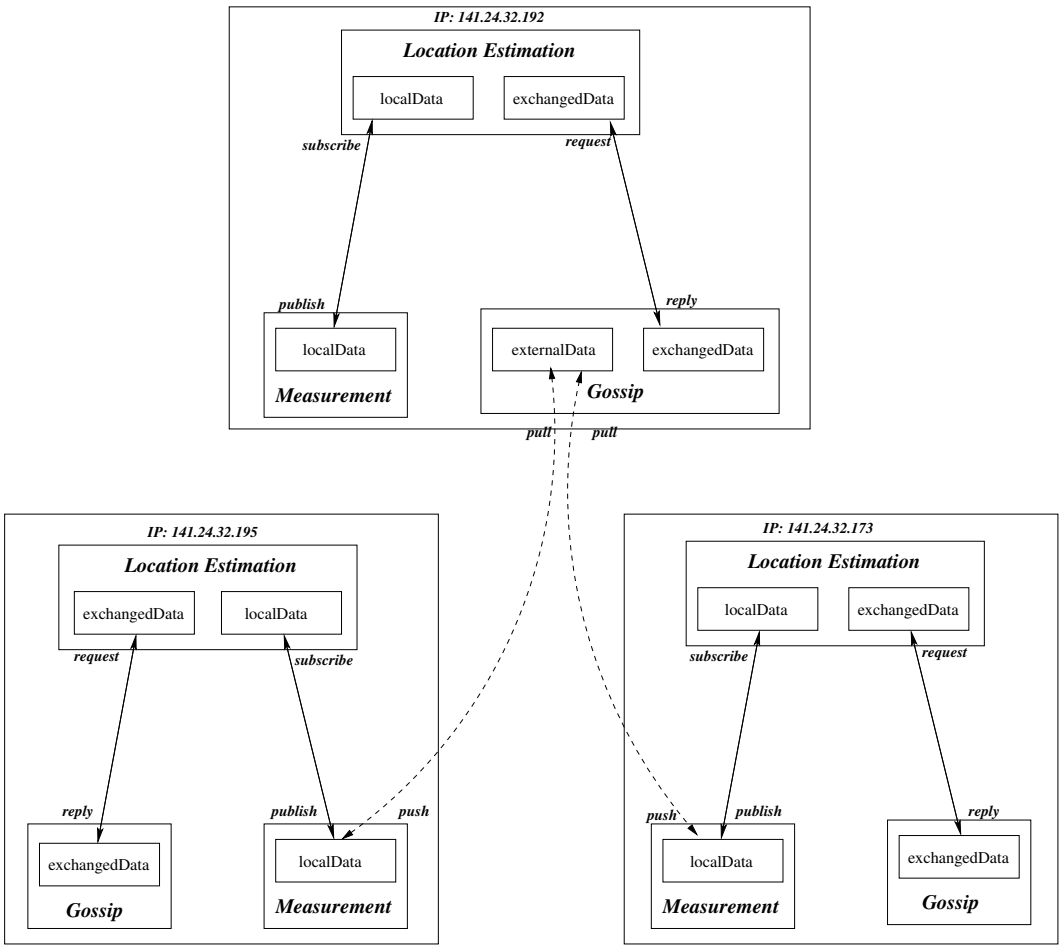


Figure 6.26: Configuration of Decentralized Directed Fusion Graph for Resource Map Generation

sip as data exchange among the data fusion components are shown in Algorithm 9 and Algorithm 10. The location estimation fusion function runs on node with IP: 141.24.32.192, needs the measurement data collected from three nodes in one area. Through data exchange with the two other selected nodes with IP: 141.24.32.195 and IP: 141.24.32.173, the measurement data from these two nodes can be combined with local measurement data from IP:141.24.32.192 as input of the location estimation fusion function. The data exchange structure is valid in each round of the gossip process, in the next iteration; two other neighbor nodes might be selected to exchange data with 141.24.32.192. These two algorithms show how the components of data fusion functions are integrated with gossip protocols, exchanging data through the sockets set up by specified configuration file of YAML listed in the figure 6.27.

Algorithm 9 Location Estimation at IP 141.24.32.192: Active Thread

Input: Local measurement data at 141.24.32.192, and data from selected two neighbor nodes
Output: A location estimated for the base station as updated new status

1: **begin** active thread:
2: *do once in each consecutive iteration:*
3: q_1, q_2 = Select2Neighbors(); ▷ in the iteration shown in Fig. 6.26, *.195, and *.173 are selected
4: $status_p = localMeasurementData_p$;
5: Send($status_p$) to q_1, q_2;
6: $status_{q1}$ = Receive($externalDataFrom_{q1}$);
7: $status_{q2}$ = Receive($externalDataFrom_{q2}$);
8: $locationEstimation_p = locationEstimationFusion(status_p, status_{q1}, status_{q2})$;
9: $status_p$ = Update($locationEstimation_p$);
10: **end** active thread

Algorithm 10 Location Estimation at IP 141.24.32.192: Passive Thread

Input: Local measurement data at 141.24.32.192, and data from selected two neighbor nodes
Output: A location estimated for the base station as updated new status

1: **begin** passive thread:
2: *do forever:*
3: $status_{q1}$ = Receive($externalDataFrom_{q1}$); ▷ passive receive data from node 141.24.32.195
4: $status_{q2}$ = Receive($externalDataFrom_{q2}$); ▷ passive receive data from node 141.24.32.173
5: $status_p = localMeasurementData_p$;
6: Send($status_p$) to q_1, q_2;
7: $locationEstimation_p = locationEstimationFusion(status_p, status_{q1}, status_{q2})$;
8: $status_p$ = Update($locationEstimation_p$);
9: **end** passive thread

```
components:
        - name: measurement-A
          ip: 141.24.32.192
          gates:
          - localData: pub

        - name: locationEstimation-A
          ip: 141.24.32.192
          gates:
          - localMeasurement: sub
          - exchangedData: rep

        - name: gossip-A
          ip: 141.24.32.192
          gates:
          - exchangedData: req
          - externalData: pull

          - name: measurement-B
          ip: 141.24.32.195
          gates:
          - localData: pub

        - name: locationEstimation-B
          ip: 141.24.32.195
          gates:
          - localMeasurement: sub
          - exchangedData: rep

        - name: gossip-B
          ip: 141.24.32.195
          gates:
          - exchangedData: req
          - externalData: pull

         - name: measurement-C
          ip: 141.24.32.173
          gates:
          - localData: pub

        - name: locationEstimation-C
          ip: 141.24.32.173
          gates:
          - localMeasurement: sub
          - exchangedData: rep

        - name: gossip-C
          ip: 141.24.32.173
          gates:
          - exchangedData: req
          - externalData: pull

connections:

- ["udp://141.24.32.192:5556", "measurement-A | localMeasurement", "locationEstimation-A | localData"]
- ["udp://141.24.32.192:5555", "gossip-A | exchangedData", "locationEstimation-A | exchangedData"]

- ["udp://141.24.32.195:5556", "measurement-B | localMeasurement", "locationEstimation-B | localData"]
- ["udp://141.24.32.195:5555", "gossip-B | exchangedData", "locationEstimation-B | exchangedData"]

- ["udp://141.24.32.173:5556", "measurement-C | localMeasurement", "locationEstimation-C | localData"]
- ["udp://141.24.32.173:5555", "gossip-C | exchangedData", "locationEstimation-C | exchangedData"]

- ["udp://141.24.32.192:5550", "measurement-B | localData", "gossip-A | externalData"]
- ["udp://141.24.32.192:5551", "measurement-C | localData", "gossip-A | externalData"]
```

Figure 6.27: YAML Description of Directed Fusion Graph for Resource Map Generation

Chapter 7

Conclusions and Future Work

The thesis is a research work on the decentralized data fusion and data harvesting framework for heterogeneous dynamic network system consists of various devices with resource constraints. The aim of this thesis is to optimize energy consumption by adopting in-network data fusion and data harvesting strategy. The contribution of thesis is a novel flexible architecture which integrates the functions of data fusion, data exchange and configuration automation as plug-ins to implement the framework. This results in a middle-ware platform, which support general purpose data-centric applications above the heterogeneous dynamic network systems. This chapter summarizes the main contributions of this thesis and provides some suggestions on future research directions in this area.

7.1 Contributions

Comparing with existing works, this framework tries to balance the generality of applications supported and the different design aims that different applications requires, with a flexible architecture.

In order to achieve the flexibility design aim, a general architecture is provided while the detailed data fusion and data exchange functions can be dynamically configured. A novel method to use directed fusion graph to model the logical structure of the distributed information fusion architecture is introduced. This directed fusion graph can accurately portray the interconnection of different data fusion components and the data exchange protocols. The directed fusion graph is then transformed into a format with markup language, so it connects both the design and the implementation of the system.

The directed fusion graph extracts and represents the three most important pillars which support the data centric applications on mobile networking and systems. Although these applications vary from each other at first glance, such as the use cases listed in this thesis, the three pillars set the general formats and patterns of such applications.

The graph can be regarded as a blueprint guiding the design of the distributed system, and also a configuration plan which can be directly transformed to be real system implementation. The graph can also be separated into local small configuration plan, used for setting up data exchange edges automatically among devices in the nearby area. The three constructing components, data fusion, data exchange and configuration of the distributed data fusion system, can all be considered and included in the directed data fusion graph.

In the field of data exchange protocols, this thesis targets energy-efficiency considering the resource constraints of the devices, and robustness as the dynamic environment might cause failures to the system. It exploits the trade-off relations between this two design aims, and propose a refined gossip strategy to reduce retransmission of redundant data. The thesis also discusses the performance of various gossip protocols under the different failure models, suggesting a design guideline to achieve different design aims for different applications. These results in this field can be integrated into the framework effortlessly. Evaluation shows reinforced gossip protocols while maintain reliability under different failure scenarios, offering flexibility for designers of distributed data fusion system to adopt based on different application requirements. Refined push-pull gossip protocols help save around 30% of the data processing time to reach the same data accuracy comparing with traditional gossip protocols. Furthermore, the refined push gossip protocol helps reduce 40% of the data message volumes transmitted within the system comparing with normal gossip protocols and 55% reduction of the data messages comparing with flooding protocol. These results show the advantage of adopting refined gossip protocols, in that they can significantly reduce energy consumption in the wireless communication consisting of resource limited devices, and further extend the lifetime of the data fusion system.

The configuration mechanism is another feature of this framework. Different from other research work which considers configuration as a post-design work separated from the main design of any middleware. This thesis considers the configuration part as another dimension of the framework, since the configuration plan decides the practical topology or fusion structure above the system. This can lead to new energy-efficiency design. At the middle-ware level, a supporting platform is provided and built upon various devices to dynamically compile and deploy the logic structure of directed fusion graph. The whole strategy in configuration sets up the foundation for the flexible architecture used for data fusion and data harvesting, and makes it easy to adapt to the dynamic environment.

The contributions in the above fields lead to a light-weight data fusion and data harvesting framework which can be deployed easily above wireless based, heterogeneous, dynamic network systems, even in extreme conditions, to handle disaster monitoring and carry out change detection tasks.

The framework can be used in general data-centric applications. The thesis provides several applications uses cases and scenarios to explain the usage of the framework and its specific features. These scenarios include a group of feature maps generation process (resource map generation in disaster scenarios), decentralized change detection, as well as the mobile query fusion framework. At the same time, evaluation results show the effectiveness of applying refined data exchange protocols, in these scenarios in achieving energy efficiency and fusion accuracy aims.

7.2 Future Directions

In general, the research on data processing middle ware for heterogeneous dynamic network systems is still of great space, though relative research on sensor networks, or distributed systems is relatively mature. In fact, more kinds of sensor nodes such as smart phone, RFID, camera sensors and even novel Google glass or Iwatch have put new challenges to this research field. Different capability of these nodes makes it possible to divide the tasks. By allocating energy consuming tasks above nodes with relatively high volume of battery, it is possible to avoid the partitioning problem of the network. Cooperation in data processing and data fusion is more energy efficient, with careful design on deploying certain fusion

components. Besides the problem caused by heterogeneity, there are several directions worth further researching. This section mainly summarizes on the following two directions:

7.2.1 Dynamic Topology of Sensor Network for Dynamic Environment

The thesis solves the problem in flexible configuration of the data fusion and data exchange components so that the framework can adapt to different application scenarios according to different design requirements.

To step further from current flexible design, it is possible to design a flexible configuration towards a dynamic environment. The general strategy could be, based on the results of change detection in dynamic environment, nodes can adjust their interconnection with their surrounding nodes. This requires intensive calculating of further optimized plan which targets one optimization aim (minimize energy consumption on energy limited node, minimize the usage of the communication bandwidth, etc). The resulting configuration and deployment plan can firstly be updated locally, and then broadcasted to neighborhood for all the nodes to update after compilation. This strategy benefits the network in that it can automatically react to environmental changes, and complete self-organization under a dynamic environment.

7.2.2 Contributed Algorithms for Optimization Energy Efficiency

In order to implement the self-adaption strategy mentioned above, it is necessary for the local node to be able to carry out computations on optimization towards some criteria based on local situation. Such local optimization algorithms play the role of connecting the sensing to the decision making on topology changes. The challenge comes from the incomplete information, which can deviate the results from global optimal value. Thus is a necessary and interesting direction worth putting more efforts into.

Bibliography

[1] Tarek Abdelzaher, B Blum, Qing Cao, Yong Chen, David Evans, Joshua George, Selvin George, Lin Gu, Tian He, Sudha Krishnamurthy, et al. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 582–589. IEEE, 2004.

[2] Artificial intelligence, 2001. http://en.wikipedia.org/wiki/Artificial-intelligence (last visited 2013/04/21).

[3] G Banon. Distinction between several subsets of fuzzy measures. *Fuzzy sets and systems*, 5(3):291–305, 1981.

[4] M. Bawa, H. Garcia-Molina, A. Gionis, and R.Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Stanford InfoLab, 2003.

[5] M. Bedworth and J. O'Brien. The omnibus model: a new model of data fusion? *Aerospace and Electronic Systems Magazine, IEEE*, 15(4):30 –36, apr 2000.

[6] Kenneth P Birman, Robbert Van Renesse, and Werner Vogels. Scalable data fusion using astrolabe. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 2, pages 1434–1441. IEEE, 2002.

[7] col J. A. Boyd. A Discourse on Winning and Losing, 1987.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. In *IEEE/ACM Trans. Netw.*, volume 14, pages 2508–2530, 2006.

[9] U. Bud and J.T. Lim. Distance Functions to Detect Changes in Data Streams. 2006.

[10] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. *mobile computing, IEEE transactions on*, 3(3):272–285, 2004.

[11] Laukik Vilas Chitnis. *Fault tolerance and scalability of data aggregation in sensor networks*. PhD thesis, University of Florida, 2008.

[12] Thomas Clouqueur, Veradej Phipatanasuphorn, Parameswaran Ramanathan, and Kewal K Saluja. Sensor deployment strategy for target detection. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 42–48. ACM, 2002.

[13] Coordinate transformations, 2009. http://kartoweb.itc.nl/geometrics/coordinate(last visited 2013/08/21).

[14] Mads Dam and Rolf Stadler. A generic protocol for network state aggregation. In *In Proc. Radiovetenskap och Kommunikation (RVK)*, 2005.

[15] B.V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24 –38, jan 1997.

[16] data fusion wikipedia homepage. http://en.wikipedia.org/wiki/Datafusion (last visited 2013/02/21).

[17] L de Souza, Ricardo Sangoi Padilha, and Christian Decker. Neural fault isolator for wireless sensor networks. In *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, pages 47–50. IEEE, 2008.

[18] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, August 1987. ACM Press.

[19] Santpal S Dhillon, Krishnendu Chakrabarty, and SS Iyengar. Sensor placement for grid coverage under imprecise detections. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 2, pages 1581–1587. IEEE, 2002.

[20] M. Dietzfelbinger. Gossiping and broadcasting versus computing functions in networks. In *Discrete Appl. Math.*, volume 137(2), pages 127–153, 2004.

[21] Digital Signal Processing, 2011. http://en.wikipedia.org/wiki/Digital-signal-processing (last visited 2013/04/21).

[22] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270. ACM, 1999.

[23] P. T. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, May 2004.

[24] Expert system, 2001. https://en.wikipedia.org/wiki/Expert-system (last visited 2013/04/21).

[25] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(3):16, 2009.

[26] Conny Franke, Marcel Karnstedt, Daniel Klan, Michael Gertz, Kai-Uwe Sattler, and Elena Chervakova. In-network detection of anomaly regions in sensor networks with obstacles. *Computer Science-Research and Development*, 24(3):153–170, 2009.

[27] Carl B Frankel and Mark D Bedworth. Control, estimation and abstraction in fusion architectures: Lessons from human information processing. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, volume 1, pages MOC5–3. IEEE, 2000.

[28] J.L. Gao. *Energy Efficient Routing for Wireless Sensor Networks*. PhD thesis, University of Carlifornia at Los Angeles, Los Angeles, USA, 2000.

[29] J. Gaonkar, S. Li and et al. Micro-Blog: Sharing and Querying Content Through Mobile Phones and Social Participation. In *MobiSys 2008*, 2008, June.

[30] L. Girod, Y. Mei, R. Newton, S. Rost, A. Thiagarajan, H. Balakrishnan, and S. Madden. The case for a signal-oriented data stream management system. In *In CIDR*, 2007.

[31] L. Girod, Y. Mei, R. Newton, S. Rost, A. Thiagarajan, H. Balakrishnan, and S. Madden. Xstream: A signal-oriented data stream management system. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 1180–1189. IEEE, 2008.

[32] Indranil Gupta, Denis Riordan, and Srinivas Sampalli. Cluster-head election using fuzzy logic for wireless sensor networks. In *Communication Networks and Services Research Conference, 2005. Proceedings of the 3rd Annual*, pages 255–260. IEEE, 2005.

[33] Malka N Halgamuge, Siddeswara Mayura Guru, and Andrew Jennings. Energy efficient cluster formation in wireless sensor networks. In *Telecommunications, 2003. ICT 2003. 10th International Conference on*, volume 2, pages 1571–1576. IEEE, 2003.

[34] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[35] Chen Hongyang, Deng Ping, Xu Yongjun, and Li Xiaowei. A robust location algorithm with biased extended kalman filtering of tdoa data for wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, volume 2, pages 883–886. IEEE, 2005.

[36] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[37] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00*, pages 56–67, New York, NY, USA, 2000. ACM.

[38] S. Iyengar and H. Qi. Introduction to special issue on "distributed sensor networks for real-time systems with adaptive configuration". In *J. Franklin Inst.*, volume 338, pages 651–653, 2001.

[39] Understanding japan's nuclear crisis, 2011. http://www.wired.com/wiredscience/2011/03/japan-nuclear-crisis/ (last visited 2012/12/21).

[40] Mark Jelasity. Peer-to-peer systems and gossip protocols. Technical report, SASO tutorials, 2007.

[41] Guang Jin and Silvia Nittel. Ned: An efficient noise-tolerant event and event boundary detection algorithm in wireless sensor networks. In *Mobile Data Management, 2006. MDM 2006. 7th International Conference on*, pages 153–153. IEEE, 2006.

[42] NB Jones and J.D.M.K. Watson. *Digital signal processing: principles, devices, and applications*. Peter Peregrinus Ltd, 1990.

[43] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[44] K.Birman. The promise, and limitations, of gossip protocols. In *SIGOPS Oper.Syst.Rev.*, volume 41(5), pages 8–13, 2007.

[45] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[46] Min Y Kim. Configuration and management of wireless sensor networks. Technical report, DTIC Document, 2005.

[47] Mieczyslaw M Kokar, Mark D Bedworth, and Carl B Frankel. Reference model for data fusion systems. In *AeroSense 2000*, pages 191–202. International Society for Optics and Photonics, 2000.

[48] Lim L. and Bhattacharjee B. Optimizing access across multiple hierarchies in data warehouses. In *HICSS'11*, pages 1–10, 2011.

[49] Leach, 2011. http://en.wikipedia.org/wiki/Low-Energy-Adaptive-Clustering-Hierarchy (last visited 2013/04/21).

[50] Qun Li and Daniela Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 44–55. ACM, 2000.

[51] Shuoqi Li, Sang H Son, and John A Stankovic. Event detection services using data service middleware in distributed sensor networks. In *Information Processing in Sensor Networks*, pages 502–517. Springer, 2003.

[52] Teng Li, Anthony Ekpenyong, and Yih-Fang Huang. Source localization and tracking using distributed asynchronous sensors. *Signal Processing, IEEE Transactions on*, 54(10):3991–4003, 2006.

[53] Ting Liu and Margaret Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. In *ACM SIGPLAN Notices*, volume 38, pages 107–118. ACM, 2003.

[54] X. Liu, X. Wu, H. Wang, R. Zhang, J. Bailey, and K. Ramamohanarao. Mining distribution change in stock order streams. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 105–108. IEEE, 2010.

[55] J. Lu, H. Yang and et al. The Jigsaw Continuous Sensing Engine for Mobile phone Applications. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*, 2010, November.

[56] RC Luo and MG Kay. Data fusion and sensor integration: State-of-the-art 1990s. *Data Fusion in Robotics and Machine Intelligence*, pages 7–135, 1992.

[57] Ren C Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *Sensors Journal, IEEE*, 2(2):107–119, 2002.

[58] S. Madden, R. Newton, L. Girod, M. Craig, S. Madden, and G. Morrisett. WaveScript: A Case-Study in Applying a Distributed Stream-Processing Language. 2008.

[59] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[60] Pedro José Marrón, Andreas Lachenmann, Daniel Minder, Jorg Hahner, Robert Sauter, and Kurt Rothermel. Tinycubus: a flexible and adaptive framework sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 278–289. IEEE.

[61] Dave McDaniel. An information fusion framework for data integration. In *Proceedings of the 13th Software Technology Conference*, 2001.

[62] M.S.Artigas, P.G. Lopez, and A.F.Gomez-Skarmeta. DECA: a hierarchical framework for decentralized aggregation in DHTs. In *In: State, R., van der Meer, S., O'Sullivan, D., Pfeifer, T.(eds.) DSOM*, volume 4269, pages 246–257, Heidelberg, 2006. Springer.

[63] Eduardo F. Nakamura, Antonio A. F. Loureiro, and Alejandro C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3), September 2007.

[64] Eduardo Freire Nakamura, Carlos Mauricio S Figueiredo, and Antonio Alfredo F Loureiro. Information fusion for data dissemination in self-organizing wireless sensor networks. In *Networking-ICN 2005*, pages 585–593. Springer, 2005.

[65] NIST. Kolmogorov-smirnov goodness-of-fit test. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm.

[66] Robert Nowak, Urbashi Mitra, and Rebecca Willett. Estimating inhomogeneous fields using wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 22(6):999–1006, 2004.

[67] Reza Olfati-Saber. Distributed kalman filter with embedded consensus filters. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 8179–8184. IEEE, 2005.

[68] Daniel Pagac, Eduardo M Nebot, and Hugh Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *Robotics and Automation, IEEE Transactions on*, 14(4):623–629, 1998.

[69] Pattern recognition, 2011. http://en.wikipedia.org/wiki/Pattern-recognition (last visited 2013/04/21).

[70] Q. Qang, H. Hassanein, and K. Xu. A Practical Perspective on Wireless Sensor Networks. In M. Ilyas and I. Mahgoub, editors, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, chapter 9. CRC Press LLC., 2004.

[71] Guillaume Doyen Rafik Makhloufi, Grégory Bonnet and Dominique Gaïti. Decentralized aggregation protocols in peer-to-peer networks: a survey. In *4th IEEE International Workshop on Modelling Autonomic Communications*, pages 111–116, Venice, Italy, 2009.

[72] R.V. Renesse, K.P. Birman, and W. Vogels. Astrolable: a robust and scalable technology for distributed system monitoring management, and data mining. In *ACM Trans. Comput. Syst.*, volume 21(2), pages 164–206, 2003.

[73] M. Rossenberg. Kismac, 2011. http://www.binaervariznz.de/projekte/programmieren/kismac/kartenspiel/ (last visited 2012/12/21).

[74] B. Roychoudhury, A. Falchuk and A. Misra. MediAlly: A Provenance Aware Remote Health Monitoring Middleware. In *8th IEEE International Conference on Pervasive Computing and Communications (PerCom))*, 2010, March.

[75] Andreas Savvides, Heemin Park, and Mani B Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 112–121. ACM, 2002.

[76] Chien-Chung Shen, Chavalit Srisathapornphat, and Chaiporn Jaikaeo. Sensor information networking architecture and applications. *Personal communications, IEEE*, 8(4):52–59, 2001.

[77] A.N. Shulsky and G.J. Schmitt. *Silent Warfare: Understanding the World of Intelligence*. Intelligence & national security library. Brassey's, 2002.

[78] A. Singh. Review Article Digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, 10(6):989–1003, 1989.

[79] Aarti Singh, Robert Nowak, and Parmesh Ramanathan. Active learning for adaptive mobile sensing networks. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 60–68. ACM, 2006.

[80] N. Sklavos and K. Touliou. A System-Level Analysis of Power Consumption and Optimizations in 3G Mobile Devices. In *proceedings of the 1st International Conference on New Technologies, Mobility and Security (NTMS'07)*, pages 225–235, 2007.

[81] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. Protocols for self-organization of a wireless sensor network. *Personal Communications, IEEE*, 7(5):16–27, 2000.

[82] Katayoun Sohrabi and Gregory J Pottie. Performance of a novel self-organization protocol for wireless ad-hoc sensor networks. In *Vehicular Technology Conference, 1999. VTC 1999-Fall. IEEE VTS 50th*, volume 2, pages 1222–1226. IEEE, 1999.

[83] Eduardo Souto, Germano Guimarães, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, Carlos Ferraz, and Judith Kelner. Mires: a publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing*, 10(1):37–44, 2006.

[84] Demetri P Spanos, Reza Olfati-Saber, and Richard M Murray. Approximate distributed kalman filtering in sensor networks with quantifiable performance. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 18. IEEE Press, 2005.

[85] Wei Wang Vikram Srinivasan and Kee-Chaing Chua. Trade-offs between mobility and density for coverage in wireless sensor networks. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 39–50. ACM, 2007.

[86] F.and etal Tennina, S. Graziosi. Distributed and cooperative localization algorithms for wsns in gps-less environments. *ATTI of Italian Navigation Institute*, 2008.

[87] Dang-Hoan Tran, Jin Yang, and K.-U. Sattler. Decentralized change detection in wireless sensor network using dft-based synopsis. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 226–235, 2011.

[88] O. Waldhorst. *Design and Quantitative Analysis of Protocols for Epidemic Information Dissemination in Mobile Ad Hoc Networks*. PhD thesis, November 2005.

[89] Miao-Miao Wang, Jian-Nong Cao, Jing Li, and Sajal K Dasi. Middleware for wireless sensor networks: A survey. *Journal of computer science and technology*, 23(3):305–326, 2008.

[90] Kamin Whitehouse, Cory Sharp, Eric Brewer, and David Culler. Hood: a neighborhood abstraction for sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 99–110. ACM, 2004.

[91] Rebecca Willett, Aline Martin, and Robert Nowak. Backcasting: adaptive sampling for sensor networks. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 124–133. IEEE, 2004.

[92] Andreas Willig, Rahul Shah, Jan Rabaey, and Adam Wolisz. Altruists in the picoradio sensor network. In *Factory Communication Systems, 2002. 4th IEEE International Workshop on*, pages 175–184. IEEE, 2002.

[93] Yaml, 2013. http://en.wikipedia.org/wiki/YAML(last visited 2013/1/21).

[94] Chin-Lung Yang, Saurabh Bagchi, and William J Chappell. Location tracking with directional antennas in wireless sensor networks. In *Microwave Symposium Digest, 2005 IEEE MTT-S International*, pages 4–pp. IEEE, 2005.

[95] Jin Yang, Tianli Mo, Lipyeow Lim, K.-U. Sattler, and A. Misra. Energy-efficient collaborative query processing framework for mobile sensing services. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 147–156, 2013.

[96] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3):9–18, 2002.

[97] L Yiyao, YV Venkatesh, and CC Ko. A knowledge-based neural network for fusing edge maps of multi-sensor images. *Information Fusion*, 2(2):121–133, 2001.

[98] Mariam Yusuf and Tarique Haider. Energy-aware fuzzy routing for wireless sensor networks. In *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on*, pages 63–69. IEEE, 2005.

[99] ZeroMQ homepage, 2012. http://www.zeromq.org/ (last visited 2012/06/21).

[100] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76, 1997.

[101] Yonggang Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scan for monitoring sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 356–362. IEEE, 2002.

[102] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment, 2002.

[103] J. Zou, A. Gilbert, M. Strauss, and I. Daubechies. Theoretical and experimental analysis of a randomized algorithm for sparse Fourier transform analysis. *Journal of Computational physics*, 211(2):572–595, 2006.

Appendix A

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalte der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch angesehen wird und den erfolglosen Abbruch des Promotionsverfahrens zu Folge hat.

Jin Yang
Ilmenau, August 21, 2015