Application-level Quality of Service and Information Quality provisioning in Sensor Networks

Andrei Tolstikov
MSc (Moscow Institute of Physics and Technology), 1994

A Thesis submitted for the degree of Doctor of Philosophy
Department of Electrical and Computer Engineering
National University of Singapore
April 2008

Contents

Summary

Nowadays distributed computing environments are becoming increasingly complex and it is becoming increasingly difficult to provide Quality of Service (QoS) guarantees to applications in such environments. The straightforward implementation of techniques such as connection admission control, differentiated services and integrated services, that are used to provide QoS guarantees in networks and simple distributed applications such as unicast or multicast streaming applications, may not be able to address the requirements of the complex systems. This thesis considers application-level quality of service in loosely coupled distributed systems, of which the sensor networks are an example. For sensor networks, the particular aspect of application quality of service called *Information Quality* is explored in detail. Three techniques are proposed, each of them represents one of the basic mechanisms of QoS management, but deeply modified to suit the particular application domain.

The first is the measurement-based admission control procedure for a sensor network query. The significant difference from the network connection admission control is in two facts. First, the structure of a sensor network query is taken into account and the probabilistic performance of the whole query is used as an admission control parameter. Second, the probability distribution for a query performance is obtained using statistical parameters measured locally on sensor network nodes thus eliminating the need for complex sensor network control.

The second technique is a resource optimization algorithm formulated to guarantee the Information Quality obtained by a sensor network data-fusion application. The algorithm not only takes into account the states of the application and of the resources, but also the state of the phenomena observed by the application. The Dynamic Bayesian Network (DBN) model is used to derive the dependency between the resources used and information quality obtained. The novelty of this approach lies in three aspects. First, it brings in the general notion of phenomena into picture, going beyond particular types phenomena such as target localization and tracking. This notion allows us to account for effects of the different phenomena state onto the information obtained. Second, it allows dynamic phenomena tracking in a resource efficient manner due to the use of the DBN model. Third, it integrates into the sensor network framework, taking into account information loss and resource constraints.

The third technique explored in this thesis is conceptually a form of a *leaky bucket* regulator, but implemented in the distributed fashion for a complex

cyclic application in a loosely coupled environment, so that no additional communication is required for coordination of execution in different administrative domains, and yet the regulation is achieved without unnecessary slowing down of the application.

The general approach used in this work is based on modelling of an application and consists of three stages. The first is to analyze an application. The second is to identify the specifics of the environment which may prevent the application from obtaining the required level of service. The third is to choose the model of application and the method of using this model which can overcome the environment specifics.

KEYWORDS: sensor networks, information quality, application QoS, sensor selection, dynamic Bayesian network, Pareto distribution, Petri net.

List of Figures

Chapter 1

Introduction

The technological advancement of electronic components is making cost of the computing devices lower and capabilities higher. The variety of the types of the computer systems is becoming broader as well, and this is especially true for distributed systems. During recent years, a new class of distributed system has emerged, which can be called loosely coupled distributed system. Not only the parts of such system do not have central control, which is common to all distributed systems, but but they may not even have a sufficient level of process coordination due to different administrative boundaries, low speeds of communication diminishing ability of components to interact or high delay in such interaction compared to the typical time duration of processes happening in them. One example of such systems are sensor networks.

With further development of such loosely coupled systems it is expected that increasingly different applications will be using these systems simultaneously. In this situation, the question of the quality of service for these applications will become important. This thesis addresses some of the issues of provisioning of application-level quality of service either for general loosely coupled systems or for sensor networks in particular.

At first we will give a general introduction of the concept of quality of service and describe in more details the class of systems we are addressing, namely, general loosely coupled systems and sensor networks. This introduction is general in the sense that we are not going to address the specific limitation of particular QoS mechanisms applied to this class of systems, but rather generally describe the concept and the idea behind them. A more detailed discussion will be presented in each of the chapters presenting the proposed methods.

The introduction covers the concept of the Quality of Service with the emphasis on the network QoS in Section 1.1, provisioning of QoS for applications in Section 1.2, overview of sensor networks in Section 1.3 and overview

of loosely coupled distributed systems in Section 1.4. In Section

1.1 Overview of Quality of Service

The term *Quality of Service* in the context of a computer system refers to the ability of the underlying infrastructure to provide assurance that certain performance parameters of the application using resources of the infrastructure are satisfied. In particular, the case of shared use of an infrastructure is considered since provisioning of performance guarantees in the case of exclusive use of resource is trivial.

Historically, the most common type of such shared infrastructure have been computer networks. Because of this reason the most of the original research in the area of Quality of Service was done in the area of networks. The performance parameters considered are packet delay, packet delay variation and packet loss probability.

The infrastructure performing computational tasks can be seen as a collection of interconnected *servers* and streams of *tasks* using these servers. The servers can be connected directly or through other servers. For example, consider s computer and a network link vs two computers connected by a network link. The tasks may also be dependent on each other (such as multihop communication) or independent. Each server has some performance characteristics and the stream of tasks arriving at this server is characterized by resource requirements to process or hold the task.

Conceptually, when we talk about Quality of Service, we talk about a set of models and methods which allow us to predict the performance characteristics of tasks being processed by a set of servers and to modify the behavior of the system so that the above performance characteristics would be at least on some minimally required level. In most cases, not all, but only a subset of the tasks processed by a system are supposed to get a guaranteed service.

The performance guarantees can be given only when both the server performance as well as the flow of incoming tasks are controlled. Therefore the research in the QoS mainly deals with these two aspects. The server performance characteristics obtained by a subset of the packets are achieved using different service disciplines and, respectively, different queueing algorithms [Zha95], [NJZ99]. The control over the flow of incoming tasks can be done in different ways as follows:

1. By changing the packets arrival process before the queue, by a technique such as *shaping* [GGPR96], or in combination with a queueing algorithm as in the case of [ZF94].

2. By using the knowledge about applications (network flows in the context of network) and performing *admission control* which essentially makes a decision whether another flow can be allowed based on the computed worst-case performance characteristics such as [LWF96].

3. By using the application specifics to provide feedback from the system to the application. For example, in [FJ93] the property of the TCP protocol is used to regulate the packet load.

The important fact to note is that in all cases there is a model of the service performance as well as a model of the service demand. These models are used to obtain the performance characteristics of the service obtained by tasks. In the case of more complex applications, the situation is similar, but the performance metrics are different.

As we mentioned, the common performance metrics for the network QoS are packet delay and delay variation and packet loss. Although these metrics are adequate for the network applications such as file transfer of streaming media, more general applications often require satisfaction of performance metrics which are more closely related to the application's characteristics and runtime behavior.

1.2 Application-level Quality of Service

Strictly speaking, it is possible to create a system which would guarantee the performance parameters of an application expressed in application-specific terms and implemented directly into the system. Examples of this kind of systems are hard real-time systems such as airplane flight control. However, in the case of most computer systems it is not reasonable to expect such a high level of integration between an application and its infrastructure.

For this reason, the approach that QoS parameters of the system are defined separately from the application-level parameters was adopted. It is assumed that the mapping between the two sets of parameters is separately established. This mapping requires the application to be modelled in terms of the tasks components using resources and obtaining specific QoS. The general framework for application modelling is presented in [CSS97]. In another work, [GN02], the application is represented as a set of components which transform the notion of the QoS, and the end-to-end application QoS is modelled as a result of a chain of transformations. The model of application is used even in the case of a single server, such as a web-server [ASB02]. One very important implication of such a mapping is the ability to consider the QoS metrics which cannot exist in the system comprising of only application

and resources. For example, the work [GT98] considers the impact of the network QoS on the user perception of the video. In this case, the user, a human, is outside of the system. But the model of the perception allows us to make some guarantees on the quality of the video as seen by a human.

In a similar way, we argue in the Chapter 2, that in sensor networks there exist an important part which is outside of the system, namely, the object or phenomenon being monitored.

Therefore, when we need to provide application-level quality of service, we have to use the model of the application. However, a very general model may not be of much use, since it may not give us enough details of the QoS metrics and requirements. We need to consider particular classes of applications in particular environments, while attempting to keep them as general as possible within the bounds of the environments.

Below, we are describing the two environments which are considered in this work, namely sensor networks and general loosely coupled system. We will give general description and identify the specifics of these environments which will be useful in later chapters, where specifics of the proposed techniques are discussed.

1.3 Overview of Sensor Networks

The decreasing cost of electronic components made it possible to install simple processors or micro-controllers into many devices used by people in everyday life, such as kitchen appliances or car controls. The fact is that in most cases people are not even aware of the fact that they are using an intelligent device. The next stage of such a development is installation of intelligent devices in the environment so that they stay there, collect information and use this information to help people to perform some tasks.

The hardware behind such intelligent infrastructures are sensor nodes, which are battery-powered wireless computer platforms having specific sensors connected to them to collect the required information. The typical size of such a node is just slightly bigger than the size of its the power source, consisting, for example of two AA size batteries. The systems comprising of large number of such nodes may be able to perform complex tasks by leveraging the total computing power of all the nodes. The example tasks include bird habitat monitoring [MPS+02], health monitoring of complex structures [XRC+04] or helping in taking care of the elderly people in home or hospital environment [BDQ+05].

Although the existence of such sensor networks offer new opportunities, they also represent a significant challenge for their designers and application

developers. The main limiting factor in the design of a sensor node is the power source. To overcome it, different energy saving can be implemented. For example, most sensor nodes use the low-power slow-speed radio and use different modes of node operation with different power consumption. In the latter case, the node may spend most of the time in the power-saving "sleep" mode and only wake up to perform sensing or communication. Such sleep - wake-up duty cycle makes communication between nodes more difficult compared to the common wireless nodes and even specialized MAC protocols are proposed which are custom-tailored to the sensor network environment [PHC04],[YHE02a]. Therefore the communication in sensor network may be not only be costly and have long average delay, but in some cases may not be possible in arbitrary time moment, thus limiting the possibility of application-level control.

Since the purpose of sensor networks is specific, they are commonly organized by a specific software, for example the data collection systems such as TinyDB [MFHH05] or collaborative target tracking systems [ZLL+03]. These types of software create the applications running on the sensor networks. The positive side of these systems is the fact that they make a limited scope of types of applications. Therefore in many cases we may limit the analysis to the few application examples. For example, the TinyDB creates tree-shaped information collection queries.

Important feature of the sensor networks in the fact that they collect the information about some phenomena or environment. Therefore the state of the environment affects the type of information collected. For example, in [DGM+04] and [CHZ02], the fact that there is a model of the objects monitored is used in making resource allocation decisions.

1.4 Overview of loosely coupled distributed systems

Sensor networks can be considered to be an example of a more general class of distributed systems, which we may call *loosely coupled distributed systems*. For certain classes of applications, the specifics of sensor networks such as mostly wireless communications, information-centric data and tight energy constraints are not so important, and therefore it does make sense to formulate a problem of application-level QoS for these applications in the more general context of loosely coupled distributed systems.

As the term suggests, loosely coupled distributed systems are characterized by a low degree of coupling between different components of the

system. Usually it happens because of difficulty in communication between components, which, in turn, may be due to different communication media or protocols, different administrative domains or specific schedule of device communication. This difficulty in communication may lead to the situation that the typical time of operation on a single device is shorter that the typical time required for coordination of task execution on different devices. In this case, the tight coordination of operations on different servers or devices would impair the performance of the whole system, and therefore decisions on how to process the tasks are done on the local level.

Another difference of loosely coupled systems from traditional distributed systems is that the types and set of both resources and applications using the resources are not fixed. The implication of this is that sometimes there is no direct connection between the type of task and the type of resource the task is supposed to be executed. The mapping of tasks to resources is done at the runtime and sometimes may be only be satisfied up to a certain degree. Moreover, the bigger the pool of resources, the larger may be the set of applications using these resources.

In addition to sensor networks, another example of such a loosely coupled system is computational Grid [FK99].

The list of the most several important features of loosely coupled distributed systems is

- It is dynamic. Resources and applications are added and removed from the system unpredictably.
- It is highly heterogeneous. It consists of many types of systems so that it may not be even possible to enlist and characterize all of them precisely.
- It is complex in structure. It may consist of many components and interaction between them may be too complex to trace.
- Has limited coordination between resource subsystems executing different tasks.

1.5 Motivation and Contribution

The traditional QoS methods for applications described in the Sections 1.2 are not adequate anymore for complex systems such as sensor networks. The main problem for this is that there is a multitude of resources and applications available in such systems, as well as the fact that QoS parameters

of applications are very different from resource QoS parameters. This calls for deeper understanding of the applications at hands and specifications of how the multitude of different resources used by an application can translate into a guarantee or at least assurance of specific application-level QoS metric.

The aim of this thesis is to:

- Understand the essential characteristics of certain classes of applications typical for sensor network or, more general, for loosely coupled distributed systems
- For each application class, propose a model of application which allows expressing of the allocation-level QoS metric in terms related to the resource level
- Propose methods of using above models to provide the guarantees or assurance for these QoS metrics in the specific environment.

Namely, there are three types of applications are considered:

1. Tree-shaped sensor data collection query, collecting similar type of information from a set of wireless sensor nodes. The query should provide Information Quality oriented metrics or support provision of such metrics by the upper-level application. The solution was obtained by deriving approximation for the distribution of a delay for the query data to be collected, aggregated and delivered to the consumer and providing examples of how assignment of loss bounds on each node in the query affects information quality metrics such as completeness or coverage.
2. General phenomena-tracking application which uses shared pool of resources and provide guarantees on the quality of collective information. The solution involves the use of Dynamic Bayesian Network model and suggests how information quality metrics such as confidence or coherence can be addressed for such model, as well as suggests a way of handling the losses of information in the network.
3. General cyclic computational application, using a variety of distributed resources aiming to guarantee that each computation cycle is completed before its deadline. The suggested technique represent a distributed regulator, which uses Timed Petri Net model to find the places and

1.6 Conclusion

We introduced the basic concept of the Quality of Service. The important note is the fact that to provide application-level QoS we need to have the model of the application, and the application specifics has to be bound to the specifics of the environment the application run in. In the following chapter, we consider in depth the application-level notion of QoS important to the sensor network environment, namely, Information Quality. In chapters 3 and 4 we analyze specific application types to propose methods to ensure the provision of the Information Quality.

Chapter 2

Quality of Information

The main goal of operation of sensor networks is collection of information about events and phenomena happening in the area where the sensors are deployed. Therefore, in deciding how the application-level quality of service can be provided for sensor networks, it is reasonable to begin with considering how the information collection is affected by the sensor network operation.

At first, we need to define the criteria of how well the information collected suits the application requirements. That is, we need to define the Quality of Information (IQ) parameters and then relate them to the operation of the sensor network and to the algorithms managing the access and use of resources. In this chapter we are presenting an overview of the Information Quality. Then follows the important contribution of this chapter, the framework for defining IQ metrics at the intersection of quality losses due to *acquisition* and *delivery* on one side and *completeness* and *uncertainty* on the other. We also describe our approach in managing IQ in the sensor network environment.

2.1 Overview of the Quality of Information

The term Information Quality is widely used in the community working with information systems. However, there is no strict definition of the term available and its meaning can be rather different depending on the nature of information. In [WS96] the taxonomy of the possible IQ definitions is given, which includes almost 200 different terms. This list includes common information descriptions such as *age* or *accuracy* as well as rarely used descriptions such as *purpose* or *conciseness*. For our purpose, we need to limit the number of IQ descriptions to those relevant for sensor networks.

Examples of a narrower set of IQ parameters arise in database informa-

tion systems [NR00] or in military battlefield information collection [PSB04]. [NR00] is particularly useful for our case because it introduces different levels of the information quality - *subject*, *process* and *object*. The subject level IQ includes quality parameters of information available to the end user, the process level includes parameters due to particular process of obtaining the information and object level includes parameters of information in the form as it is stored in the database. However, the model of the database is not directly applicable to the case of sensor networks. In databases, the information is stored somewhere and the problem of handing information translates to a problem of searching and fetching the necessary information. In the case when the information delivered is of unsatisfactory quality, the operation may be repeated. In sensor networks, however, the information is not stored, and the repeat operation may fetch different information just because the monitored environment has changed. That is, the *object* and *process* levels of IQ are tightly bound in the sensor networks. Therefore we are going to distinguish only two layers of information for the case of sensor networks

1. High-level collective information, which is combined information obtained from fusion of, in general, heterogeneous sensor data. This is equivalent to the subject level of the [NR00] classification. Further, we will be using the term *high level information* when we talk about this level of information.

2. Low-level information is usually delivered by a sensor network from homogeneous data sources. The IQ parameters for this type of information have to be assessed as they are being passed through the network. This is equivalent to the combined *object* and *process* levels of the [NR00] classification. We will be using the term *data level information* when we talk about this level of information.

Below we analyze the sensor network information acquisition in order to arrive at IQ metrics which are important. We are going to choose from those IQ parameters presented in the above papers.

2.2 Quality of Information metrics in the sensor networks

We base our approach in identifying the IQ metrics on the following premise: the Information Quality is the description of imperfection in the information, and quantitative information metrics therefore should reflect specific details

of the information imperfection. In [BHA+01], the possible defects of information named are ambiguity, uncertainty, imprecision, incompleteness and inconsistency.

For the metrics in the sensor network environment, when the values are usually represented by some statistical distribution, the uncertainty and imprecision are described by the same distribution. On the other hand, the defects of the ambiguity and inconsistency are handled either by the consumer of the sensor network information or by the information fusion algorithm, the latter case affecting the value distribution. Therefore we propose considering two basic defects: incompleteness and uncertainty.

Below we are going to formally define the information metrics. For this, we are going to use the following notations

Physical state: (actual state) - tuple $\mathbf{X} = (x_1, x_2, x_3, ..., x_n)$

Estimated state: tuple $\mathbf{Z} = (z_1, z_2, z_3, ..., z_n)$

Measurement: - tuple $\mathbf{V} = (v_1, v_2, v_3, ..., v_k)$[1]

Then we can define the metrics formally as follows:

Information completeness is defined as a relationship between the set of physical values in the environment $\mathbf{X}$ and set of estimated state $\mathbf{Z}$, indicating whether we are able to estimate a particular value x_i. This is general relationship because different values may be of different importance. *Information uncertainty* is defined only for variables we are able to estimate. It is given by the probability $U_i = P(x_i|\mathbf{Z})$

In addition to this basic classification we define the metrics according to the process due to which the information is affected, that is *acquisition* or *delivery* and according to the level of the information as defined in the previous section - *high level* or *data level*.

2.2.1 Acquisition and Completeness

This metric describes how many values we may be missing to capture for some reason. For example, we can miss recording events in the statio-temporal domain either in space or in time. Missing event in time can happen when the sampling rate is so low that some events can happen between consecutive samplings. In this case we need to define utility of sampling rate.

[1]Strictly speaking, the number of measured values k is different from the number of estimated variables n. In most cases, $n \leq k$, because measurements are usually combined, for example, by averaging. However, sometimes one measurement can be used to estimate more than one variable, for example battery voltage may also give an estimation of the ambient temperature [DGM+04]

$SR_{a-c} = utility(\frac{InterEventTime}{SamplingPeriod})$,

where $utility(x)$ is some function equal to 1 for $x \leq 1$ and monotonically decreasing to 0 for $x > 1$. $EventTime$ is equal to such a sampling period when we are guaranteed that we do not miss any important events.

Coverage C_{a-c} is the absolute coverage of the territory of interest by the sensor modalities

$C_{a-c} = \frac{CoveredArea}{TotalArea}$

Another possible metric is information coherence, which characterizes discrepancy between the actual phenomena state and its representation in the information collection system. In particular we may interested in the delay between the moment the phenomena state changes to the moment this change is reflected at the information consumer end, which we call *information coherence delay*. One of the important reasons of this delay is incomplete sensor information due to sensors not being activated at the time of change.

The information coherence delay can be described as time interval τ,

$\tau = min(t_B - t_A) : \ X(t_A) = Z(t_A) = A, X(t_B) = Z(t_B) = B$ in the vicinity of the moment when system state changes from A to B. Here $X(t)$ is the actual value of a variable X at time t and Z is the measured value of X which is available to the consumer.

2.2.2 Acquisition and Uncertainty

There are three major reasons for information uncertainty in acquisition.

First, during the time between two samplings the environment may change. We need to characterize this change, and this can be done through introducing the probability of change in the environment by certain value. The shorter the sampling rate, the smaller is the possible change, however, the dependency may be non-linear. We express the metric of losses due to sampling rate as a probability of the value difference exceeds certain value ε given an estimation $\mathbf{Z}$ and sampling rate.

$SR_{a-u} = P(\delta x < \epsilon | \mathbf{Z}, SamplingRate)$

Second, the measurement itself is not perfect, therefore the measured values are not equal to their real values. Note, that since most probably the measurement of final values of interest is indirect, the physical values are different from $\mathbf{X}$. The measurement uncertainty can be expressed as a conditional probability distribution of physical values w given the measurement $\mathbf{V}$.

$MU_{a-u} = P(w|\mathbf{V})$

From measured values final values are derived through information fusion, and we denote the uncertainty of the fused result as a conditional probability of value of interest x given the estimation $\mathbf{Z}$.

$FU_{a-u} = P(x|\mathbf{Z})$

If the final answer of a system is only one most likely state x_{max}, then the above probability $P(x_{max}|\mathbf{Z})$ becomes a value of *confidence* that the current state is x_{max}.

The information fusion uncertainty depends on the algorithm used for fusion or estimation.

2.2.3 Delivery and Completeness

Completeness losses in delivery occur when data is lost along the way and because of this we become unable to estimate certain variables of interest. There is a rather fine line between losses in space and time here. Loss in space may occur if several readings are lost from a particular area in the network. Loss in time may occur when not enough data is delivered to a consumer for some time and events are missed. A possible metric in this case may be the ratio of time when enough data was delivered for event detection to the total observation time. In particular, this ratio can be represented as

$DC = 1 - \frac{TimeLost}{T}$

where $TimeLost$ is the total accumulated time when data lost consecutively for the period $EventTime$, making possible that we missed an event.

This type of metric on the data level can be called *data completeness* and is quite similar to the notion of completeness used in the database systems for the raw data.

2.2.4 Delivery and Uncertainty

There are two main reasons for uncertainty due to delivery. First, we need to characterize the impact on the uncertainty of the result because of the losses of certain measurements. Since in this case we need to highlight the difference between distributions, we may use the entropy difference as a metric of this uncertainty.

$H_{diff} = H(\mathbf{V}) - H(\mathbf{Z})$

The second reason for uncertainty is lack of *data coherence*, and the metric has to account for the difference between the measured value at particular time and value as seen at the same time by consumer.

Here, we assume that consumer has some kind of predictive function $pz(t)$ which estimates the value at the consumer since the available reading. In the simple case the function can be equal to the last measurement. Uncertainty due to data coherence is therefore characterized by the conditional probability that the difference exceeds certain value ε, given that the last observed state is $\mathbf{Z}$.

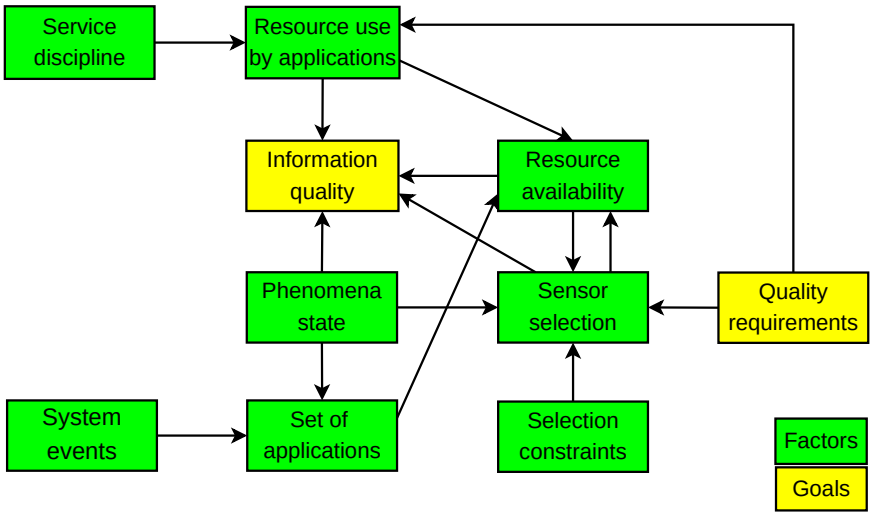

Figure 2.1: Diagram describing the dependency between factors affecting the quality of information delivered to a consumer.

$$DC_{d-u} = P(|z(t) - pz(t)| < \varepsilon \mid \mathbf{Z})$$

2.3 Information quality dependency

To tackle the problem of information quality assurance, we first need to understand the factors on which the IQ of delivered information depends. Figure 2.1 depicts the dependencies between different mechanisms existing in sensor networks. In drawing these dependencies we assume a general sensor network model with only one important assumption - that there is a redundancy in the number of sensors and we have a choice of sensor to select among all available to do the phenomena monitoring.

There are four major factors affecting the quality of information delivered to a sensor network consumer

1. **State of physical phenomena** under consideration. This may include both the changes in the environment that affect the measured values at sensors as well as the changes of sensor condition themselves such as specifics of measuring modalities.

2. **Sensor selection**, that is, the choice of sensors participating in data acquisition.

3. **Resources available** on the sensor nodes for data acquisition, processing or transmission.

4. **Resource use by applications**. In this category are included the structure of an application, operator deployment on sensor nodes and throughput characteristics of an application.

Omitted from the list are the factors which depend almost entirely on a particular network configuration and which we cannot change such as topology of sensor deployment.

There is a dependency between three of these factors. When the state of phenomena changes, we would probably need to update the set of sensors participating in the measurement because the current set may no longer give satisfactory quality of information. The change in the set of participating sensors leads to a change in resources available on individual sensors. This change in resource availability, in turn, may affect the quality of information delivered such that it is no longer satisfactory. Since we consider dynamic environment, we have to assume that all of these three factors are dynamic and therefore we need to consider their overall effect on the IQ. Most important, we have to include phenomena state awareness into the framework.

2.4 Conclusion

The definition of the IQ metrics allows us to build systems which use the above definition to translate the underlying resources used into guarantees on defined IQ metrics. It may not be possible to address all the possible IQ metrics within one framework, since different levels of information have to be considered.

In the next two chapters, we will introduce two frameworks which address certain IQ metrics, the first one focused on the data level information, and the second one on the high-level information.

Chapter 3

Data-level query admission-control

A significant number of sensor networks simply collect data without processing its content to extract meaning from the underlying semantic. Aggregation functions in this case are simple and easily computed, and sensors are selected for the duration of the entire process of the data acquisition. Below, we consider such a scenario to provide the IQ guarantees for distributed sensor network queries. The contribution of this chapter is analytical solution for approximated distribution of delay and loss of aggregated data for a tree-shaped sensor query, assuming that wireless MAC protocol uses exponential back-off in case of transmission errors. This approximation allows control over data level IQ metrics of *data completeness*, *data coherence* and *coverage*.

3.1 Introduction

Recent developments in sensor networks have made it possible to gather up-to-date information about the environment, through SQL type queries [BW01] that capture streams of aggregated information over long periods of time. However, for this information to be useful it has to be timely and complete; in other words, there has to be a preservation of bounds on the *delay* and *loss* of data. These considerations constitute attributes of *information quality* delivered by a sensor network [BNQP05, BDQ$^+$05, LN00].

In the case of aggregated data being delivered to a consumer, (and this is most often the case in sensor networks), instead of the ratio of delivered data to sensor produced data, *completeness* may actually mean the ratio of data that is used in producing the aggregated result delivered to the consumer, to

the data that is generated by a sensor. As seen above, data loss may arise from resource unavailability or large delay which makes intermediate nodes assume that the data was lost.

If we want to have a method of comparing data completeness from different nodes in the network, we need to find a way of quantifying the amount of data lost. For some environments it would be preferably to have an admission control procedure which would ensure certain bounds on probability that the data is lost and not taken into account when producing query result. However, this procedure should take into account sensor network specifics as compared to other network environments. In particular, it is necessary to minimize overhead introduced into operation of a sensor network, especially communication overhead.

Because of the limits on the communication overhead we cannot assume that we have complete up-to-date knowledge about network configuration and operation. Also we have to assume that we cannot employ protocols with high traffic that would coordinate the resource allocation. However, we assume that we can piggyback some mechanisms already existing in sensor networks. The examples of such mechanisms are routing or query dissemination. In particular, the TinyDB semantic routing tree [MFHH05] construction procedure can be used to collect information about query tree.

An important assumption, which is also a motivation for this work, is that in addition to sensors over which we have limited control there could also be other sensors deployed in the same environment sharing the same wireless media. These other sensors can participate in some other applications and in many cases cannot be controlled. However, even if we can control other sensors, still, due to complexity of interference between many applications consisting of many queries it may be impossible to analytically predict effects of their operation on a query under consideration. For this reason we propose to use measurement to obtain the important parameters of network operation, and use admission control to limit the number of queries executed on a node so that the requirements of bounds on loss of data from existing queries are satisfied. The additional benefit of measurement-based approach is that many parameters used in admission control decision are obtained locally on a sensor, minimizing additional communication.

3.1.1 Motivation for the choice of method

The main difficulty in tackling the problem of QoS in sensor network is the lack of the appropriate modelling of the sensor network behavior.

The most universal approach for analysis of network behavior from the point of view of QoS provided is *network calculus* [BT01]. In fact, there are

attempts to use the network calculus model for building the model of sensor network QoS [SR05] and as a basis for the QoS control scheme [ZPB02]. However, in its basic form, network calculus requires the network elements to have deterministic bounds on the service time. In the sensor network environment, however, the service time of the wireless shared access networks is unpredictable and in general not bound. In this case the apparent way is to use the stochastic networks calculus. However, existing stochastic network calculus approaches have certain limitations. The method described in [SS99] has limitations on the envelope functions for the arrival and service process, namely the requirement that for N node path we need to have an envelope function which has N times integral. Besides, the fact that it is service time, not arrival process, is unpredictable, creates additional difficulty. Approach described by [Lee95] considers the case of variable service time of similar packets. However, this work only considers envelope functions which have exponentially bounded burstiness, which may not be the case for the sensor networks, because the service delay distribution in the case of wireless network may be very different from the exponential.

In fact, many wireless protocols employ CSMA-based media access protocol with exponential back-off increase in the case of failures. According to [JNR05] the service delay in this case have a Pareto distribution, which, compared to the exponentially bounded burstiness model, has a heavy tail which may significantly affect the statistical QoS parameters.

After taking into account the problems of developing a general model, we decided to build a restricted model of the sensor network query based on the following assumptions:

- We consider the delay on the node down the stream to be independent from the service on the previous node. By this we assume that the arrival process on the downstream node is only determined by the rate of arrival, and that the burstiness on this node is similar to the one on the previous. In fact, since we assume that the sensor data is time-generated, then the arrival process on each node is a periodic sequence.
- The delay due to CPU processing is not considered.
- The network part of the sensor network is based on the CSMA protocol with exponential increase of back-off interval.

These assumption, although quite restrictive from point of view of query capabilities, enable us to derive a simple model which uses very little global knowledge to derive query-level statistical performance parameters.

3.1.2 System assumptions

We assume that there is a set of sensors on which continuous queries are deployed. Queries are in the form which is used in TinyDB, that is

```
SELECT aggregate_value FROM sensors
   WHERE condition_1
   GROUP BY attribute
   HAVING condition_2
   SAMPLE PERIOD period
```

These type of queries are deployed over a subset of sensors from sensor network and query communication form a tree from this subset. Each sensor from a leaf node of a tree produces data. Non-leaf sensors may or may not produce data. Every non-leaf sensor node which has more that one child or generate data itself also perform aggregation of the data units.

We assume that the data is aggregated on the non-leaf nodes in the following manner. Aggregation operation requires the data from all children to be aggregated before sending the result to a node up the query tree. Every non-leaf node knows the number of its children in the query tree. Data units are aggregated only if they were generated at the same epoch of a continuous query. The computation of aggregated value of $k > 2$ data units does not require all the data units to be available. Instead, computation of aggregated value is done pairwise, when only two values are aggregated at a time and the partial result is aggregated with other data units or partial results. Therefore aggregation of $k > 2$ value takes $k - 1$ operations and in no time more than one data unit or partial result need to be stored.

When first data unit from a particular epoch of a query arrives to a node, it is placed into the pairing buffer. As soon as another data unit arrives, the two of them are handed to the CPU for aggregation. If data units from other children are expected, the result of an aggregation operation is placed back into the pairing buffer, otherwise it is placed into the network buffer. However, if data from some children does not arrive by some deadline, the node decides that the data was lost and sends partial aggregate to the network buffer for transmission up the query tree.

The flows of data inside the sensor node is shown on the Figure 3.1

Therefore the data has to wait in three buffers inside a node

- Pairing buffer - in this buffer arrived data from different children nodes is waiting for the date from remaining children to arrive
- CPU buffer - waiting for CPU resources to perform aggregation

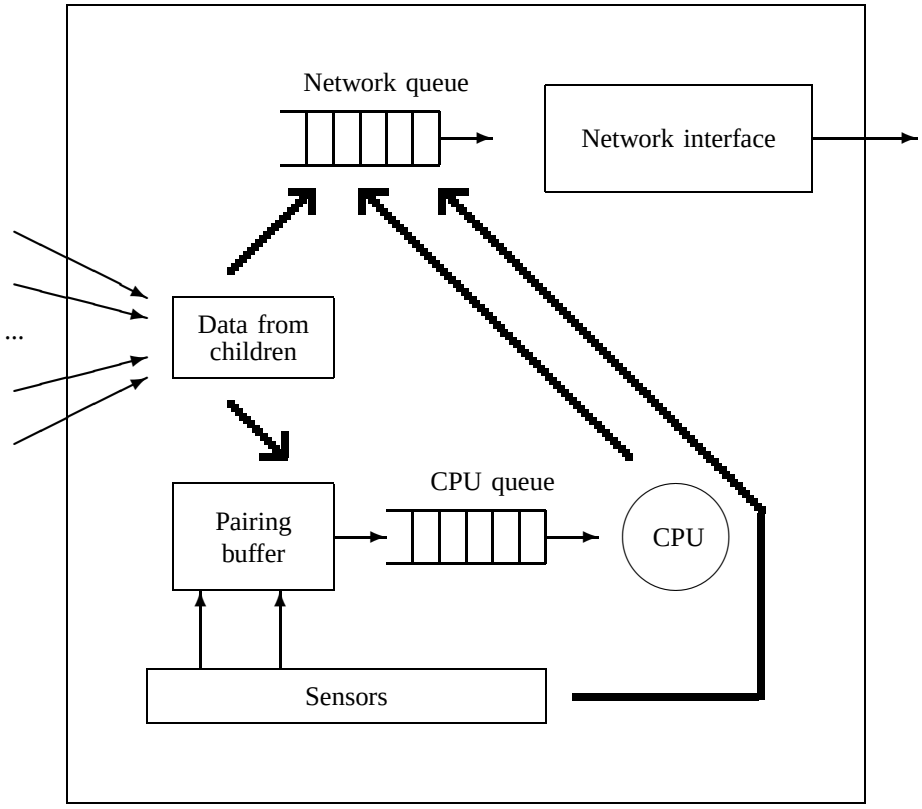

Figure 3.1: The flow of data inside a sensor node and structure of the waiting buffers. Data units arriving from children nodes are either sent to pairing buffer to wait for arrival of other children or sent directly to the network interface module for transmission. Data units after aggregation are either sent to the network buffer or back to the pairing buffer in the case of more data units expected

- Network buffer - aggregated data unit waiting for transmission to a parent node up the tree

In this paper we assume that aggregation operation is relatively simple and therefore the CPU is lightly loaded. In this case CPU queue is short and does not significantly impact delay characteristics. One of the reasons for this is that effective CPU speed can be easily raised either by using nodes with more powerful CPU or making wake-up periods longer. However, occupation of the pairing and network buffers depend mostly on the outside conditions such as total transmission rate in the particular wireless channel. Since we can not change much the network parameters but can change CPU, the network will likely become a bottleneck and therefore make the most impact on the data unit loss. Another reason why it is the communication phase which contributes mainly to the loss of data units is that the performance of wireless network interface is highly non-deterministic compared to CPU.

Therefore we focus our admission control scheme on the loss due to the network delays and overflow of network and pairing buffers. Another reason for concentrating on the network part specifically is that the communication phase consumes significant share of sensor battery. However, even if data generation rate of a sensor is low, it is possible that the delay in obtaining wireless channel for transmission is high, meaning that the node has to spend much time listening to the channel and therefore suffer from high battery drain. The admission control would decrease the total amount of traffic in the channel, thus leading to battery saving.

The rest of the chapter is organized as follows. In the next section we describe the model of wireless network access and propose distribution for delay characterization. Section 3.3 describes how query loss characteristics can be obtained from the parameters of wireless access model. Section 3.4 describes the admission control procedure. In section 3.5 we present our simulation results to support the proposed procedure.

3.2 Wireless delay model

Currently available sensor nodes use variations of CSMA/CA media access protocol. Newly proposed MAC protocols [YHE02b, WC01] addressing specifics of sensor networks are also CSMA/CA based. Also there is a standard [ANS03] aimed for the use in personal and sensor networks, which in particular uses CSMA/CA with exponential back-off. Therefore our analysis we based on the model where CSMA/CA with exponential back-off is used for wireless access.

[JNR05] showed that for the case of slotted CSMA/CA MAC protocol and exponential packet arrival the probability of waiting time (comprising of queuing and service delay) W exceeding a value T, for the large T is given by

$$Pr(W > T) = \frac{\lambda(W_{min}C^{'}(1))^{B}}{1-\lambda\beta^{'}(1)}\alpha^{*}(logT)T^{1-B} + O(T^{-B}) \qquad (3.1)$$

where $C(z)$ is the probability generating function for the number of busy slots between two consecutive idle slots, $\beta(z)$ is the probability generating function for service delay, $B = -log_2 p$ where p is the collision probability, w_{min} is the minimum backoff interval, λ is the arrival rate and $\alpha^{*}(x)$ is some periodic function with small fluctuation. $C^{'}(1)$ and $\beta^{'}(1)$ are therefore equal to the average busy interval and average service time, correspondingly. Note, that by the service time we mean time to acquire the channel and transmit a packet.

Important is the fact that asymptotically the distribution behaves as a heavy-tail Pareto distribution. Since we want to limit the data loss in a sensor network to a small probability, we therefore cannot ignore the heavy-tail property and need to use Pareto distribution to approximate actual packet delay distribution. The equation (3.1) also gives us the important characteristic of this distribution, namely that the power parameter is equal to $1 - B$.

The assumption of exponential arrival may not hold in actual sensor networks where continuous queries are producing results in fixed interval times. Especially since the wireless channel in the case of a single CSMA/CA MAC controller provides common media for receiving and transmission of data for different neighbor nodes, which significantly reduces possibility of arrival of several packets in a short time interval. Because of these factors the arrival process on a node may not be very different from the fixed rate arrival from point of view of influence on the total packet waiting time. In fact, in our simulation scenario the probability of delay distribution for nodes on the edge of the network with strict deterministic arrival was not much different from the one closer to a query root and in both cases was proportional to the T^{-B}. That is the power parameter of the distribution is the same as in the case of service delay distribution. (Asymptotic characterization of service delay is placed into appendix). For this reason, we assume for the rest of the paper that the arrival process to the network interface of sensor nodes is deterministic with packets arriving in fixed intervals.

Also note that the fact that the tail of the delay distribution behaves as a Pareto distribution is valid for all protocols which use exponential back-off in the case of transmission failure. In the work of [JNR05] collision was con-

sidered as such a failure. In the IEEE 802.11 protocol [ANS99] collision is not detected on the sender, and back-off timer exponential increment happens when transmission attempt does not lead to a positive confirmation. However, as long as we have a method to quantify the probability of failures, we can quantify the parameter of the service and full delay distributions.

We would like to approximate actual distribution with another, parameters of which we could easily obtain. This approximated distribution should be close to the actual delay distribution for the large values of T. We suggest to use a Pareto distribution as an approximation. The general form of Pareto distribution is

$$P(X > x) = (\frac{x}{x_{min}})^{-k}$$

with the mean value

$$E(X) = \frac{k}{k-1} x_{min}$$

We do not know both parameters k and x_{min}. However, they can be evaluated using following measured values.

B - the logarithm of probability of contention window increment

$\overline{W}$ - the average waiting time of a packet on a network interface of a sensor

R - the ratio of packets whose delay is higher than the $\overline{W}$

$\overline{U}$ the average delay for packets whose delay is higher than the $\overline{W}$

To obtain k and x_{min} from these measured parameters, we can use one of the four methods:

1. We assume that $k = B$ and the mean value of the approximated Pareto distribution is equal to the actual mean value. In this case, the delay distribution takes the form

$$Pr(delay > T) = (\frac{TB}{\overline{W}(B-1)})^{-B} \tag{3.2}$$

2. We assume that $k = B$ and that the point ($\overline{W}$,R) belongs to the distribution, i.e. $Pr(delay > \overline{W}) = R$. Then the distribution is given by

$$Pr(delay > T) = R(\frac{T}{\overline{W}})^{-B} \tag{3.3}$$

3. We assume that $k = B$, but instead of using mean value for the whole distribution, we only use mean value for the delays above mean delay.

Since we take only part of the values, actual distribution has to be adjusted by the ratio factor R. Then the distribution is given by

$$Pr(delay > T) = R(\frac{TB}{\overline{U}(B-1)})^{-B} \tag{3.4}$$

4. We assume that the point ($\overline{W}$,R) belongs to the distribution and that the delay values after this point have an average of $\overline{U}$. In this case the distribution can be presented as

$$P(X > x) = (\frac{x}{x_{min}})^{-k} = (\frac{\overline{W}}{x_{min}})^{-k}(\frac{x}{\overline{W}})^{-k}$$

The first term is the probability $P(X > \overline{W}) = R$, the second is the distribution with minimum value $\overline{W}$, which, according to the measurement, has the average $\overline{U}$. Therefore $k = \frac{\overline{U}}{\overline{U}-\overline{W}}$ and the distribution is given by

$$Pr(delay > T) = R(\frac{T}{\overline{W}})^{-\frac{\overline{U}}{\overline{U}-\overline{W}}} \tag{3.5}$$

The reason why we suggested so many ways of computing approximated distribution parameters is because each of these methods has some weaknesses. The first method, originally suggested in [TBT05], has two weaknesses. First weakness is that the method requires B to be greater than 1. Second weakness is that the approximated distribution of the tail is strongly affected by the actual distribution in the range of small delays. The average delay in the actual case can be smaller than the average delay of the Pareto distribution that is close to the actual distribution tail. The second method eliminates first weakness but not the second. The second weakness is addressed in the third and forth methods by shifting the approximated distribution to the region of values greater than $\overline{W}$. However, the third method also requires B to be greater than 1. The forth method is the most universal, and can be used for other protocols or arrival patterns where it is not easy to find the distribution power parameter k analytically. However, it may also suffer from the strong influence of the actual distribution in the range of small delays.

3.3 Loss and delay in a node

As a next step after proposing delay model we need to derive a characterization of respective losses in terms of proposed node delay distributions.

As we already mentioned, we assume the following processing of data in a sensor node. First, a data unit is placed into pairing buffer to wait for the data from other children. Partial aggregation is performed on units stored in the pairing buffer. Then the data is handed to the network interface module for transmission up the query tree. Therefore the data loss could be due to three reasons: the network network overflow, the pairing buffer overflow and the exceeding of the timeout set on a parent node for arrival of data from a current node. For simplicity we assume that network buffer and pairing buffer are separate. Also, we assume that CPU is not a bottleneck and therefore ignore the time and loss probabilities associated with CPU processing.

3.3.1 Loss in the network buffer

Since we assume that data is arriving at a node in fixed intervals, then the occupancy distribution for the network buffer is linked to the total packet delay distribution. If arriving packet sees that a queue already contains k packets it means that the total delay for the packet being serviced is more than $(k+1)P_i$, where P_i is the period of packet arrivals. The total queue length L has a distribution

$$Pr(L > l) = Pr(delay > \frac{P_i l}{M_i})$$

where M_i is the data packet size.

3.3.2 Loss due to timeout

We assume that each node waits for arrival of data from all the children for some time $T_{timeout}$ and assumes that the data is lost and the incomplete aggregate have to be transmitted if timeout expires. Therefore there could be a case when data is not included into final result without being lost, but just due to the fact that timeout is reached. To evaluate the loss of this kind, we need to find a probability that data propagation time in the subtree rooted in a current node exceeds the value $T_{timeout}$.

Theorem 3.3.1. *This probability is given by*

$$Pr(T_{subtree} > t) \approx \sum_i P(T_i > t) N_i \tag{3.6}$$

where nodes i are all nodes below the current node and N_i is the number of children of a node i. $P(T_i > t)$ are distributions for separate nodes and are given by the corresponding distribution from one of the equations (3.2, 3.4, 3.3, 3.5)

We assume that each node in a tree has a Pareto delay distribution in the form $Pr[t_i > t] = A_i t^{-\alpha_i}$.

The tree with one level of depth has a delay $T = max(t_1, t_2, ..., t_i) + t_{parent}$, that is maximum delay for children plus delay at the parent node.

For the maximum we have

$$Pr[max(A, B) > x] = 1 - Pr[A < x] * Pr[B < x]$$

Therefore for the Pareto distribution, ignoring the terms of smaller order, we have

$$Pr[max(t_i) > t] \approx \sum_i A_i t^{-\alpha_i} = \sum_i Pr[t_i > t]$$

Since we consider neighboring nodes, then the values of $B = -log_2 p$, where p is the collision probability are quite close. That is, the values of α_i from the distribution are also close. Then, according to [HdV05], for the case of $|\alpha_i - \alpha_j| < 1$ we have

$$Pr[t_i + t_j > t] \approx A_i t^{-\alpha_i} + A_j t^{-\alpha_j}$$

For the general case

$$Pr[x + t_j > t] = \int_{-\infty}^{+\infty} Pr[x > t - s] f_j(s) ds$$

where f_j is a probability density function of the random variable t_j. If x is the maximum delay for the children and t_j is a delay at the parent, then $Pr[x = max(t_i) > t] \approx \sum_i Pr[t_i > t]$ and therefore

$$Pr[max(t_i) + t_j > t] \approx \int_{-\infty}^{+\infty} \sum_i Pr[t_i > t - s] f_j(s) ds =$$

$$= \sum_i \int_{-\infty}^{+\infty} Pr[t_i > t - s] f_j(s) ds = \sum_i Pr[t_i + t_j > t] \approx$$

$$\approx \sum_i Pr[t_i > t] + N_j * Pr[t_j > t]$$

where N_j is the number of children of a node j.

Since for this kind of distribution, similar to the original case of Pareto distribution , the distribution of maximum of several values is still a sum of distribution, we can repeat this for the tree having a depth more than one level. The resulting distribution is

$$Pr(T_{subtree} > t) \approx \sum_i A_i t^{-\alpha_i} N_i$$

where nodes i are all nodes below the current node and N_i is the number of children of a node i.

3.3.3 Loss in the pairing buffer

We can obtain a bound on the pairing buffer loss if we assume that at least one data unit arrives immediately after it was generated by a sensor modality. This assumption is reasonable because, except for the nodes which only relay data, most sensor nodes from a query tree will produce data readings from local sensor modalities. These modalities generate one unit of data per epoch of the query. The locally generated data unit is placed into pairing buffer at the start of the epoch. This data unit is stored in the pairing buffer until the arrival of another unit from the same epoch. Then they are aggregated and the result of an aggregation is put back into pairing buffer. Therefore from the moment the data is generated to the moment when data from the last child arrives the pairing buffer contains one data units from this epoch of the query.

Let us assume that data units belonging to the same query are processed in the first-come-first-served order. Then the total number of data units belonging to a query i stored in the pairing buffer is $\lfloor T_i^{elapsed}/P_i \rfloor$ and the buffer space from one query is $L_i = M_i \lfloor T_i^{elapsed}/P_i \rfloor$ where $T_i^{elapsed}$ is the time elapsed since generation of the last unsent data from the query i, P_i is the period and M_i is the data unit size for the query i. The total amount of buffer is therefore $L = \sum_i M_i \lfloor T_i^{elapsed}/P_i \rfloor \leq \sum_i M_i T_i^{elapsed}/P_i$, .

The probability that the buffer space on a node occupied by the query i exceeds the value l is

$$Pr(L_i > l) \leq Pr(T_i^{elapsed} > l * P_i/M_i) \tag{3.7}$$

Therefore the problem of finding distribution of pairing buffer space first requires to find distribution of time delays from a subtree and then find a distribution of their sum. However, since a distribution of a sum of many variables having distributions such as the one from Equation 3.6 would be too complex, we have to use only to leading terms of the distributions. For example, we can sum coefficients of all terms from Eq.3.6 where orders are not less than highest minus 1. From [HdV05], we then can use the approximation for that case which gives for close B $Pr[L > l] = \sum_i Pr[L_i > i]$.

This result assumes the first-come-first-served order for the data form the same query. However, it is not always valid. In our system assumptions data waits for arrival of data from all the children. If some data is lost on the MAC level of the child then the next epoch data may arrive earlier than the current epoch reaches timeout for transmission of incomplete result. This violation of FCFS order may increase the pairing buffer use compared to the model of Equation 3.7.

3.4 Admission of continuous queries

Admission of a new query is performed on a candidate query tree given by some external routing algorithm such as semantic routing tree [MFHH05]. Admission control scheme consists of three components performed in different parts of the network. First component is per-node estimation of the new transmission delay probability parameters. It is done on each node included in the candidate query tree. The second component is run either on the root node or (if any) entity controlling the query distribution and operation. It assigns loss probabilities to the nodes of the query tree. The third component computes approximation functions of query-level probability distributions, computes estimated loss probabilities for already accepted and incoming queries and compares to the assigned probabilities. It is also done on every node of the tree, although pairing buffer check is only done on aggregation nodes.

When query arrives, the following steps of admission control are performed:

- By piggy-backing the query tree routing protocol, collect the current node parameters.
- Estimate the node parameters which would be observed after a new query is admitted.
- Re-compute the query-level delay and loss probabilities for already accepted queries which would be observed after. If distributions are not satisfactory, reject the query.
- If already accepted queries are not compromised, compute the probability distributions for delay and loss for every aggregation node of a new query.
- Based on the query-level IQ requirements, decide on the acceptable delay and loss probabilities for every aggregation node as described in the later section 3.4.2. If there is no assignment which satisfies the per-aggregation-node computed expected losses, reject the query. If no specific IQ requirements are given, the query can be accepted or rejected based on the query-level delay or loss probabilities.
- If all the above is done satisfactorily, accept a query.

3.4.1 Node parameters estimation

The parameters of delay distribution on a node for already accepted queries can be measured during their operation. However, when a new query arrives, we need to predict its expected on the node delay distribution. The general form of distribution is

$$Pr(W > T) = (\frac{T}{T_{min}})^{-B}$$

If we assume that the value of B remains the same as before admission of a query, then, assuming that values of $(W_{min}C^{'}(1))^{B} * \alpha^{*}(logT)$ from Equation 3.1 give a constant, we adjust T_{min} according to

$$T'_{min} = (\frac{\lambda'(1-\lambda\overline{\beta})}{\lambda(1-\lambda'\overline{\beta})})^{\frac{1}{B}} T_{min} \tag{3.8}$$

Where T'_{min} and λ' are new values of T_{min} and λ. As a base for distribution any of the equations 3.2, 3.4, 3.3 and 3.5 can be used. However, we recommend to use Equation 3.5 because it computes the Pareto distribution power parameter and therefore is valid not only in the case of arrival similar to deterministic.

All these parameters of the proposed approximated distribution could be measured locally on a sensor and do not require much processing power or additional network activity. Although Equation 3.8 was obtained by making some assumptions for the Equation 3.1, it still gave good results for the case of non-exponential arrival in our experiments.

3.4.2 Loss probability assignment

When a new query arrives, we would like to verify that the loss characteristics of a new query can be satisfied without violation of the loss characteristics of the existing queries. Since we want to make admission scheme to be based on local parameters as much as possible, it is not a good idea to compute loss probability on a particular node and then recompute total probability for every query tree which includes current node. It is more reasonable to assign to each node of a newly arrived query tree some bound on loss probability and then make admission decision on the basis of these bounds.

The query is deployed in the form of a tree. We assume that we can obtain the basic characteristics of the tree and subtrees for each node - namely, the number of nodes and the depth of a tree. This can be done for example by piggybacking the routing protocol. From this tree information and the required limits on the end-to-end loss probability we compute rough estimation on the required loss for each node.

This assignment depends primarily on the goal we would like to achieve. In the context of the data-level IQ metrics we can achieve data *completeness*, *coverage* and *temporal coherence*. Below we will describe example assignments for different IQ metrics using the following scenario. We assume that the set of sensors **Q** is selected for data acquisition and set **R** is a set of sensor nodes relaying the data to the consumer, the sink node in the center of the deployment and

- **Data completeness and coverage** specify the bounds on the loss of data. Completeness specifies the total loss and coverage specifies geographical loss variation. Therefore we need to bound the loss probability on different nodes throughout the network. This bound is obtained from the total bound on the loss and topological properties of the network.

  If we expect the total loss of data from any source sensor node to reach the consumer node at most to be equal to p, then for a node having a subtree of depth N where the total loss probability should not exceed p (possible coverage requirement), we assign local probability bound equal to $p_{local} = 1 - (1-p)^{1/N}$. In the case the consumer node is positioned in the geographical center of the network, the value of N is expected to be $\frac{\sqrt{|R|}}{2}$

- **Temporal coherence** specifies the bounds on the deadline for the data delivery and on the probability of its violation. Since we want to define this bound for any case of sensors acquiring the data, we may want to limit the probability of loss for the case of any node allocation. For example, let us consider the case we want to satisfy the deadline time T_D with probability p in the network of with the data consumer positioned in the center. Every aggregation adds one term to the sum in Eq.3.6. Therefore the total probability of delay for the query exceeding T_D can be given by

  $$Pr(t > T_D) \approx L(T_D) * (max|\mathbf{Q}| * 2 + max|\mathbf{R}|)$$

  If the number of acquisition nodes is bound by n, then we can set a requirement for loss probability $L(T_D)$ on a single node to be bounded above by $\frac{p}{2|\mathbf{Q}|+\sqrt{|\mathbf{R}|}/2}$

For every node of the tree we need to recalculate parameters of distribution function to take into account the load from a new query. In particular, we calculate the expected change in the mean waiting time according to

Equation 3.8. Note, that as a current value of $\overline{W}$ we are using measured value. The value $\overline{W}_{new}$ gives us estimate on the $\overline{W}$ after acceptance of the new query, and it should be replaced with the measured $\overline{W}$ later, when a new change happens in the query allocation. Then we estimate new loss probabilities for the pairing and network buffers, probability of timeout and check that it is still satisfactory. If it is, then the query can be accepted.

3.4.3 Loss probabilities estimation

This component combines the probability of losses due to various factors according to the models presented in the section 3.3. The functions representing probabilities of loss are computed from node delay distributions, which, in turn, use the parameters predicted according to Equation 3.8.

Every node of a tree except the root computes probability of loss due to network buffer overflow according to

$$Pr(L > S_n) = Pr(delay > \frac{S_n}{\lambda_i M_i}) = (\frac{S_n}{\lambda_i M_i T_{min}})^{-B} \tag{3.9}$$

where S_n is the size of network buffer, λ_i is a data rate on a node.

Every aggregation node computes probabilities of loss due to timeout according to the

$$P_{timeout} = Pr(subtreedelay > T_{timeout}) = \sum_i (\frac{T_{timeout}}{T^i_{min}})^{-B^i} N^i \tag{3.10}$$

where i are all the nodes in the subtree under given aggregation node, N^i - number of children of node i.

Loss due to pairing buffer overflow is computed according to

$$P_{pairing} = Pr(L > S) = \sum_j \sum_i (\frac{SP^j}{T^i_{min} M^j})^{-B^i} N^i \tag{3.11}$$

Where, j are queries using a particular aggregation node, i - nodes used by subtrees of queries, S_p is the size of pairing buffer, M_j is data unit size, P^j - period of a query j.

Note that the acceptance of the query means that the loss due to lack of resources is limited. However, if we choose not to send some data to the parent for aggregation, such as proposed in [MSFC02] for the case of MAX aggregation, this does not negatively affect the outcome of the query because this decision was taken after considering the impact of data on hands on the

query result. Therefore this admission control mechanism is transparent from point of view of algorithms attempting to minimize communication in a sensor network.

3.5 Simulation evaluation

We evaluated the validity of the node and query delay and loss models through simulation using NS2.

3.5.1 Simulation setup

We simulated the following scenario. There are 27 nodes and there are several queries running on these nodes. The scenario is depicted on the 3.2

As a source of the data for the queries we used the Constant Bit Rate (CBR) traffic generators. Each of the CBR generators for one query produced packets at the same rate r. To avoid the effect of the peak load on the network in the moments epoch data is generated the CBR source was modified to produce the packets at times $t_i = t_0 + \frac{1}{r}i + e$ where e is uniformly distributed random variable from the interval $(-\frac{1}{2r}, \frac{1}{2r}]$. Although this randomization of data generation does affect the total time for a query data to reach the consumer node, this effect is small compared to the total time.

We implemented the sensor nodes performing the data aggregation according to the model described on Figure 3.1. Each node has one local sensor modality represented by the CBR generator and some number of children nodes. If there are several queries running on a node, each query has its own aggregation agent but they all share the same aggregation buffer. The data aggregation works as follows. The node knows the total number of its children. Every data unit has has a epoch number field and data is aggregated only belonging to the same epoch, so that for every epoch there is a combined query result delivered to a query root. The data is aggregated pair-wise, therefore only at most one data reading is stored in the aggregation buffer for a given epoch. In this simulation we did not include queueing at the CPU, because of the assumption stated in section 3.1.2.

We implemented logging of contention window increments in the IEEE802.11 MAC implementation. The probability of failure with subsequent backoff increment is therefore was computed as a ratio $\frac{C_{inc}}{C_{ACK}+C_{inc}}$, where C_{inc} is the count of increments and C_{ACK} is a count of successful (acknowledged with ACK) transmissions.

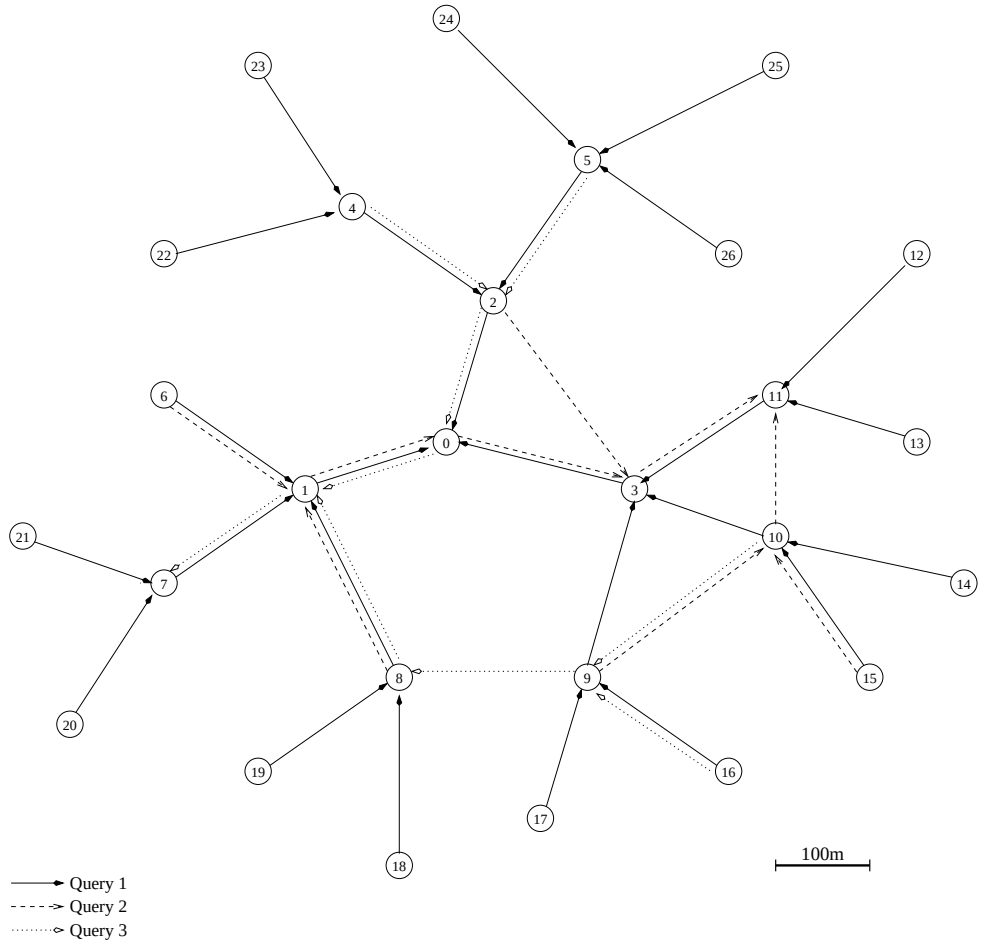

Figure 3.2: The structure of the sensor network used in the simulation. The sensor network consists of 27 nodes. There are 3 queries running on the nodes, the direction of dataflow for each of them is shown by the corresponding arcs

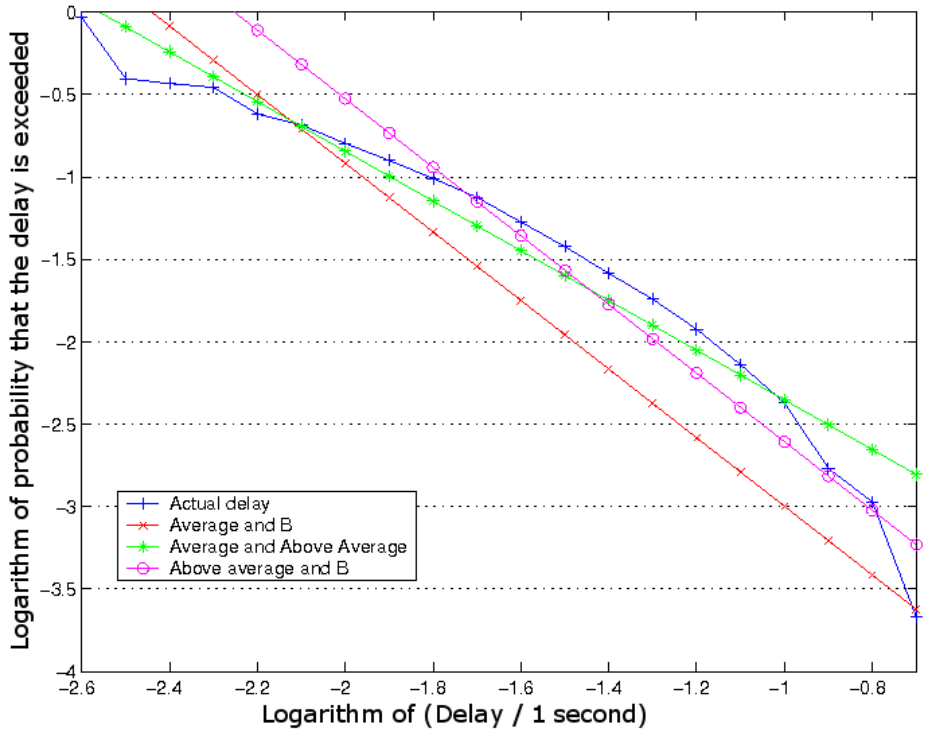

Figure 3.3: Simulation results. The actual and approximated distribution of the total delay in a single node. Three approximation methods, described in the section 3.2, are presented

3.5.2 Node delay distribution

First we need to confirm that we can use approximations provided by Equations 3.2, 3.4, 3.3 and 3.5. In the above experiment we computed the actual delay distribution of packets in a node. The example distribution for a node 3 is presented on the Figure 3.3. The figure shows four delay probability distributions - one of them, denoted "Actual delay" is a distribution of delay for a sub-query obtained using simulation. Three others are approximated distributions using approximations expressed by Equations 3.4, 3.3 and 3.5. Since Pareto distribution is a straight line when logarithmic scale is used, we use logarithmic scale for our case as well.

It can bee seen that, although with some difference, the three methods approximate the actual distribution rather closely.

3.5.3 Query delay distribution

The next step is to evaluate how well the query delay is given by the Equation 3.6. In our case the query delay is defined for the data readings belonging to the same epoch and is equal to the time since the last data unit from the epoch was generated to the moment the aggregated result is produced at the sink node. We chose the last data unit as a starting moment because in this way the delay is mainly the characteristic of the data aggregation and propagation process, not the data generation as ti would be in the case we chose the first data unit as a starting point. In terms of aggregation, when two data units are aggregated, the timestamp is put into the result equal to the maximum of two timestamps of aggregated data units.

Figure 3.4 shows the example of actual and approximated query delay distribution. Two approximations are used, given by Equation 3.3 (Average and B) and by Equation 3.5 (Average and Above Average). In this case, however, we see that the more complex approximation of Equation 3.5 follows actual distribution much better.

3.5.4 Pairing buffer occupancy

Important parameter for the admission of a query onto a sensor node is buffer space for storage of data units to be aggregated. We separated the types of storage into network buffer and buffer for storage of data units waiting arrival of other data units from the same epoch for aggregation. Figure 3.5 shows the actual and approximated distributions of pairing buffer occupancy. The approximated distribution is obtained by summing the distributions obtained from sub-query delay distributions given by Equation 3.11.

3.5.5 Network buffer occupancy

This experiment checks validity of the use of Equation 3.9 in approximating the buffer requirements and loss probability due to network buffer overflow. The result is shown on the Figure 3.6 and demonstrate the similar asymptotic behavior of the actual and approximated buffer distributions. The approximated distribution is obtained using "Average and Above average" node delay distribution.

3.5.6 Query Admission control

When a new query is being admitted to a network, we need to make sure that the existing and additional queries will still obtain satisfactory statistical

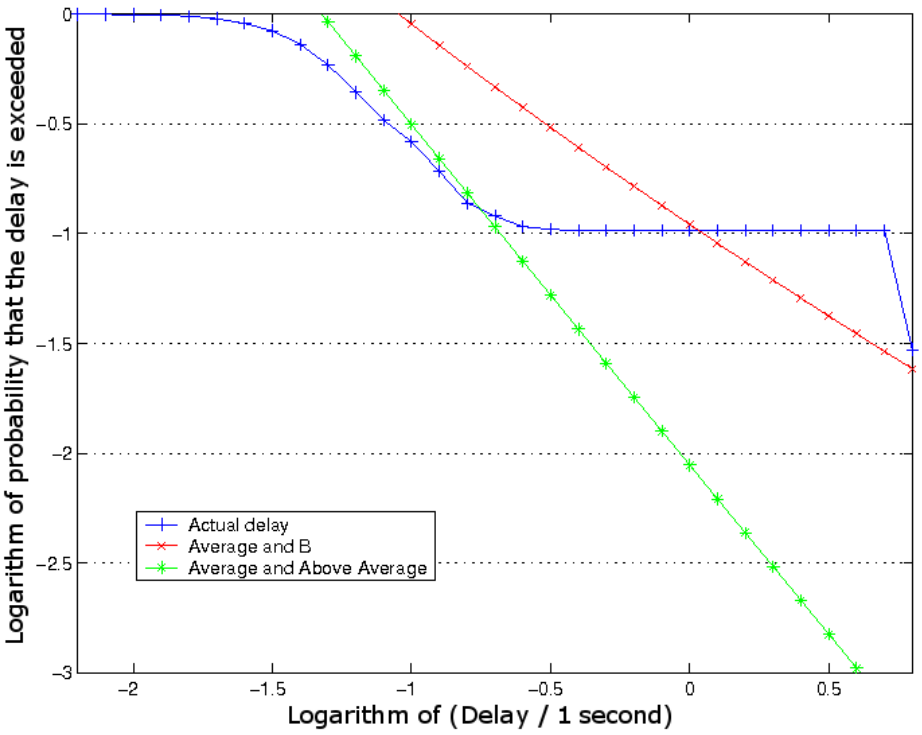

Figure 3.4: Simulation results. The actual and approximated distribution of the query delay. Because of the limitations on the failure probability, the method "Above average and B" is not presented. However, it still can be used on some of the nodes where failure probability is less than 1/2. The long horizontal extension of the actual delay distribution is due to the losses on the MAC level which delay some data until local deadline.

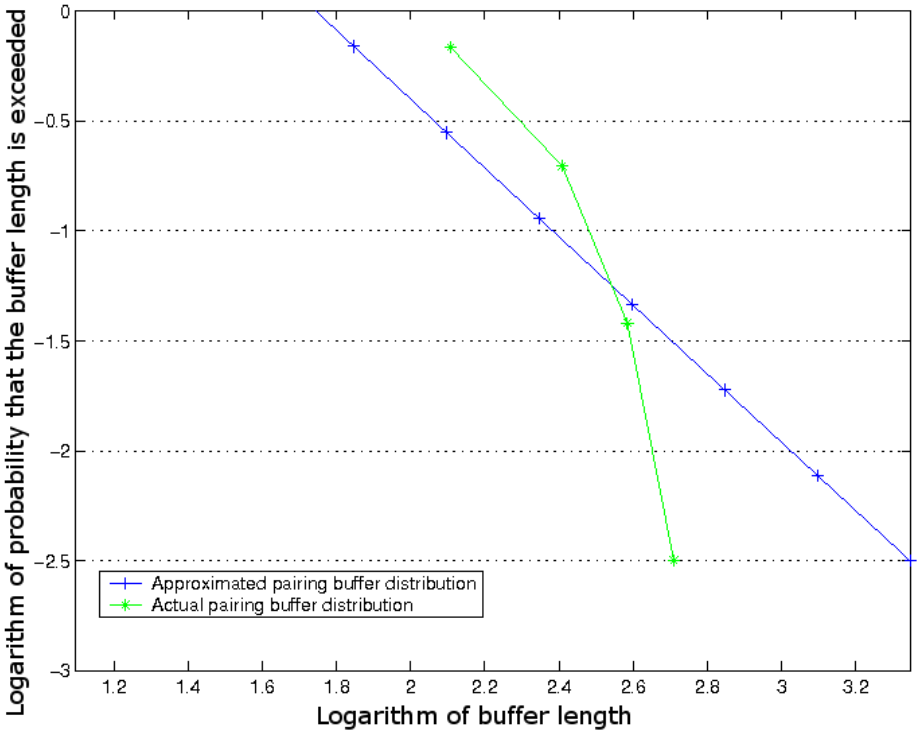

Figure 3.5: Simulation results. The actual and approximated distribution of the pairing buffer occupancy for node 7 in the system with 3 queries. Approximation takes into account delay distribution of 2 queries using buffer space on a node

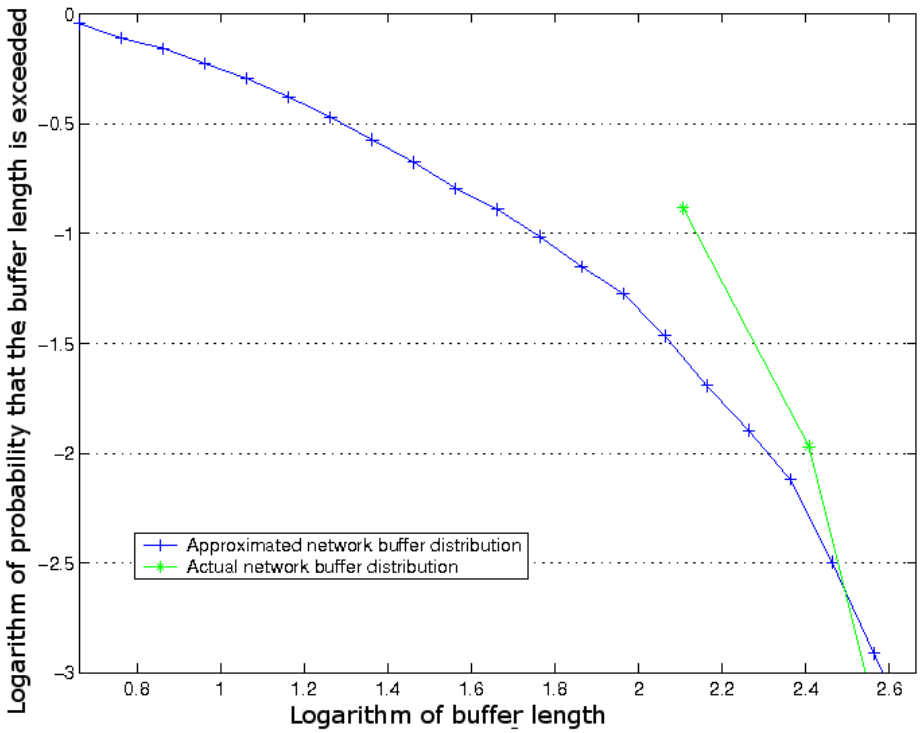

Figure 3.6: Simulation results. The actual and approximated distribution of the network buffer occupancy for the node 6.

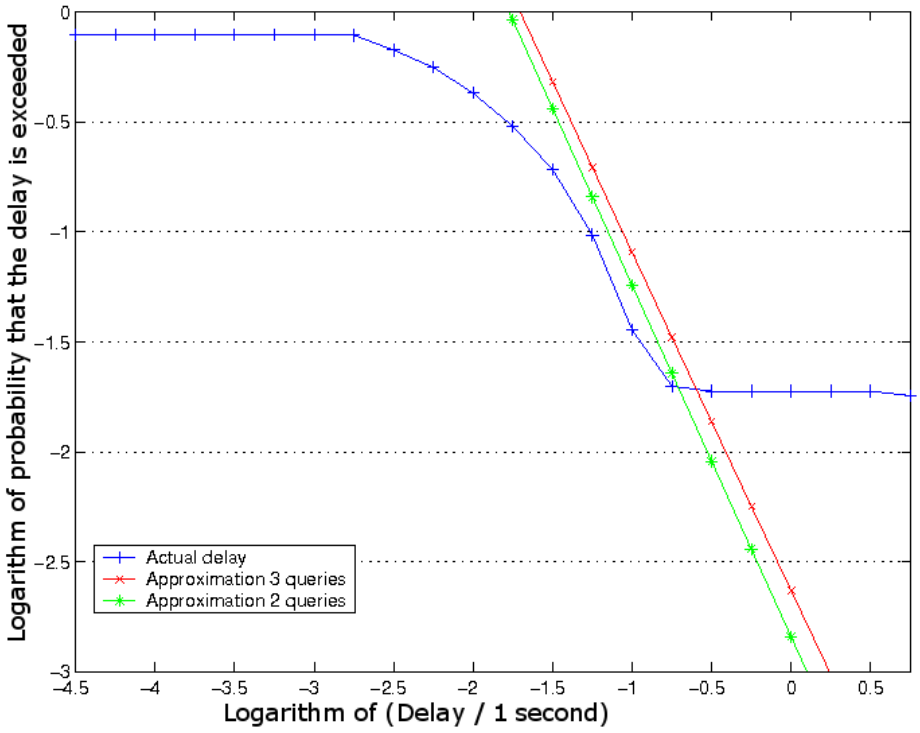

Figure 3.7: Simulation results. The actual and approximated distribution of the query delay for the case of admission of the 3rd query. The 3rd query rate is 4 kbps. The "Approximation 2" is the approximation of the distribution based on the measured parameters of the system with only two queries. The "Approximation 3" is the approximation for the query delay based on the parameters measured for all three queries.

QoS. For this we need to estimate the parameters of the queries beforehand. We propose to use the estimation according to Equation 3.8 to adjust local parameters of each node and then use those adjusted parameters to find the performance of the queries after admission of the new one.

The figures 3.7, 3.8 show the example of such an approximation for two different admitted query rates. The second rate, 8kbps, it quite high for this network configuration, where the main query creating the load runs and the rate 10kbps. It can be seen that for relatively small rate the change in the query performance is reflected quite well. For higher rates, however, it would be useful to have a network model which can give the change in the probability of transmission failure.

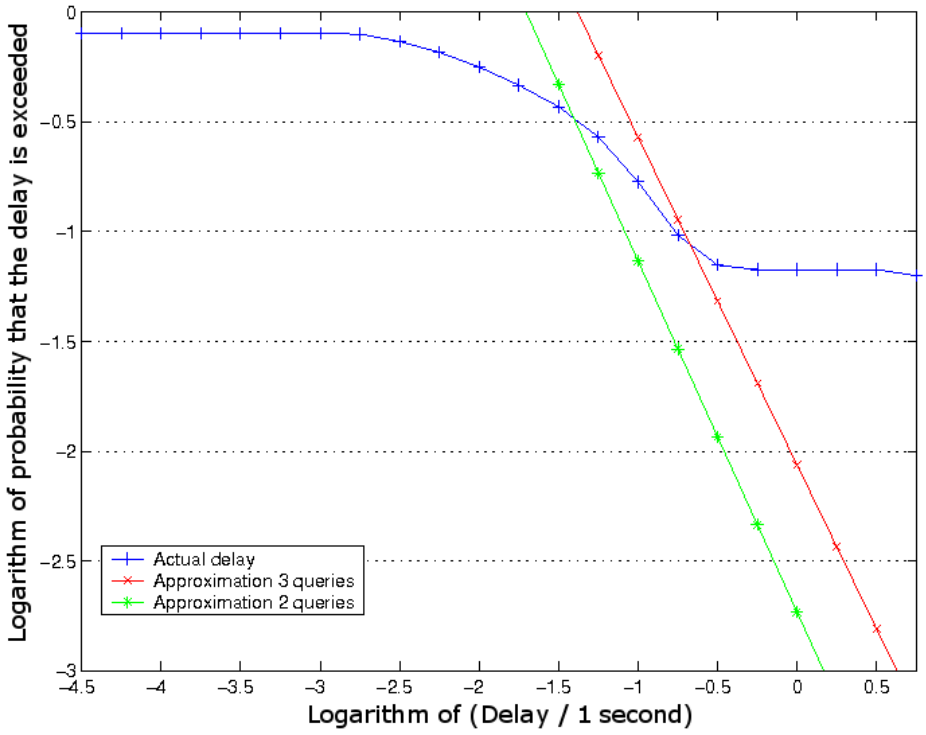

Figure 3.8: Simulation results. The actual and approximated distribution of the query delay for the case of admission of the 3rd query. The 3rd query rate is 8 kbps. The "Approximation 2" is the approximation of the distribution based on the measured parameters of the system with only two queries. The "Approximation 3" is the approximation for the query delay based on the parameters measured for all three queries.

3.6 Conclusion

In this chapter, we proposed an approximation scheme which gives an estimation for delay and loss of aggregated sensor data in a tree-shaped sensor network continuous SQL-type query. Different node approximation methods were proposed and result for the query or sub-queries was derived. We provided examples of how this approximation can be used in computing data-level IQ metrics of data coherence, data completeness and coverage. We also indicated how this scheme can be used in making admission control of such queries in a sensor network.

The scheme uses the stochastic model of wireless data channel based on Pareto distribution. The parameters of this Pareto distribution are obtained by the local measurement of statistical service parameter. The strong side of this scheme is that it involves only minimal information exchange during query dissemination and admission time, and this information can be transferred by piggy-backing the query routing algorithm. The drawback of the scheme that for the sake of simplicity and uniform representation of query delay distributions as linear combination of node delay distributions, we had to exclude CPU delay from the consideration.

The novelty of the scheme is in the fact that it models the probabilistic behavior of the whole query and that the basic information for this model obtained by building probability distributions of local parameters.

The scheme was developed for the sensor networks based on CSMA/CA media access protocol with exponential back-off and therefore may be applicable to networks based on a broad range of MAC protocols, including IEEE 802.15.4 [ANS03].

Chapter 4

Phenomena-aware IQ management

As described in the section 2.3, if we wish to provide a good information quality in the sensor network observing a dynamic system, we have to include the observed phenomena state as one of the factors in making a decision about resource use. In this chapter, we describe how such a system can be built in the case of a system collecting and fusing discrete observation values. In this chapter, since the values are obtained in a process of fusion, the addressed level of information is the high-level collective information.

4.1 Introduction

Multi-modal sensor networks are expected to become an important part of the future ubiquitous intelligent computing. They are capable of obtaining information and deriving complex characteristics of the phenomena taking place in the environment being monitored. However, since the processing of information from sensors of different modalities is complex, and applications, built in such environments, are highly dependent on the particular content of the information, the deployed sensor network applications are mainly custom-made. However, for sensor networks to become widely used there has to be a way to separate the complexity of information processing from the application development.

One of the approaches to such separation is to use an architecture where a layer which processes information supplies the applications with the processed high-level information. The additional advantage of such a separation is that the high level information can be reused by different applications, enabling more complex environment intelligence and saving resources for

data acquisition. However, in this case we need to introduce some metrics which characterize the quality of the supplied information and ensure that the data supplied is suitable for the application needs. The efficient way to satisfy these application requirements is to choose the most appropriate sensor nodes and sensor modalities which would provide a required IQ for the current state of the system. And since the state of the environment being monitored changes, the IQ management system becomes a sensor resource management system depending on the state of the monitored environment.

The motivating scenario we use here is the application for patient activity tracking and behavior analysis in the home or hospital ward environment. This kind of scenario is very interesting because of the two reasons. First, the sensors for this system are essentially multi-modal, since the environment may change drastically enough to make useless any particular sensor modality, be it video, audio or position sensors. The examples of such changes are changes in the lighting for video, change in the noise level for sound, obstruction for position sensors such as ultrasonic or absence of tags for the tag-based positioning. Second, the set of applications may also change drastically, as the changing number of people or other changes in the environment invoke either new copies of existing applications or applications which are idle most of the time and only are supposed to respond to certain conditions. These changes in the application set may bring a heavy load on the underlying sensor infrastructure and therefore resource contention need to be addressed.

In the class of information uncertainty we see three main causes of IQ loss. First is uncertainty due to measurement error. Second is the change in uncertainty introduced by the multi-modal fusion. Third is the uncertainty increase due to losses of useful data in the process of delivery. First two are acquisition related, the third is delivery related. We need to find a model which potentially may enable to address all three of these causes. In addition, the it would be beneficial to address some information completeness metrics as well.

In the example of the patient tracking application the main IQ metrics are tracking accuracy, activity or behavior recognition false alarm rate, timeliness for notification of critical situations and completeness of the activity tracking in time. This is not exhaustive, but fairly representative set of metrics.

4.2 Objectives and scope

The objective of this work is to propose methods which enable to manage Information Quality in sensor networks for a more general class of phenomena-monitoring sensor network applications. We need to identify a possible class

of applications which can be general enough to be used in many installed systems yet specific enough so that the model can be created and used. In addition, since we want to make it phenomena-aware, and information about phenomena is collected by a variety of heterogeneous sensors, we need a model which can effectively translate the resource management decisions into phenomena-related metrics.

Therefore, to list down the main goals, we need to:

- Find general application class which would benefit from phenomena-aware resource management
- Propose a model of application which reflect the information collection and resource decisions
- Find IQ metrics which can be enforced using the model
- Find a way of translating model metrics into IQ metrics described in Chapter 2.
- Demonstrate that the model can be used for realistic systems.

The assumptions we use should be related to correspondence of actual applications and the model used. We state these assumptions later when we introduce the applications in Sections 4.5 and 4.7.

4.3 Related work

There are a few architectures which use a notion of quality of obtained information and do the sensor management with the aim to satisfy certain IQ bounds. MiLAN [HMCP04] addresses the question of how to guarantee IQ in the presence of resource constraints. However, MilAN uses very rough and non-transparent notion of quality. In QUASAR [LHY+04], the IQ notion of *application error tolerance* is introduced and addressed by using approximation for the cases when data is lost or unavailable due to resource limitations. However, the described approach is only applicable to rather simple in-network processing.

The work [DGM+04], which considers environment monitoring scenario, provides assurance on the deviation of the value acquired, thus ensuring uncertainty, by exploiting pair-wise correlation between different variables describing environment. For example, the voltage measurement on one node can be used instead of temperature o the other due to high correlation between these two values. However, such correlation cannot be used in the

dynamic environment since it describes the average case, and it also does not consider the case of resources being unavailable.

Midfusion architecture[AKS04], similar to our approach, separates information fusion into two levels - sensor and application, each of them represented by Bayesian network (BN). For the complete processing in the particular environment the BNs are combined in such a way that the results given by the combined BN satisfy certain requirement. The sensors are chosen on the basis of expected *definitiveness* of the result. However, this architecture requires particular representation of application and sensor levels and does not allow general requirements on information quality. Besides, the resource allocation is made only when either sensor availability or application requirements are changed and does not depend on the state of the observed system. Another problem is that, as we will demonstrate later, the inference on the basis of a static BN is prone to frequent estimation change due to the lack of the memory of the past.

In [ZLL+03] Feng Zhao et. al. described the conceptually framework of the information-directed approach to signal and information processing. In the paper the problem considered as an example was a tracking problem, which was formulated as a constrained optimization problem $Tr = \langle N, T, M, Q, O, C \rangle$, which includes the following models:

- Sensor network model N
- Set of targets T
- Signal propagation model M
- Set of queries Q
- Objective function O
- Set of constraints C

For a more general case of information acquisition, we can formulate phenomena monitoring problem as a similar constrained optimization problem $Ph = \langle N, X, M, E, I, R, O, C \rangle$, with the following models:

- Sensor network model N
- Set of observed environment characteristics X
- Sensing measurement model M and Estimation model E

- Model of information gain I with respect to resource use R. Objective function O is derived to obtain required tradeoff.
- Set of constraints C

A significant difference from the tracking scenario is the inclusion of the information gain model, which in many cases can be quite complex. The particular model depends mainly on the chosen method of information fusion.

In [ZJ06] was proposed a framework which made dynamic resource allocations according to the current observed phenomena state. The goal of the framework was to choose optimal set of sensors to monitor a system while maintaining a certain level of the quality of information obtained. As a model for fusion authors proposed to use the Dynamic Bayesian Network (DBN) [Gha97] model. However, their model was somehow unrealistic for the distributed sensor network scenario because it assumed that there is no data loss and that resources are always available.

In this chapter, we present a general set of IQ metrics and apply the generalized information-based approach of [ZLL+03] to the fusion system based on DBN as in [ZJ06] with more realistic assumptions that:

- There is an IQ degradation due to the losses of data in the network
- Resources are not always available due to contention between queries providing different variables

so that the mapping between the IQ metrics and the resource use can be obtained.

4.4 Notations and definitions

4.4.1 Notations

$\Theta = \{\theta_1, ..., \theta_K\}$ - set of system state descriptions obtained by the fusion system. In the terms of Bayesian networks, they represent a set of *hypotheses*.

Total set of sensors $\mathbf{S} = \{S_1, ..., S_n\}$ and set of sensors producing data at time t $\mathbf{S}_t \subseteq \mathbf{S}$

The measurement of a sensor i at the time t is $z_i^{(t)}$, the value of the measurements belonging to a finite set of possible values $F_i = \{f_1^i, ..., f_L^i\}$. The meaning of each f_j^i is either a particular *feature* extracted by a complex sensor modality such as video or audio, or an area of possible values for a continuous sensor modality such as position. By $\mathbf{Z}^{(t)}$ we denote a vector of measurements $z_i^{(t)}$.

Note, that such feature-based sensor characterization is in fact very general and very common. It is very general because it does not make any assumptions at all about the physical type of sensors, be it simple reed switches or complex systems of video and audio recognition. And yet in fact many system, even complex, use features because they try to hide the complexity of the underlying data and reduce the amount of information passed up to the top-level applications. Therefore they use more simple units actually describing some type of event or information detected.

The possible examples of sensors and features are:

- Video camera as a sensor and fact of presence or absence of a person or object from the camera view
- Microphone array and presence of noise from a certain direction or specific sound such as water flow
- Accelerometer on a person's wrist and hand specific motion detected or acceleration threshold exceeded
- Opening or closing of a door detected with a reed switch
- Proximity of a RFID-tagged object detected in a certain location by RFID reader

4.4.2 Bayesian Network model

Bayesian network [Jen01] is a graphical representation of the dependency between random variables. Because of the hierarchical dependency between variables the joint probability distribution can be expressed in terms of conditional probabilities of variables depending only on the variables immediately above the variable in the hierarchy, called parents and denoted by a set $\pi(X_i)$. The joint probability is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\pi(X_i)) \tag{4.1}$$

The example of the Bayesian network is presented on the Figure 4.1.

Usually, the top node of the Bayesian network model describes the system state Θ, called *hypothesis*, which we want to estimate. The leaf nodes of the network represent measured values z_i (in the case of sensor network - sensor readings or features). Given the joint probability, we can use it to estimate the state of the system based on the sensor data measured. It is done by computing the marginal distribution $Pr(\Theta = \theta_k|\mathbf{Z})$.

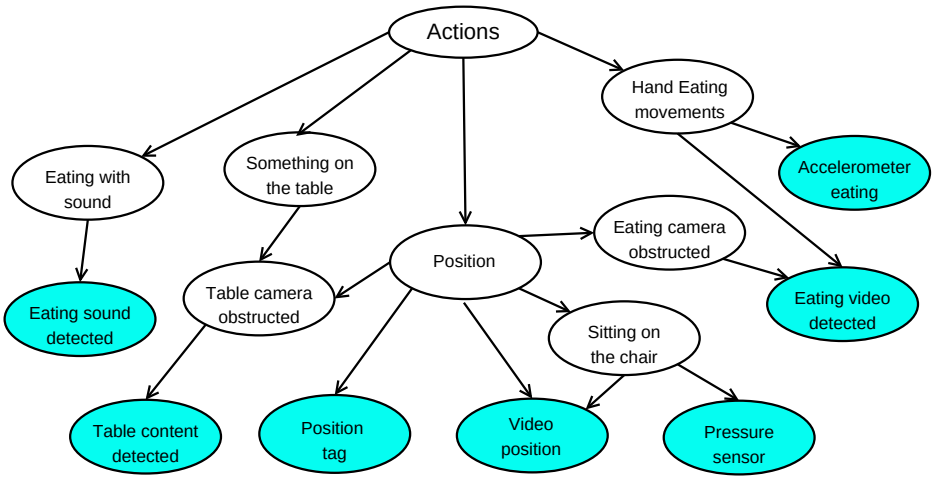

Figure 4.1: The Bayesian Network for estimation of the quality of action recognition of eating in the kitchen. The top node represent the activity we want to detect. Blue nodes represent the features provided by different sensor modalities. Actions node has three possible values: Nobody present, Person in the kitchen and Person eating

4.4.3 Dynamic Bayesian network model

Dynamic Bayesian network [Mur02] is a Bayesian network constructed from above Bayesian network by adding the temporal behavior to some of the system variables. There is a copy of the static BN for every time moment, describing the dependency between random variables at the particular moment. In addition, the nodes of BN describing the random variables which evolve in time become *temporal nodes*. These variables are assumed to have first-order Markov dependency on their values at the previous time moment and therefore each temporal node has an additional parent - a copy of the same node from the previous time slice.

The joint probability distribution therefore has time component and is given by

$$P(\mathbf{X}_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{n} P(X_i^t | \pi(X_i^t)) \tag{4.2}$$

Since the first order Markov process is assumed for the state of the *temporal nodes*, it is sufficient to keep only two time slices of the DBN to describe the system. First time slice describes the past, the second describes the future. If there was any data taken at the last time moment, it is added as evidence nodes to the first copy of the BN. Note, that although the first

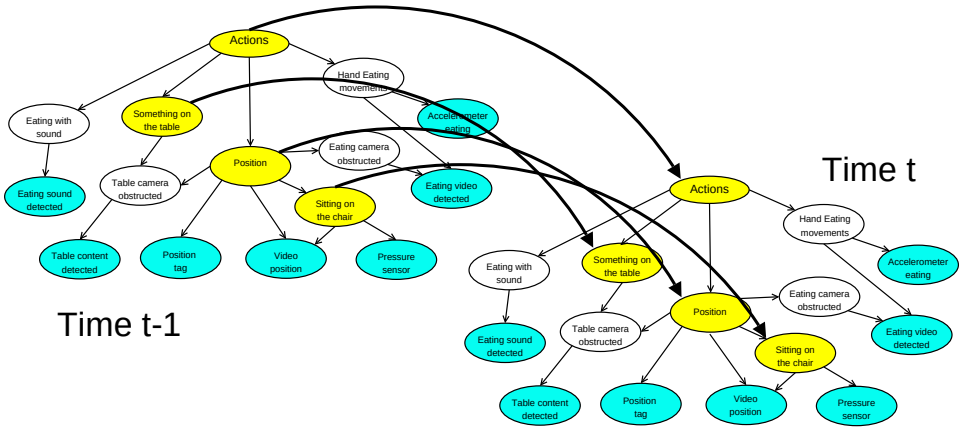

Figure 4.2: The Dynamic version of the Bayesian Network from the previous figure. Yellow nodes are temporal nodes. In this case, the timed nodes are Activity, Something on the table, Position and Sitting.

copy of the BN is built on the basis of the original static BN, the conditional probability distributions (CPD) for the temporal nodes are different, because they depend on all the past observations. On the other hand, the CPDs for the temporal nodes in the second copy of the BN are equal to the original CPDs. This change in CPDs of the temporal nodes represents all the dynamics of the system. On the other hand, the fact that the CPD for the temporal nodes is changing every time step actually represent the memory of all the past sensor readings embedded in the model. Even if one of the sensors give a new information at the next time step, it will not affect the distribution of the hypothesis state much, unless it comes from a very reliable sensor or correlates with other data.

Similar to the Bayesian network, usually the top nodes of the both copies of the BN represent a hypothesis Θ and leaf nodes represent sensor readings or features z_i. The only difference is that both Θ and z_i have different time indices in the two copies of the BN, e.g. $z_i^{(t-1)}$ and $z_i^{(t)}$. The estimation of the state in the past is done using evidence readings $\mathbf{Z}^{(t-1)}$ and prediction of the system state at the next time moment $\Theta^{(t)}$ is done by obtaining the distribution $Pr(\Theta^{(t)} = \theta_k)$. Since distribution at the time $t-1$ together with obtained evidence defines the distribution at the time moment t, we can use joint distribution $Pr(\Theta = \theta_k, z_1 = f_{j1}^1, ..., z_n = f_{jn}^n)$ to find which set of sensors would likely to give satisfactory information quality.

Figure 4.2 shows an example of the Dynamic Bayesian network corresponding to the Bayesian network from the Figure 4.1.

4.4.4 Information uncertainty metric

The uncertainty of a variable is represented by a probability distribution. However, in the case of the hypothesis of the multi-variable system, the probability distribution is the goal of the estimation algorithm. Selection of the sensors, however, has to be done in advance before the hypothesis distribution can be computed.

For this we are going to use the entropy based metrics. Entropy of a discrete random variable is defined as

$$H(\Theta) = -\sum_i Pr(\Theta = \theta_i) log_2(Pr(\Theta = \theta_i))$$

In the case when there is possible multitude of distributions depending on some set of variables such as sensor readings, the metric of the uncertainty of the distribution is the conditional entropy $H(\Theta|\mathbf{S})$ of the variable, which is the expected value of the entropy of the hypothesis distribution obtained with sensor readings as evidence, averaged over all possible sensor reading combinations.

$$H(\Theta|\mathbf{S}) = Pr(\mathbf{S} = S)[H(\Theta)|S]$$

There are two commonly used metrics. First, we can use the conditional entropy itself, which is computed as

$$H^{(t)}(\Theta|\mathbf{S}) = -\sum_{\Theta} \sum_{\mathbf{Z}^{(t)} \in F_1 \ldots F_n} P_t(\theta, z_1^{(t)}, \ldots z_n^{(t)}) \log P(\theta | z_1^{(t)}, \ldots z_n^{(t)}) \qquad (4.3)$$

The good sensor selection has to have a low value of the conditional entropy.

Another metric, similar to [ZJ06] is the *information gain* or *entropy reduction* and defined

$$I(\Theta; \mathbf{S}) = H(\Theta) - H(\Theta|\mathbf{S})$$

and can be computed as

$$I^{(t)}(\Theta; \mathbf{S}) = \sum_{\Theta} \sum_{\mathbf{Z}^{(t)} \in F_1 \ldots F_n} P_t(\theta, z_1^{(t)}, \ldots z_n^{(t)}) \log \frac{P(\theta | z_1^{(t)}, \ldots z_n^{(t)})}{P(\theta)} \qquad (4.4)$$

Instead of the entropy H, the normalized entropy $\overline{H}$ can be used, where $\overline{H}(\Theta) = H(\Theta)/log(|\theta|)$, $|\theta|$ is the number of hypothesis states.

Below, as the uncertainly metric in the sensor selection we are using the conditional (not normalized) entropy $H^{(t)}(\Theta|\mathbf{S})$

4.5 Single application case without resource constraints

First, we consider the simple case of a single (standalone) application using the sensor network with the aim of getting certain high-level information. Since the application is standalone, no other applications are competing for the shared resources and therefore the problem of resource management is the problem of utility maximization without resource constraints (assuming that any resource is able to accommodate at least this application, otherwise it is useless).

4.5.1 Optimization problem formulation

The optimization problem in general form is

Maximize: $U(\mathbf{R}, \mathbf{S}_t)$

Subject to: $No\ constraints$

The variables are $\mathbf{R}$ - the set of resources used, and $\mathbf{R}$ - the set of sensors selected. Strictly speaking, $\mathbf{R}$ is a function of set $\mathbf{R}$, network N, and protocols in the network such as routing or media access control. However, we are not exploring these dependencies here.

The most common case of the utility function is a sum of utility due to the information obtained minus the cost of resources used.

The optimization problem is therefore

Maximize: $U * H^{(t)}(\Theta|\mathbf{S}) - \sum_i C_i R_i$

Subject to: $No\ constraints$

Where U is the utility gain from the information acquisition (negative here since entropy has to be minimized) and C_i's are the costs of resources.

4.5.2 Sensor resource model

The choice of the sensor resource model is dictated by the fact that for the fusion we use features as the low-level data. Since features already represent high-level data, we assume that every sensor modality has a software which runs on a sensor node and consume node resources. Therefore for operation of this software we need the physical sensor to be available as well as certain amount of node resources such as memory.

If we assume the constant network structure and routing, then, when the sensor is selected, the tuple of resources

$$R_k = (CPU, Memory, Sensor, Energy, Network)$$

is selected. The fact that the resources are selected by a set does not change optimization problem formulation in the case of a single application since there is no resource constraints and we can just sum the cost of resources for this sensor and present is as a sensor cost.

Note, that in the case of a sensor attached to a remote node, a tuple of resources used can be quite long, as it will have to include not only resources of a node with a sensor, but also CPU, network and energy resources of all the nodes which forward the information. If on one of such nodes the amount of resources is insufficient, the entire tuple cannot be allocated and the sensor should not be available.

4.6 Sensor selection

4.6.1 Applicability of the Bayesian network model

We can evaluate the conditional entropy for the sensor set using both BN as well as DBN model. However, for several reasons, Bayesian network model is not very suitable for the dynamic phenomena-aware resource management. First, since there is no notion of time in the Bayesian network, the estimation of the state Θ is only valid for the moment when the sensor readings were taken. Second, many sensor readings has to be taken at the same time to provide satisfactory IQ because sensor readings from the previous time moments cannot be reused. Third, Bayesian network does not provide a way to estimate system state in the future time moment and therefore does not allow predictive sensor selection.

Therefore the dynamic adaptation of the selected sensors which can be done using Bayesian network include reaction to the changes in application IQ requirements and the change in the set of avaiable sensor modalities. Such an adaptation was realized in the Midfusion [AKS04], with no adaptation to the phenomena state.

Below, we describe the process of data fusion combined with sensor selection based on the Dynamic Bayesian network model and explain how it can address the limitations stated above.

4.6.2 Sensor selection using Dynamic Bayesian network

The metric of IQ we are considering here is the certainty of the hypothesis, that is the $Pr(\Theta = \theta_k|\mathbf{Z}^{(1)}, ...\mathbf{Z}^{(t)})$. We can only estimate the actual certainty when we have a measured data. To evaluate certainty of the result in the

future we have to use another metric which would give us the estimation of certainty of the result for the whole range of the possible measured data values. In this case such a metric is the expected entropy of the result given the set of sensors used.

Now assume that there is a fusion system with the goal of tracking the state of a certain phenomena. The phenomena state can be derived from the sensors and we want, depending on the state of the phenomena being tracked, use only subset of the sensors.

The phenomena tracking is done in the following way. Below we present one cycle of the acquisition, estimation and selection process. In this process we assume that the data is acquired from all the sensors in parallel.

1 Acquire sensor readings according to the set $\mathtt{S}_{t-1}$

2 Update hypothesis distribution using acquired data $Pr(\Theta|\mathtt{Z}^{(t-1)})$

3 Update joint distribution for the next step $Pr(\Theta^{(t)}, s_1, ..., s_n)$

4 Select subset of sensors $\mathtt{S}_t$ for the next step so that entropy $H^{(t)}(\Theta) < H_{threshold}$

5 Update joint distribution for temporal nodes for the next step $Pr(\theta^{(t)}, X_{\tau 1}, ..., X_{\tau m})$

Primary objective of the sensor selection algorithm is therefore to choose sensors which would give the expected entropy of the state hypothesis below certain threshold. Please note, that, unlike the case of the static Bayesian network, in the dynamic case the data obtained previously is still useful, because it affects the joint probability distribution $Pr(\Theta = \theta_k, z_1 = f_{j1}^1, ..., z_n = f_{jn}^n)$. Because of this, the number of sensors required to satisfy the application requirements can be low or even no sensor readings may be required.

4.6.3 Addressing Confidence: Choice of threshold

We can address the *confidence* of estimation of most likely state by setting specific entropy threshold. The more details on confidence is included in Section 2.2.2.

If we want to enforce specific confidence $q = P(x_{max}|\mathbf{Z}) < 0.5$, we can select the threshold level to be

$$H_{threshold} = min\{H|P(x_{max}) = q\} = -(q\ log_2(q) + (1-q)\ log_2(1-q))$$

4.6.4 Addressing Coherence: Sensor Selection in the case of high certainty

However, the memory effect of the DBN model has a drawback, too. If the entropy obtained during previous time step is already quite low, then, depending on the transition probabilities between hypothesis states, it may stay low for several time steps even if no new data is obtained. Although it does allow to save on the resource use, this effect carries a risk that we may miss the change in the system state. Since, in this case, the certainty is already high, we do not need to improve it by selecting sensors which would give low expected entropy for all the system states. Instead, we would like to focus on the possibility of the change in the state.

Essentially, we would like to enforce the *information coherence*, which is described in Section 2.2.1. The interesting observation related to this is the fact that, according to our experience, if we want to obtain reliable result when monitoring some phenomena, we need to address at least two metrics, one of them of acquisition completeness type (Section 2.2.1) and another of acquisition uncertainty type (Section 2.2.2).

In this case, we can use the metric which expresses expected entropy of the result for the actual phenomena states different from the current most likely state. We propose to use expected entropy conditional on the actual system state.

It is computed similar to the expected entropy for the sensor set in the Equation 4.3, except that the probability of a particular sensor reading set is conditional on the actual system state.

$$H^{(t)}(\Theta|\mathbf{S},\omega_i) = \sum_{\mathbf{Z}^{(t)}} Pr(z_1^{(t)},...z_n^{(t)}|\omega_i)\times$$

$$\times \sum_{\Theta\in F_1...F_n} Pr_t(\theta|z_1^{(t)},...z_n^{(t)}) \log Pr(\theta|z_1^{(t)},...z_n^{(t)}) \tag{4.5}$$

The distribution $Pr(z_1^{(t)},...z_n^{(t)}|\omega_i)$ is computed using the static Bayesian network. This is because in particular static network correctly describes distribution of the readings for a particular actual state, whereas Dynamic Bayesian network describes our belief about the readings distribution based on the current estimated system state.

To combine entropies for the different actual states, we propose to compute their weighted sum according to the estimated state likelihood at the next time step.

$$\widehat{H}^{(t)}(\Theta|\mathbf{S},\Omega=\theta_{max}) =$$

$$= \sum_{\omega \neq \theta_{max}} \frac{Pr(\theta = \omega_i)}{1 - Pr(\theta = \theta_{max})} H^{(t)}(\Theta|\mathbf{S}, \omega_i) \tag{4.6}$$

Therefore the step 4 of the phenomena tracking can be modified:

`4 If` $H^{(t)}(\Theta) > H_{threshold}$`, then Select subset of sensors` S_t `for the next step so that` $H^{(t)}(\Theta) < H_{threshold}$

`else Select subset of sensors` S_t `for the next step so that` $\widehat{H}^{(t)}(\theta_{max}) > \widehat{H}_{threshold}$

Please notice that the threshold $\widehat{H}_{threshold}$ is different from $H_{threshold}$. Moreover, the fact that the the value of $\widehat{H}^{(t)}(\theta_{max})$ should be large is counter-intuitive. Usually we may want to minimize the entropy. However, this is not the case when we are looking for a change - in this case the most useful sensors are those which would eliminate our belief that the actual state is θ_{max}. And in this case we need to pass through a state of low certanty.

In more simple case we can select not a set, but only one sensor, which would give us the highest $\widehat{H}^{(t)}(\theta_{max})$. This is in fact an algorithm used in a testbed implementation described in Section 4.9.

4.6.5 Sensor selection with losses

If we have losses in the system, due to network or computational overload, then not all of the measured data will be used in the final result. We can estimate the impact of the losses on the expected entropy If we denote as σ the subset of $\mathbf{S}$, including the measurements which are not lost and are actually used in the final result, then the expected information gain in a system with losses is

$$J^{(t)}(\Theta|\mathbf{S}) = \sum_{\sigma \subseteq \mathbf{S}} H^{(t)}(\Theta|\sigma) Pr(\sigma) \tag{4.7}$$

where $Pr(\sigma)$ is the probability that the set of readings not lost is equal to the subset σ.

If we assume that the data from different sensors is lost independently and loss probability of data from a sensor i is q_i, then, denoting

$$Q_i(\sigma) = \begin{cases} 1 - q_i & if \quad i \in \sigma \\ q_i & if \quad i \ni \sigma \end{cases}$$

we get

$$J^{(t)}(\Theta|\mathbf{S}) = \sum_{\sigma \subseteq \mathbf{S}} H^{(t)}(\Theta|\sigma) \prod_i Q_i(\sigma) \tag{4.8}$$

For approximation, we assume that the information gain from any subset of the set $\mathbf{S}$ is less than the one of the $\mathbf{S}$ itself. This assumption is reasonable

since we optimize the resource cost, and the optimization algorithm would have chosen the subset instead of the set $\mathbf{S}$ if it would give better quality.

Equation 4.8 contains a sum of $n!$ elements, n - number of sensors in $\mathbf{S}$. However, it can be approximated by including to the sum only those elements which have sum of probabilities sufficiently close to 1. Depending on the loss probabilities, this sum can contain only one element, $n+1$ elements including losses of no more than one reading, or $n_2 + n + 1$ element including losses of no more than two readings.

In the first case, it is

$$J^{(t)}(\Theta|\mathbf{S}) = H^{(t)}(\Theta|\mathbf{S}) \prod_i (1 - q_i)$$

with the error being bounded by $H^{(t)}(\Theta|\mathbf{S})(1 - \prod_i (1 - q_i))$

For larger loss probabilities we can have an approximation including single losses

$$J^{(t)}(\Theta|\mathbf{S}) = H^{(t)}(\Theta|\mathbf{S}) \prod_i (1 - q_i) + \sum_i I^{(t)}(\Theta|\mathbf{S} - i) q_i \prod_{j \neq i} (1 - q_j)$$

with the error bound $H^{(t)}(\Theta|\mathbf{S})(1 - \prod_i (1 - q_i) - \sum_i q_i \prod_{j \neq i} (1 - q_j))$
or similar sum for double losses.

4.6.6 Sensor selection with slow sensor modality

However, in the case of the resource limitations there may be a case when a rate of a certain sensor modality is limited due to the complexity of the feature extraction algorithm. In this case we may have to use the modality not every cycle of the estimation of the system state, but only for the subset of cycles. These cycles can be either asynchronous or synchronized with the rest of the sensors.

The first case can be modelled by changing the probability that the sensor reading is used in the final result to reflect the absence of the reading due to low rate. That is, the total probability of delivery will be decreased by the ratio of modality rate to the total rate.

The second case is more complex as the sets of sensor data are no more arbitrary. In this case, knowing that the readings from the particular sensor will be missed, we can recover the IQ loss by adding other sensors.

Effectively, it means that we create separate sets of sensors and schedule them to be fused in different time moments. Therefore, in assessing the expected information gain from the two-set configuration, we need to consider the delivered result from both of them as

$$J^{(t)}(\Theta|\mathbf{S}) = J^{(t)}(\Theta|\mathbf{S}_2)\rho_1 + J^{(t)}_{1+2}(\Theta|\mathbf{S}_2)\rho_2 \tag{4.9}$$

where ρ_1 and ρ_2 are the relative frequencies of the sets of sensors

4.7 Multiple applications with resource constraints

When we have multiple applications sharing the same resources, we have to account for the fact that the different application bring different benefits to the system as well as for the fact that some bottleneck resources may not be always available.

The problem of providing the IQ in the resource-constraint environment can be formulated as following using a sum of utilities:

$$\text{Maximize } U = \textstyle\sum_j U_j(y_{ij}(t))$$
$$\text{Subject to } \begin{cases} y_{ij}(t) \in 0, 1 \\ \sum_{ij} \mathbf{R}_{ij} y_{ij} \leq \mathbf{r}^{max} \end{cases}$$

Application j has a utility $U_j(y_{ij}(t))$ depending on the selection of individual resources $y_{ij}(t)$. Instead of sensor selection $\mathbf{S}$ such as in section 4.5.2, we used individual sensor selections y_{ij}, $\mathbf{R}_{ij}$ is a tuple of resources, where each R_{ijk} is the amount of resource k used by application j when i'th sensor is selected. Here, we used the linear model of resource use, although it may not always be the case. However, we need this linear model for the decomposition of the optimization problem.

We propose to use the dual decomposition [CLCD07] to address the complexity of the multi-variable optimization problem.

The relaxed problem is

$$\text{Maximize } U = \textstyle\sum_j U_j(y_{ij}(t)) - \lambda^T(\sum_{ij} \mathbf{R}_{ij} y_{ij} - \mathbf{r}^{max})$$
$$\text{Subject to } y_{ij}(t) \in 0, 1$$

And optimization problem is decomposed into two layers of optimization. On the lower layer the application-level optimization is solved similar to the optimization in section 4.5.1. The upper layer optimization problem solves the problem of assigning the resource costs so that the global utility can be maximized.

The respective problems are:

For the lower layer

Maximize $U_j(y_{ij}(t)) - \lambda^T(\sum_j \mathbf{R}_{ij} y_{ij})$
Subject to $y_{ij}(t) \in 0, 1$

and for the upper layer

Maximize $U = \sum_j G_j(\lambda) - \lambda^T \mathbf{r}^{max}$
Subject to $lambda \geq 0$

Where $G_j(\lambda)$ is the solution of the lower layer problem of the application j.

The fact that we used the resource costs here does not mean that the actual resource cost cannot be added to the utility functions. In this case, we would have two types of costs - one which is used in resource management and one that affects the total utility of the system.

This decomposition has a benefit that it uses the result of the application level optimization and is simple. Unfortunately, such a dual decomposition is iterative and does not guarantee the satisfaction of the resource constraints during iterations.

4.8 Simulation evaluation

We created a simulation to evaluate the performance of the fusion system with phenomena-aware resource management. The simulation based on the ayes Net toolbox [Bay], and compares the performance of the BN and DBN based systems. We should note that the BN based system cannot actually be implemented in a real system since it cannot actually select resources in advance.

4.8.1 Simulation setup

We used Bayes Net toolbox [Bay] to evaluate the use of the Bayesian Networks as well as Dynamic BN for the phenomena-aware resource management. The scenario we used was the observation of eating habits of a dementia patient in the home environment. The task of the observation was to track the time when a person was eating in the kitchen. For both fusion and uncertainty estimation the we used the networks shown correspondingly in Figures 4.1 and 4.2.

In our model, we want to detect when person is eating while sitting at the table. The sensor modalities used are video (position, table content, eating

movements), RFID tag (position), audio (eating sound), pressure (sitting), accelerometer (eating movements). There are three states of the *Action* to be detected: "No person", "Person in the kitchen" and "Person Eating". Note, that in our model it is impossible to have a direct transition from the state "No person" to "Person Eating". In the case of DBN, the states which introduce the system dynamics and therefore has to be considered as temporal are *Action*, *Position*, *Sitting on the chair* and *Something on the table*. The reason for this choice is that these states affect the detection of eating yet represent system dynamics which cannot be affected by the sensor choice. For example, even if the person is sitting at the table but there is nothing on it, most probably he is not eating and accelerometer readings are not useful.

We set the optimization parameters for both cases so that the system is required to have a certainty of current activity recognition certainty to be at least 80%. If the certainty is above these 80%, the resource cost is minimized. If the certainty is below 80%, the system has to select a set of sensors which would give the best certainty possible.

Both simulated examples used the same scenario. Person appears in the kitchen, wanders around for a while, coming in, out, to the table etc. Then at some point he sits at the table, starts eating after a while, then, after eating, removes everything from the table and leaves the kitchen. In both cases we used exactly the same sequence of events. Also, in both cases we used the same set of sensor readings, which were generated using our knowledge of scenario and static Bayesian network model to represent the imperfection in the sensing.

These sensor readings were used in the recognition of the activity using full set of sensors or using reduced set chosen by the phenomena-aware resource management. Here we present four results - activity recognition using all the sensors with BN model, all sensors with DBN model, reduced sensor set with BN model and reduced set with DBN model.

In our simulation, each sensor modality had its own cost reflecting relative resource use. For example, the cost of video positioning was 10 times more than the cost of pressure sensor on a chair. We used these costs to select more cost-efficient sets of sensors.

Although we used both static and dynamic Bayesian networks, in fact the static BN was not properly used for the sensor selection. This is because of the absence of the phenomena state dynamics in the BN model as explained in section 4.6.1. Therefore we used post-factum sensor selection, when we used the data from all the sensors to select a subset which would give a good uncertainty bound for the phenomena state. Such a selection is equal to the case when we have ideal algorithm for sensor selection which is able to select

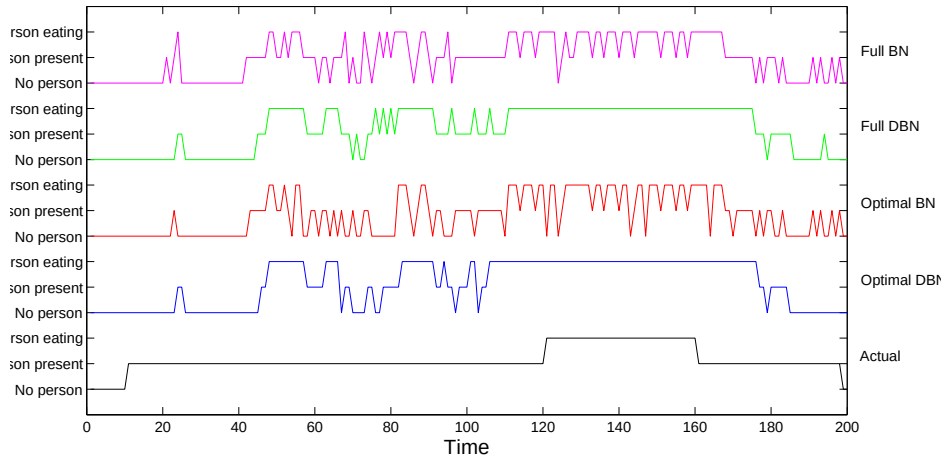

Figure 4.3: Simulation results. The comparison of the actual state of the system with the estimated state derived from corresponding models. The problem of the BN model in this case - high volatility of the state estimation

a set of sensors giving optimal cost. Although it is not realistic case, we still used it to compare this ideal sensor selection using BN model with realistic sensor selection using DBN model.

4.8.2 Simulation results

The compared results are presented on the Figures 4.3, 4.4 and 4.5. As it can be seen on Figure.4.4, the moments when the certainty of activity recognition fall below the required 80% level for the case of the DBN correspond to lower than required level of the certainty in the case of the BN model. On the contrary, low certainty for the BN model does no mean that the DBN model also would give a low certainty result. However, the phenomena-aware model based on the DBN used, on the average, much less resources. This is mainly due to the fact that the DBN model relies significantly on the readings obtained in the past. The stronger drop in the certainty for the case of the DBN is due to the fact that, unlike BN model which used post-factum selection, DBN model has to detect the state change first and in the case of the few sensors being on the initial recognition certainty can be low. After the system chose the set of sensors giving best information quality at the next step, the certainty significantly increased.

Another comment concerns the lack of memory in the BN model. In

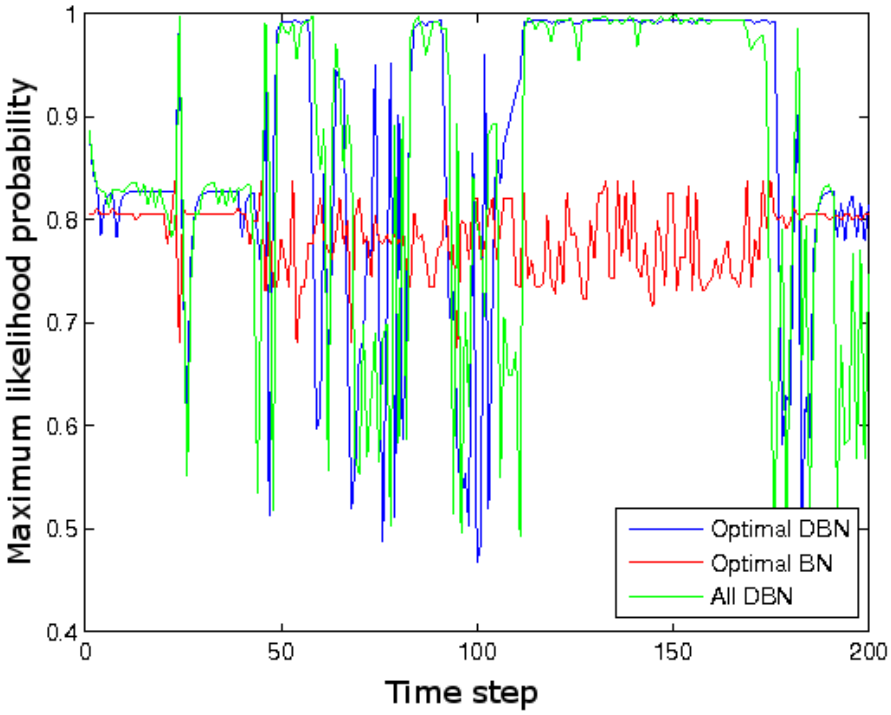

Figure 4.4: Simulation results. Certainty comparison for different models and different set of sensors. As it can be seen, use of reduced set of sensors for the Dynamic Bayesian network does not significantly affect the certainty of the result.

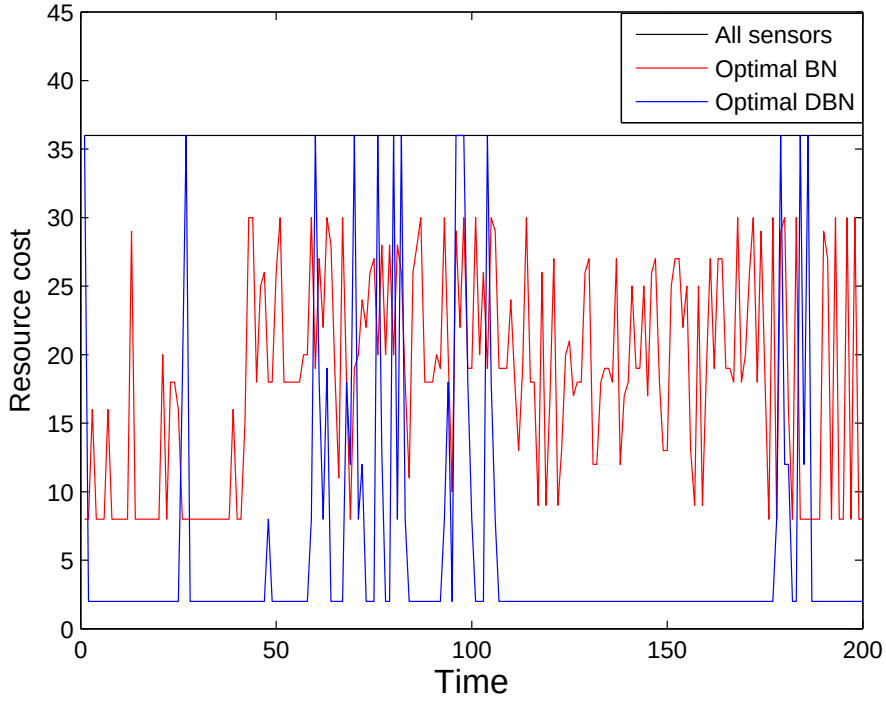

Figure 4.5: Simulation results. The comparison of the cost of sensors to achieve a required level of the information quality using phenomena-aware resource management. It can be seen, that the memory property of the Dynamic Bayesian network model allows to obtain a good quality at the fraction of a cost.

our model, there are some states of the Action and Position which prohibit immediate transitions from one to another. For example, Action state *Person Eating* can not be immediately after Action state *No person*, but only after passing through intermediate state *Person in the kitchen*. However, in the case of the BN model, due to absence of the connection to the past readings, the estimated state can change between these two states very often. This problem is much less pronounced in the case of the DBN model.

As it can be seen on the Figure 4.3, the estimated state in the case of the BN is much more volatile. It even has several changes between two states which cannot do not have a valid transition - the states "No person" and "Person Eating". This is due to presence of the memory of the past in the DBN model. This memory effect also allows to save on the resource use.

4.9 Testbed experimental implementation

In this section we present live implementation of an online activity recognition system which uses the DBN model for inferencing of the persons activity with the aim to identify if he is eating. Although in our evaluation we only used four types of sensors, the application of this framework is not limited to only these sensors. In fact virtually any sensor which processes information and produces simplified description of what was detected, can also be used. Besides, the system allows testing of different heuristics of the real-time sensor selection based on the DBN entropy-based metrics.

4.9.1 Phenomena monitored

The monitoring scenario is based on a real-world need to monitor the activity of the elderly people at home to assess their ability of independent living. The systems similar to this should provide information about persons habits and abilities as well as serve as a first level of help in case of emergencies. The performance of a typical system should not only allow post-factum understanding of actions a person took during the day, but also current moment understanding of what the person is doing so that help in a form of either information / reminders or in a form of human care-giver involvement can be despatched. Therefore the system should be able to operate in a time frame of a typical human action duration.

4.9.2 Hardware configuration

Currently, the prototype uses four sensor modalities.

1. Wrist-worn accelerometer to detect the hand movements. A sensor platform iMote2 from Crossbow [Cro] was used together with data acquisition board holding 3-axis accelerometer. The output of a sensor is acceleration $\pm 2g$ presented by discrete values of ± 2048 for each axis. This is the only sensor in the current setup which uses non-trivial features obtained from a very large space of possible values. Essentially, in this setup the features provide information about approximate angle of a hand with respect to gravity. However, even this information can be useful, because in fact many important hand motions have very narrow range of such angles.

2. We installed the short-range RFID reader under the table surface to detect whether the cup (tagged with RFID tag) is being used

3. Since the scenario assumed that we want to detect eating when the person is sitting, we installed the pressure sensors to detect if the person is sitting

4. Three PIR sensors in the environment were used to detect the position of the person. One sensor was positioned near the entrance, one near the table and one - under the table facing persons legs to provide alternative way to detect if the person is sitting.

All sensors except RFID reader were connected to the wireless sensor nodes.

4.9.3 Software configuration

The configured system is able to do estimation of a person eating activity approximately 10 times per second, the rate sufficient to detect movements of a person around the house as well as local movements such as standing up / sitting or raising an arm. The prototype uses simple scenario with only a few actions detected. However, the strong point is that this system is able to do both activity recognition as well as resource management in the real time and thus gave us a good system which can be used in more complex real-life experiments as well as provide insights into what are the performance issues can be faced when implementing such system for different application

The activity states detected in this system are *Person Absent*, *Person Present*, *Person Sitting*, *Person Eating*, *Person Drinking*. Since the states of the hypothesis mode has to be disjoint but states of the actual system are not, the state *Person Sitting* actually mean that the person is present, sitting, but not eating or drinking. Likewise, the states *Person Eating* and

Figure 4.6: Illustrations of the activity detection testbed. Wrist-worn accelerometer was used for hand movement detection

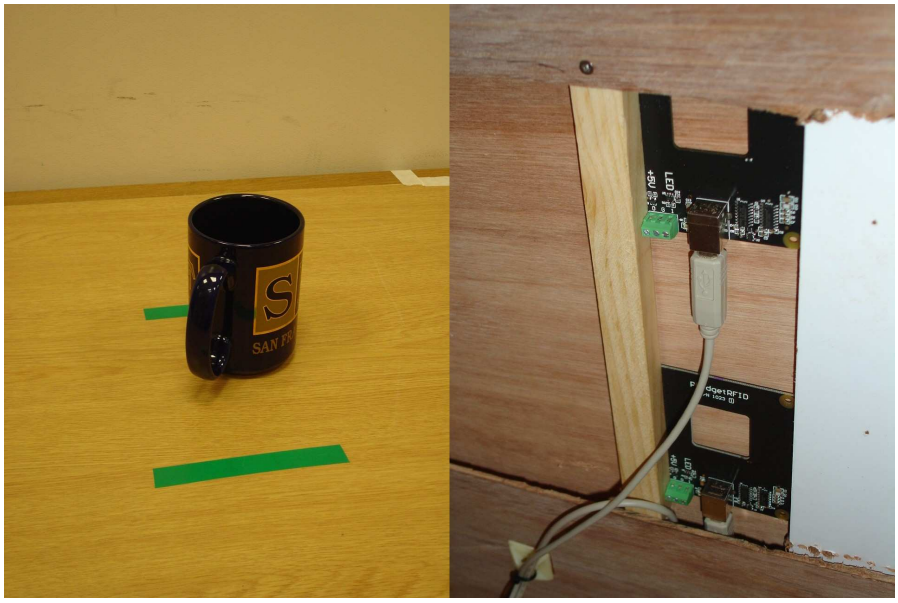

Figure 4.7: Illustrations of the activity detection testbed. Short-range RFID reader was used for detection of the object (cup) being used

Figure 4.8: Illustrations of the activity detection testbed. Pressure sensors installed in the pad on the chair were used to detect if a person is sitting

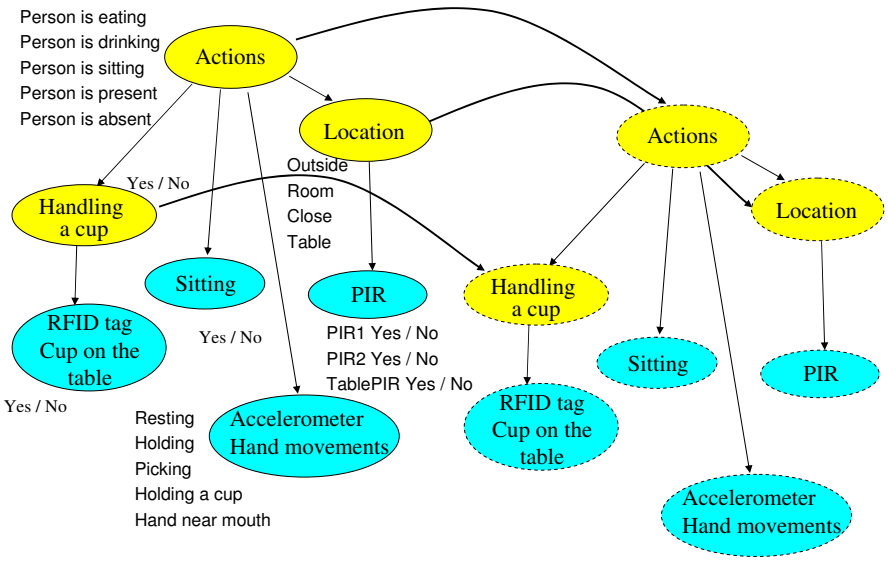

Figure 4.9: The DBN of an activity detection system, which was implemented on a testbed. The possible states of variables are shown next to corresponding nodes

Person Drinking mean that the person is present, sitting and either eating or drinking. Figure 4.9 shows the structure of the DBN as well as the possible states of variables.

We used the Microsoft Belief Networks toolkit [MSB] to do the BN inferencing.

We used the recorded video of the activity for manual annotation of the training data. All the intermediate states of the DBN were also taken from this video. After that the data was used to compute all the conditional probability tables for the DBN.

Figure 4.10 shows the comparison of the actual state and the state detected by the recognizer. The long vertical lines on the graph correspond to the moments showed by a snapshots from the video showed on the Figure 4.11. Figure 4.12 shows the confidence of the detection of activity. The low confidence of a recognition happens mainly during the change of activity.

4.9.4 Observations

There are three observations which we made while using the testbed implementation

1. From our experience, it is now enough to enforce only one metric of

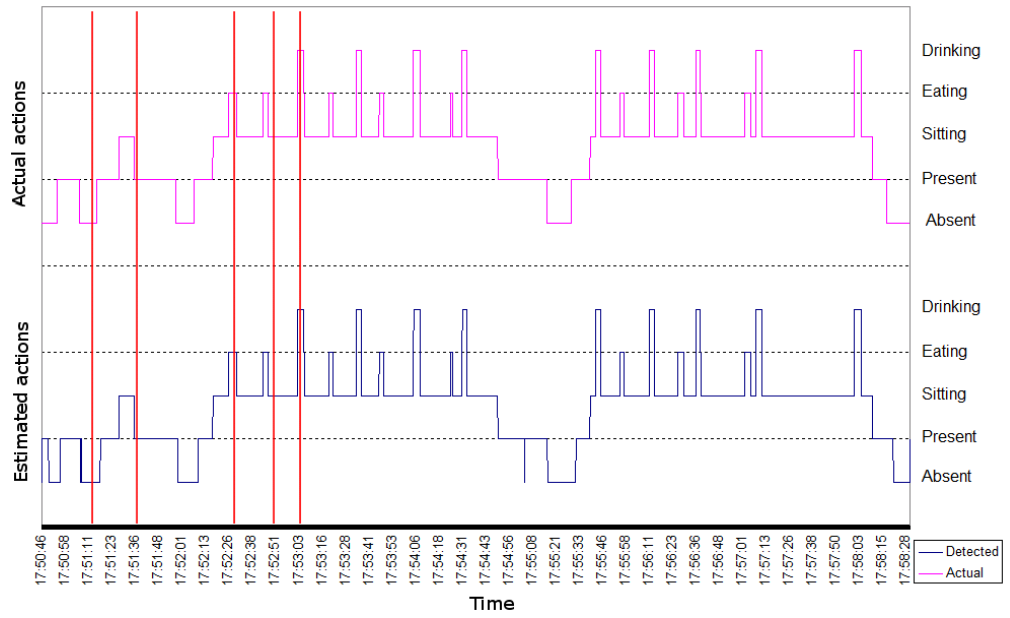

Figure 4.10: Activity detection testbed results. Correctness of the online activity recognition. The top graph shows the actual activity of a person. The lower graph shows the activity detected by a system. The long vertical lines correspond to the moments shown on the Figure 4.11

Figure 4.11: Activity detection testbed results. The fragments of video recording corresponding to the long vertical lines in the Figure 4.10

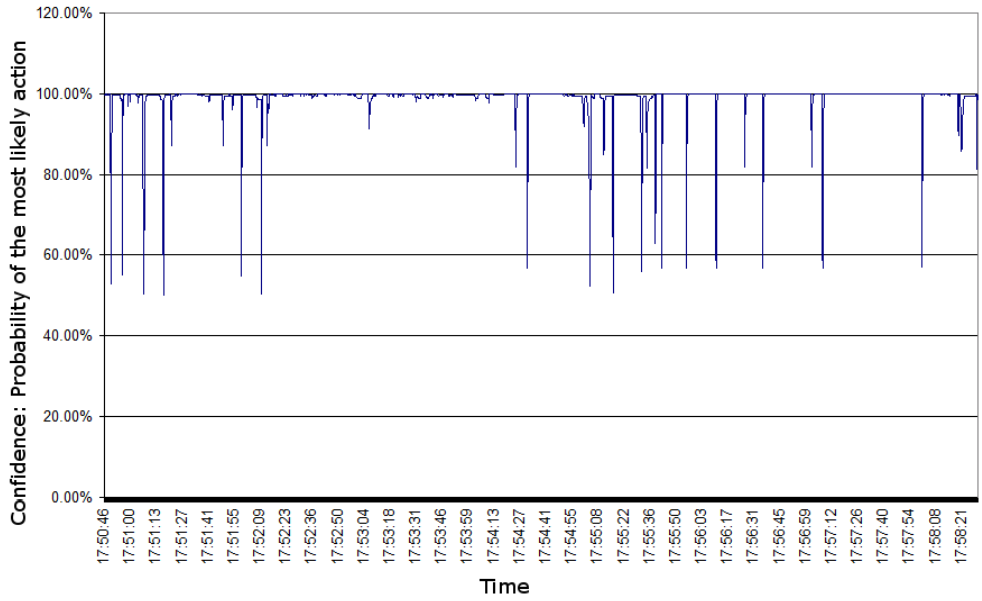

Figure 4.12: Activity detection testbed results. Confidence level of the online activity recognition.

IQ. To make the system follow the phenomena reliably, we need not only enforce uncertainty metric such as confidence of state estimation, but some completeness metric as well. In our case, the easiest metric to enforce was information coherence.

2. The benefit of a model in comparison with rule-based system is that model may implicitly contain certain rules, which may not be discovered by a simple observation. For example, when one sensor high certainty selection algorithm was used described in Section 4.6.4, we noticed that when a person was sitting on a chair with both hands resting on the table, the pressure sensor was activated very rarely. That is, the system did not expect the state to change from "Person Sitting" to simple "Person Present" i.e. not sitting. It turned out that normally a person does not stand up from the chair holding hands in front. Most probably the person will change the hand position, which will be detected by accelerometer. And since eating and drinking is also detected by accelerometer, it is the most useful sensor for detection of state change at the moment. That is, the model implicitly contained a rule "Person has to move hands to stand up" which may not be included if the rules were created through an observation.

3. The drawback if a model is in the fact that the model should be based on as large dataset as possible, so that rare combinations of node states would not be considered impossible, thus making the state estimation incorrect or impossible.

4.10 Conclusion and future work

In this chapter, we proposed a framework which describes how applications using sensor network data can be provided with certain information quality assurance. The framework uses Dynamic Bayesian Network model.

The original contribution described in the chapter are:

- Two IQ metrics were addressed: confidence of state estimation and information coherence. Confidence of state estimation bring together IQ loss from both acquisition-based uncertainty as well as delivery-based uncertainty. Information coherence metric here addresses only acquisition completeness class of losses.
- Delivery-based uncertainty losses were addressed by suggesting a way to account for decreased quality of final fusion result after expected losses
- Somewhat counter-intuitive way to address information coherence was proposed, based on expected entropy under assumption of a change in phenomena state
- Another important contribution, described in this chapter, is that this framework was implemented on a real testbed and similar systems now can be easily developed.

In comparison with other model-based systems, this framework has following advantages:

- We defined thoroughly the metrics of Information Quality and suggested a generic approach for defining IQ metrics for different scenarios
- We use general notion of phenomena being monitored by a sensor network, which goes beyond the particular types of phenomena such as target localization and tracking in [ZLL+03]
- For this general notion of phenomena, we change resource allocation based on the dynamic phenomena state to provide the required IQ, in contrast with average phenomena state in [AKS04]

- We address the issues of the data losses and resource constraints which are essential features of sensor network infrastructure in comparison with [ZJ06].

In comparison of our model-based approach with arbitrary rule-based system we would like to to emphasize the following aspect. If a rule set is reach enough, the discrete-state phenomena tracking can be done with high level of estimation correctness. Similarly, for such a system, the reach set of rules can be defined to do the sensor management. However, the DBN-model based approach has two strong points here. First, the probability representation of the hypothesis state can give us the uncertainty measure of the final result, that is it can provide the system with IQ metric. Second, the fusion rules as well as sensor selection rules in the DBN case are discovered automatically as long as the DBN model is built.

The important advantage of the described system is that the sensor data as well as system states are generic. Even the assumption of the discrete values is reasonable because in many cases the higher-level applications are not interested in raw continuous values. For example, from point of view of analysis of a person behavior at home, the actual position is not very important, more important is the proximity to certain household objects, which is already discrete information.

However, there are serious weaknesses in the current framework. The main weakness is the fact that the optimization problem described above performs exhaustive search of optimal subset of a set of sensors, and since all the possible subsets may be tested, it is NP hard in the number of sensor. Currently no heuristics are proposed. Another problem is that obtaining the exact values of conditional entropy or information gain becomes very computationally difficult when the width of the Bayesian network increases. Therefore there is a need for an algorithm which would use approximate values of information gain yet would give a good IQ. These concerns has to be addressed in the future and implemented on the online activity detection testbed.

Another direction of a future work is further development of the application decomposition to allow the resilient use of shared resources by several applications. The important problem to solve is how to avoid the resource constraint violation without excessive information exchange.

Chapter 5

Cyclic computation deadline

In section 1.4 we briefly introduced loosely coupled distributed systems, and listed the main challenges in providing with QoS for applications in such environments are that:

- It is dynamic. Resources and applications are added and removed from the system unpredictably.
- It is highly heterogeneous. It consists of many types of systems so that it may not be even possible to enlist and characterize all of them precisely.
- It is complex in structure. It may consist of many components and interaction between them may be too complex to trace.
- Has limited coordination between resource subsystems executing different tasks.

However, for the provision of the QoS the resource use has to be controlled to prevent certain applications from starving others. Here, we present distributed scheme of the resource load control in loosely coupled distributed system which does not require complex coordination between different resource domains yet achieves guaranteed resource allocation to different applications. Essentially this is a distributed application-level regulator, which uses Timed Petri Net model to select tasks to be delayed from execution and a duration of the delays.

5.1 Quality of service in loosely coupled distributed systems

In this section we will highlight specific features of the loosely coupled environment which affect the way quality of service could be provided. In particular, we will explain why traditional methods to achieve QoS guarantees may not be suitable for such environments and what is being done and could be done to overcome these difficulties.

5.1.1 Specifics of loosely coupled distributed systems

As explained in section 1.4, the primary characteristics of a loosely coupled distributed system is that it

- is dynamic
- is highly heterogeneous
- is complex in structure
- has limited coordination between resource subsystems.

The best level of QoS guarantees is provided in the area of real-time systems. In real-time systems, there are two types of tasks: guaranteed and non-guaranteed. The guaranteed tasks are carefully admitted to a system using complex admission procedure. Resources on which tasks are executed are chosen in advance. The performance of these tasks is carefully measured using these resources. The complete set of tasks is checked to ensure that each task will complete within predefined time limits regardless of conditions caused by other tasks. No tasks are accepted after the system is validated. The non-guaranteed tasks are given lower priority than guaranteed and therefore will not affect the timing of the guaranteed tasks.

However, in loosely coupled systems, the set of tasks assigned to one resource is changing. Moreover, since resources are chosen dynamically, it may not be possible to predict correctly the performance of a task on the assigned resource.

Another type of systems in which QoS guarantees are provided are networks. Networks always perform only one type of a task. This task is the data transmission over a series of network elements. If we look at this task from the perspective of the application structure, we can see that it may be represented as a chain consisting of many elementary operations. Each of these elementary operations is forwarding of a data packet on a router. The

input of one operation is the output of the previous; the operations are the same and each operation is performed by a different router. However, in the case of an application running loosely coupled system, the structure of the application could be very complex. Besides, application components may be different in nature, they can run on completely different types of resources with different timings and each resource may execute many components of the application.

In traditional distributed systems a distributed application is run on different types of resources. However, it is still assumed that there is a limited set of resources with well known characteristics and that an application-level agent may obtain certain level of control over resources. Besides, the control channel can be available and be fast compared to the speed of the typical process. Therefore the agent has centralized control over the application execution. This centralized control makes application level QoS easily achievable because the agent knows the application QoS requirements and can modify the application execution on the fly to satisfy these requirements.

5.1.2 Existing approaches to providing QoS in loosely coupled distributed systems

The closest example of computation QoS control is the QoS provided in computational Grid environment [FK99]. Mainly systems there try to address the decoupling between the application and resource control. For example, to address the multitude of resources and application, [FKL+99] uses advanced resource reservation to ensure application can get it when the time comes. The implicit assumption was that either a resource would remain in the same state throughout application execution or that resource domain would substitute resources in the case when some of them become unavailable. Later this architecture was improved [FRS00] to include adaptation in resource reservation to ensure the required execution parameters in the case when performance estimation was made incorrectly. This improvement actually addressed the problem of lack of knowledge about resources in Grids. Another improvement was described in [RFG+00]. It addresses the problem of resource heterogeneity by providing extension to the MPI [MPI] interface. This extension can make a network reservation for a newly created MPI communicator. Therefore an application developer can implement reservation of different communication resources without knowing their exact characteristics and reservation protocol specifics.

However, the approach of this team has some weaknesses from the perspective of dynamic nature of loosely coupled environment and its complexity.

First of all, since they are using advanced reservation they assume that the resources will be available in any case. However, it may happen that the advanced reservation is not quite adequate for the required level of performance. They try to overcome the problem of inadequate reservation by introducing the possibility to adjust a reservation on the fly. However, unless application has exclusive control over resources it may happen that it is not possible to allocate more resources. Besides, in the case of transient disruption of a service there is no way to adapt to it in a flexible manner because resources are assumed to be allocated for the whole duration of the application execution.

The second weakness lies in the following: the fact that resources are reserved does not mean that the application would obtain the required level of service. The reason behind this is that a resource is able to provide a specific level of service only if the demand does not exceed certain level. Therefore the resource demand have to be regulated. This demand regulation includes both regulation of resource demand from other applications and demand from the application itself. The latter is as important as the former because the execution of one application component influences the service obtained by the other and this influence should be considered.

Another team, lead by Nahrstedt from University of Illinois at Urbana-Champaign, developed a scheme that considers resources as service components and application as a complex interconnection of these service components [XNVW00, GN02]. These service components are actually black boxes that can provide specific output quality of service parameters given that input parameters are of a certain level or better and a necessary amount of resources is allocated. This approach makes possible to account for complex structure of the loosely coupled system and to build applications with predefined End-to-end Quality of Service characteristics. In addition, depending on the way the application is built out of separate components, it is possible to consider some effects of dynamic system nature. In particular [XNVW00] considered the dynamic user arrival-departures and resource contention that may be caused by this arrival. Another work [GN02] considered the possibility of dynamic application composition depending on the availability of resources.

However, the approach of Nahrstedt team has one drawback. It uses measurement-based data to evaluate the amount of resources that should be allocated within service component to provide required output QoS given specific input QoS. However, measurement based allocations are vulnerable to fluctuations in resource performance. Therefore there could be a transient violation in the quality of service provided to a specific application component and this violation may affect the application end-to-end quality of service.

Therefore there is a need to provide an application in a loosely coupled

distributed system with a scheme that would ensure that the application end-to-end quality of service is satisfied in the presence of other applications and when application components are assigned in arbitrary manner to different resources. This scheme should address the dynamic and unpredictable nature of such environment.

Below we present the proposed research direction and explain why a particular technique was chosen to create the above mentioned QoS provision scheme.

5.1.3 Proposed technique

To provide an application with specific end-to-end quality of service in loosely coupled distributed systems, we need to answer following questions:

1. How to choose resources which are to be used by the application?
2. How to map application components onto resources so that the level of service provided to the application components can possibly satisfy the application end-to-end QoS requirements?
3. How to enforce the QoS obtained by application components on assigned resources so that end-to-end application QoS is satisfied?

The research described in this thesis focused on the third point of the list, namely on guaranteeing that QoS obtained by an application components will satisfy the application end-to-end QoS requirements. The first two questions were left outside of the scope of the research.

The scheme that enforces end-to-end QoS should satisfy following requirements:

- It should enforce QoS in an "open system", that is in an environment where other users and application are using the resources
- It should be robust in the presence of transient violations of QoS on the level of application components.
- It should target applications having a complex structure
- It should assume that it is not possible to have direct control over resources either from application or from other resource domain. Application may request certain service parameters but it is up to the resource manager of a particular domain to decide whether parameters could be satisfied and how exactly it could be done.

- It should minimize the amount of state exchange information between application and resource domains.

We propose using Time Petri net model because:

- The area of Petri nets has well-developed analysis techniques. If application execution is presented as a Petri net process, it can be analyzed using these techniques.
- The state information contained in a Time Petri net can serve as a source of application state information, which, in turn, could be used for adaptation in case of transient QoS violations

In addition, the Time Petri net model is well suited for modelling of distributed applications because of the following advantages:

- Similar to the commonly used Directed Acyclic Graph model, it can describe rather complex application structure
- In addition, it contains state information, so that timing characteristics between particular states of application can be analyzed
- Unlike DAG, it models communication and computational tasks in a uniform way

In addition to proposing a scheme for the QoS guarantees in loosely coupled environment, the significance of this research may include a new way of use of Time Petri nets in distributed systems. In particular, use of Time Petri net as a source of approximated application state information could be useful in building distributed systems in which communication between different components is limited.

5.2 Computation Model and Assumptions

To describe the behavior of a cyclic computation, we need a model that enables the cyclic representation of a computation which can be easily analyzed. We find that the Time Petri Net model [Wan98] is suitable for this purpose. This model, in turn, can be obtained from the commonly used Directed Acyclic Graph (DAG) model. In this section, we state the assumptions underlying the DAG model of a computation, introduce the terms and concepts used in the Time Petri Net model and describe how a Petri net model of a computation can be obtained.

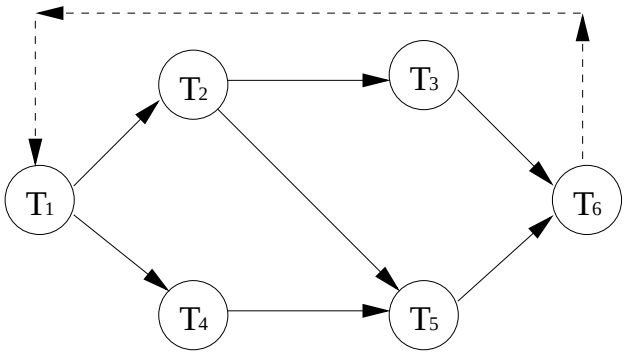

Figure 5.1: An example of the DAG model of a computation. The dashed line shows that a task T_6 from one cycle is a parent of the task T_1 from the next cycle

5.2.1 DAG model

A computation is represented by a graph ***G(V,E)*** where ***V*** is a set of n vertices and ***E*** is a set of directed edges as shown in Figure 5.1. Each vertex V_i represents an indivisible task that should be executed on a single processor. Incoming edges represent inputs to a task; outgoing edges represent outputs. A task may execute only after all inputs are available. After a task is executed, it produces outputs. A vertex with no incoming edges is called the entry task and a vertex with no outgoing edges is called the exit task. If a task V_i has an edge connecting it to a task V_j, we call task V_i the parent of the task V_j. V_j, therefore, is a child of V_i. Both vertices and edges have weights assigned to them which represent the costs of computation or communication between the tasks.

In the case of a cyclic computation, its DAG model contains several instances of the same computation tasks corresponding to the number of cycles the computation is run. For our purposes, we restrict the computation DAG to just one cycle. In addition, we maintain information about parent-child relationships between tasks from different cycles.

By adopting this computation model, we implicitly assume that a computation does not contain conditional branches at the level of tasks. This is because, in this model, *all* the incoming inputs of the task should be available and *all* the outgoing outputs are produced. There may still be conditional branches within tasks, and these are represented by the fact that the task execution time may vary within some bounds.

Another assumption about a computation is that it does not have control over its own execution. The code for all tasks are placed on the respective

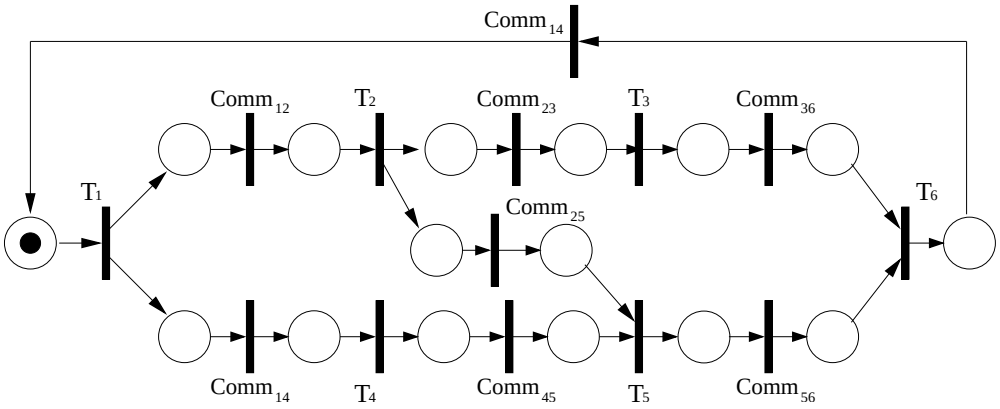

Figure 5.2: An example of a Petri net model of computation obtained from the DAG in Figure 5.1. The dot in the leftmost place is a token. This token enables the task T_1, thus making T_1 the starting task of a cycle

resources and then some set of initial tasks is started. Each task, after its completion, initializes its child tasks. The decision on when a task is run and when child tasks are initialized is made by local schedulers and not by a computation-level scheduler. In this case, the timing characteristics of a computation depend on how tasks are scheduled on resources and the current load of the servers.

5.2.2 Petri Net model

A place-transition Petri net is a tuple ($\mathbf{P}$, $\mathbf{T}$, $\mathbf{I}$, $\mathbf{O}$, M_0) and can be represented as a directed graph with two types of nodes. $\mathbf{P}$ is the set of nodes called *places* which are represented by circles in a diagram. $\mathbf{T}$ is the set of nodes called *transitions* which are represented by shaded or black rectangles in a diagram. $\mathbf{I}$ is a set of directed edges from places to transitions and $\mathbf{O}$ is a set of directed edges from transitions to places. Marking M of a Petri net is a placement of *tokens* in the places. M_0 is the initial marking of a Petri net. If a place has an edge of type $\mathbf{I}$ to a transition T then it is called an *input place* of the transition T. If a place has an edge of type $\mathbf{O}$ from a transition T then it is called an *output place* of the transition T. A transition is called *enabled* if all of its input places contain at least one token. This transition may *fire*, consuming one token from each of its input places and putting one token into each of its output places.

Figure 5.2 shows the example of a Petri net. The dot in the leftmost place is a token. Therefore, the only transition enabled is T_1. After T_1 is fired, the

token is removed from the input place of T_1 and one token is placed into each of the two output places, thus enabling transitions $Comm_{12}$ and $Comm_{14}$.

Finally, a Petri net is called *decision-free* if every place has an outgoing arc to one and only one transition.

5.2.3 Time Petri Net

As can be seen from the definition above, a Petri net model does not have the notion of *time*. However, we can introduce time by assigning a timing function to either places or transitions, and defining a firing rule which states how and when tokens are consumed in input places of transitions and produced in output places.

A Time Petri Net [Wan98] is an example of a Petri net in which time is assigned to transitions. In the Time Petri Net model, each transition is assigned a time interval. After being enabled, a transition cannot fire until the beginning of this time interval and must fire before its end.

More formally, a Time Petri Net is a tuple ($\mathbf{P}$, $\mathbf{T}$, $\mathbf{I}$, $\mathbf{O}$, M_0, SI) made up of a place-transition net $PN = (\mathbf{P}, \mathbf{T}, \mathbf{I}, \mathbf{O}, M_0)$ and a function $SI : \mathbf{T} \rightarrow Q^* \times (Q^* \cup \infty)$, where Q^* is the set of positive rational numbers. The function SI therefore assigns a pair of positive rational numbers (x, y) to each transition, where x is finite and y may be infinite. $SI(t) = (x, y)$, $x \leq y$, is called the *static interval* of transition t. In Time Petri Nets, the firing rule is given by the following:

- Transition t, once enabled, cannot fire until the time x since it is enabled and must fire not later than time y after it is enabled.
- Tokens are consumed and produced at the moment when the transition fires.

5.2.4 Construction of a Petri net from a DAG

A Petri net model is constructed from a DAG model of a computation in the following way:

1. A DAG of a single computation cycle is taken.
2. Edges going from the nodes of a cycle to the nodes of the next cycle are replaced with edges going from the same nodes to corresponding nodes of the same cycle.

3. Every vertex is replaced with a set of one transition and several places corresponding to the number of edges of a vertex. Places that correspond to edges directed towards a vertex are assigned an edge to a transition. Places that correspond to edges directed from a vertex are assigned an edge from a transition.

4. Every edge of the DAG is replaced with a transition[1].

5. Tokens are placed in the input places of the transitions corresponding to the entry task(s) of the first computation cycle.

Note that, by construction, this Petri net is decision free because any place has only one edge going to a transition.

In the Time Petri net model of a computation, each transition has the meaning of a task execution on one processor or server (can be either CPU or network link). The static interval of the transition has the meaning of the minimum and maximum time necessary to execute the task on this processor. The places have the meaning of conditions that must be satisfied before the task may be executed. The presence of a token in a place means that a corresponding condition is satisfied. The task may be executed only when all necessary conditions are satisfied, and this state is represented in the Petri net model as an enabled transition.

The Time Petri Net model is well-suited for representing a parallel computation because it captures the possibility of parallel execution and, given a function for the static interval, permits an analysis of the timing behavior of computations. Unlike the DAG model which usually models computational tasks as graph vertices and communications as graph edges, the PN model models all tasks uniformly as transitions. Because of this, the DAG model is better suited for task mapping because only computational tasks could be mapped onto resources independently, while the mapping of a communication task depends on the mapping of the corresponding computational tasks. However, the PN-based model is better suited for timing analysis of a computation execution because all the timing characteristics are bound to only one object type, i.e. transitions. Besides, a PN model contains state information and therefore permits analysis of timing between different computation states.

[1]This enables the Petri net model to take into account the communication process between tasks.

5.3 Timing Guarantees from Petri Net Model

In this section, we explain how the Petri net model can be used to provide timing guarantees. Here we use an EDF-based system as an example. However, this scheme is valid not only for EDF, but for any scheduler where actual task completion time could be significantly smaller than its deadline. In fact, every resource may use its own scheduler. The role of the proposed scheme is to provide a resource admission control with the information about the minimum task period and to ensure that the actual period is never lower than this minimum period value.

5.3.1 EDF admission control

In the case of EDF with preemption, the sufficient condition [LL73] for a set of periodic tasks with periods of at least P_i, execution times E_i and relative deadlines D_i to be *schedulable* on a single processor is that the following inequality is satisfied:

$$\sum_i \frac{E_i}{\min(D_i, P_i)} \leq 1 \tag{5.1}$$

The ratio $\frac{E_i}{\min(D_i,P_i)}$ is called the *task density* for task i. In order to guarantee the deadline satisfaction for a set of tasks, we need to make a decision on whether a particular computation should be accepted. To perform this admission control[2], we need to

- sum the densities of computation tasks assigned to the same resource
- for each resource, check whether the sum of densities of already admitted tasks and tasks from a newly-arriving computation satisfies expression (5.1) above.

For task i, the values of E_i and D_i are known in advance, but the task period P_i, which is equal to the cycle time of the computation this task belongs to, is not known in advance because it is affected by the way the computation is run. However, we can make some assumptions about the task period. Note that a computation may consist of many tasks, and according to the adopted computation model described above, all the tasks from a computation cycle should be completed during a given cycle. Therefore, the computation cycle deadline is at least equal to the longest task deadline.

If we assume that every task in a computation cycle is run only once during the cycle (if this not so, we can simply consider every instance as a

[2]This EDF admission control test serves to prevent the resource or server from being overloaded, which would lead to deadline violations for some of the admitted tasks.

separate task), then the task period is equal to a computation cycle time P and is the same for all the tasks, that is $P_i = P$ for all i's. This period can be either greater or less than a task deadline depending on the computation structure and the difference between the maximum and minimum response times of computation tasks on the assigned resources. For complex computations consisting of many tasks with a low degree of parallelism, the cycle deadline and the actual computation cycle time is likely to be significantly larger than the deadline of any task.

Consider the case where a resource is given only values of execution time E_i and deadline D_i and does not have any information at all about the computation and is, therefore, unable to determine the task period. In this case, a resource can assume that the computation is complex enough such that the task period is always longer than its deadline and use the ratio $\frac{E_i}{D_i}$ for computing task density. However, there is a risk that this assumption is wrong and that the task deadline can be violated.

On the other hand, if we have information about the structure of the computations, we can try to provide resources with the minimum computation cycle times P_{min}. This can arise in two situations: (1) the characteristics of the tasks in the computation enables the minimum of the overall computation cycle time to be determined, or (2) the computation execution behavior is modified, e.g. through shaping, such that the lower bound of the computation cycle time is fixed as P_{min}. With knowledge of P_{min}, the smallest P_i values and the worst case task densities in the EDF admission control test can be determined such that deadline satisfaction can be guaranteed for all tasks.

5.3.2 Minimum cycle Time of a Petri Net

To find the values of P_{min} , we use a property of a deterministic decision free Petri net. According to [Mur89], the minimum cycle time of a deterministic Time Petri net can be found from the following equation

$$P_{min} = \max\{\frac{T_k}{N_k} : k = 1, 2, ...q\} \tag{5.2}$$

where $T_k = \sum_{t_i \in L_k} \tau_i$ is the sum of times of transitions in circuit k, $N_k = \sum_{p_i \in L_k} M(p_i)$ is the total number of tokens in the places in circuit k, L_k is circuit k and q is the number of circuits in the Petri net.

Given a Petri net containing P places, N tokens and a set of times, the minimum cycle time can be obtained using the algorithm provided by [IP94]. The algorithm complexity is $O(NP+NE_N)$ where N is the number of tokens, P is the number of places and $E_N = O(N^2)$ is the number of edges in a graph

based on the positions of tokens in the net. This algorithm finds both the minimum cycle time and the cycle which produces the minimum cycle time. If there are several cycles giving the same minimum cycle time, then only one cycle is given depending on the algorithm implementation.

As described in section 5.2.1, the execution of an application which does not contain conditional branching on the level of tasks can be modelled as a decision free Petri net. Therefore the equation (5.2) can be used to find the minimum cycle time of an application execution. To obtain this cycle time, we need to use the values of minimum execution time of tasks t_k^{min} as T_k values in the equation.

5.3.3 Computation execution modes

Next, we need to determine the values of the minimum task execution times to be used in expression (5.2). For this purpose, we need to consider the way a computation is executed on resources and how control is transferred between tasks.

We consider two ways in which tasks can be synchronized, *greedy* and *non-greedy*. Synchronization between computation tasks is *greedy* when a child task is released as soon as all its parent tasks are complete. When a task is released, it is added to a scheduler queue for execution and is processed when its turn arrives.

Non-greedy synchronization between computation tasks assumes that a task release may be delayed from the moment when all the parent tasks have finished. This delay is defined by the specific synchronization protocol that is in use in the system. We consider the following synchronization rule: a child task is released after all parent tasks are complete, but not earlier than the maximum of $r_i + e_i$, where r_i are the release times of the parent tasks and e_i are some values associated with each parent task such that $0 \leq e_i \leq D_i$. This rule means that a signal about task completion is sent to initialize a child task only after some eligibility time e_i since the parent task was released. The behavior of the system in this case is modified as if no task can complete in time less than e_i.

If greedy synchronization is used between all of the tasks, then to find a bound on the minimum cycle time P_{min}, we take the minimum response time of the particular scheduler in use as values of τ_i in expression (5.2). In our case where all schedulers are EDF, these times are equal to the respective E_i values, the execution times of tasks on the assigned resources. However, in this case, admission control would actually admit all computations according to the peak performance of unloaded resources. This type of admission control may not be reasonable in a dynamic environment where resources are

shared among a number of users and computation tasks.

If non-greedy synchronization is used, then we need to use the maximum of execution time E_i and eligibility time e_i as the value for τ_i in expression (5.2).

5.3.4 Application cycle control using non-greedy synchronization

As explained in the previous section, non-greedy synchronization changes application behavior. In particular, it delays execution of application tasks and therefore the total cycle time of the application may increase and the maximum rate of tasks requesting a service may decrease. We can exploit this property to limit this maximum rate to a value suitable for a task scheduler. To achieve this, we need to introduce sufficient delay into application execution so that schedulability condition is always satisfied. However, the delay introduced should not lead to deadline violations. Therefore we need to introduce delays and prove that the created delay will ensure schedulability condition without deadline violation. This is similar to introducing some sort of "leaky bucket" regulator affecting the whole application.

5.3.5 Choice of eligibility times and feasible rates

Minimum application cycle time is therefore regulated by a choice of the eligibility times for tasks.

We can choose a value of eligibility time for a task k from the interval $[0,D_k]$, where D_k is the deadline of the task k. Since deadline values were chosen so that application deadline is satisfied as long as all task deadlines are satisfied, the assignment of the eligibility time within this interval will not lead to an application deadline violation.

This interval of choice of eligibility times gives us the limits on the minimum application cycle time we can guarantee. The lower bound on the application cycle time is obtained from the equation (5.2) when values of t_k^{min} are used as T_K. The upper bound on the minimum cycle time can be obtained when values of task deadlines D_k are used. For convenience, let us denote these two values of minimum cycle time as P_{lower} and P_{upper}, accordingly. These values give a range of feasible application execution rates.

Proposition 5.3.1. *We may select a set of eligibility times so that:*

- *The minimum cycle time is equal to a value P_{min} of our choice where $P_{lower} \leq P_{min} \leq P_{upper}$*

- *The application cycle deadline is not violated*

Proof. In this proof, we use an assumption (stated above) that the task deadlines are assigned in such a way that if neither task deadline is violated, the whole application cycle deadline is not violated.

Suppose we chose a value P_{min}, $P_{lower} \leq P_{min} \leq P_{upper}$

We consider the cycle in Petri Net which gives the minimum cycle time of the application when each of the tasks has the eligibility time equal to its deadline. We denote this cycle L_{upper}.

We assign zero eligibility times to all tasks except tasks on the cycle L_{upper}.

Since deadline values were used in equation (5.2), the sum of deadlines of tasks included in the L_{upper} is $\sum_{t_i \in L_{upper}} D_i = P_{upper} N_{upper}$, where N_{upper} is a number of tokens in the cycle L_{upper}.

For the tasks from the cycle L_{upper} we assign eligibility times so that their sum $\sum_{t_i \in L_{upper}} e_i = P_{min} N_{upper}$. It can be any assignment, as long as $e_i \leq D_i$, and since $P_{min} \leq P_{upper}$, we can always find such assignment.

Since actual minimum cycle time P_{actual} is defined by values $\max(t_k^{min}, e_k)$ and for the cycle L_{upper} we have that $\sum_{t_i \in L_{upper}} e_i / N_{upper} = P_{min} \leq$

$\leq \sum_{t_i \in L_{upper}} \max(t_i^{min}, e_i)/N_{upper} \leq max(T_i/N_i) = P_{actual}$, we can guarantee that the actual minimum cycle time P_{actual} is equal to or longer than the P_{min}. □

This choice of eligibility times may not give the exact value of P_{min}, but it has an advantage that it is independent of values t_k^{min} and therefore independent of the task-to-resource assignment.

The whole scheme works in the following way. The admission control input is an application with task mapping provided by some outside algorithm. The assumption that the mapping is given is reasonable because this problem itself is NP-hard and may require a complex solution [TGEO06]. When an application task is being admitted for execution on a particular resource, the value of the minimum cycle time of the task has to be submitted to the resource in order to obtain a guaranteed level of service. The minimum cycle time of an application is the minimum period time of a task from the application. We can choose a minimum application cycle time from the interval $[P_{lower}, P_{upper}]$, give it to a resource for admission control, choose a set of eligibility times for application tasks and therefore enforce that task period will never be less than the specified value. Since we assume that a resource provides a guaranteed service for applications satisfying admission control parameters and the parameters are enforced by the regulation scheme, task deadlines are guaranteed.

5.3.6 Comparison with other regulators

The proposed scheme, although described above in relation with EDF scheduler, in fact is a *regulator* which can be coupled with many existing single server scheduling algorithms. However, when implemented with other algorithms, the rate of delay may need to be adjusted to reflect admission criteria of a particular scheduling algorithm.

Since this is a regulator, in discussion of benefits of using this technique, the benefits and drawbacks of general regulators are valid for this technique as well. For example, the average delay may increase, and overall throughput of the system may decrease. However, the guarantee can be given that certain characteristics, such as end-to-end transmission delay, wound not exceed specified level [ZF94].

The advantage of the proposed technique, however, is that it allows a distributed regulator. In this case, first, only subset of servers need to have a regulator enforced before the scheduler. Second, there is a way to estimate the reasonable level of delay brought in by the regulator, by estimating the actual minimum cycle time and comparing it with the required. Third, the model makes sure that even for complex application structure the delay introduced would not lead to the deadline being exceeded. In comparison, the leaky bucket regulator in end-to-end transmission in the network has to be used before every server, and it is not clear how what would be the effect of using it for complex application structures, in comparison with linear task graph of network transmission.

5.4 Simulation study

The scheme was verified using a simulation. We simulated an environment based on the *earliest deadline first* (EDF) scheduler. We chose EDF because it has a very simple admission control, it can be easily implemented and it is commonly used in distributed real-time systems. The simulation was done using Simgrid [Sim] toolkit.

5.4.1 Simulation setup

There are six computational servers. Servers are fully connected, i.e. there is a link from each server to every other server. This type of network connectivity corresponds to the case where every server has certain amount of reserved bandwidth to every other server. We do not restrict the scenario to the homogeneous environment and generated the rates of the servers and

links to be different. The particular rates for each scenario were randomly generated and fixed for the duration of the simulation.

Every server uses the EDF algorithm to schedule incoming tasks. Link bandwidth is divided between the different computations such that tasks of the same computation are scheduled according to EDF with processing rate corresponding to the bandwidth allocated for this computation. This link model corresponds to a situation where an amount of network bandwidth is allocated to a computation, and communication tasks are scheduled by the computation according to EDF.

A continuous stream of computations was generated. The tasks of a computation that are executed on computational servers are partitioned in advance to run on two or three different servers. For example, tasks T_1, T_2, T_3 and T_6 from the DAG shown in Figure 5.1 were assigned to one server while tasks T_4 and T_5 were assigned to another. Each computation required 50 cycles to run. When a computation arrived, tasks were assigned to the specified resources and the EDF admission control decision was made. This admission control was done for both computational and network resources. After that, the admitted tasks were added to the respective servers for scheduling and execution.

We carried out simulations to: (1) explore how deadlines are satisfied and how application or computation cycle times are affected by different load conditions, and (2) evaluate how system-wide performance is affected under different load conditions and different task synchronization methods.

The load was regulated using different computation arrival rates. Computation arrival times were exponentially distributed. For every arrival rate, we generated 10 random scenarios of server and link speeds. The server and link speeds were generated independently from a uniform distribution in some pre-determined range. The arrival rates were chosen such that for a particular set of server and link speeds, the load on resources from arriving computations varied from light to heavy. The acceptance rate varied from approximately 50% to almost 100%.

We considered the cases of greedy and non-greedy synchronization between tasks. As described earlier, in the first case, a task is released as long as all of its parent tasks are complete. In the second case, a task is released only after all of its parent tasks are complete, but not earlier than $\max(r_j + e_j)$, where r_j is the release time and e_j is the eligibility time of j^{th} parent. Admission control for greedy synchronization assumed that the computation period is longer than any task deadline and used only the D_i and E_i values. In the case of non-greedy synchronization, the circuit L_D defining the upper bound on minimum cycle time was identified in each case, and the eligibility times were distributed across several tasks in this circuit.

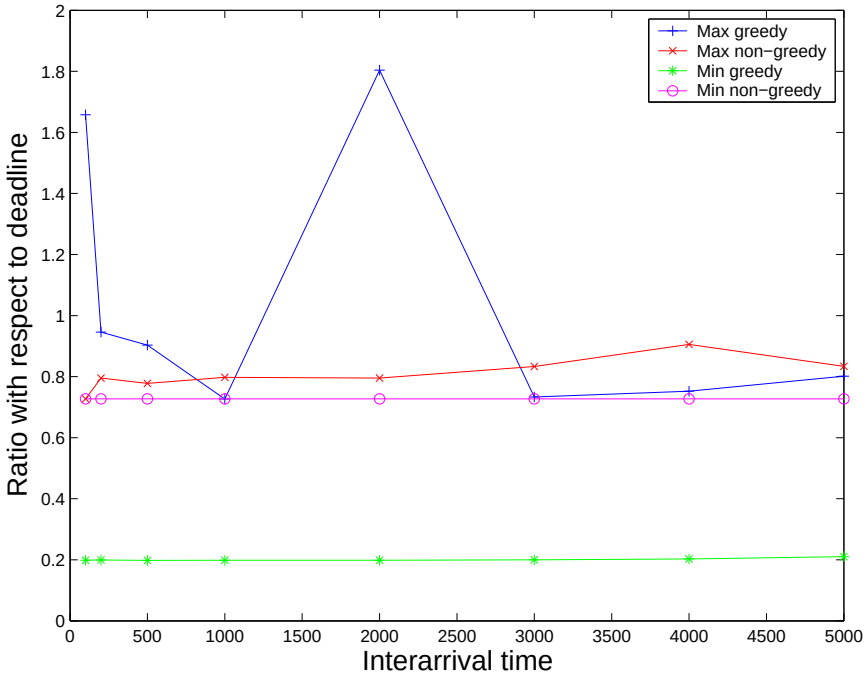

Figure 5.3: Simulation results: The ratio of minimum and maximum cycle time to an application deadline

We set the minimum cycle time $P_{min} = \max(D_i)$. Hence, the sum of ratios $\frac{E_i}{D_i}$ of tasks was used in the admission control test in both cases.

5.4.2 Simulation results

Figure 5.3 shows the deviation in application cycle time for the cases when greedy and non-greedy synchronization was used. It can be seen that when the demand control scheme was used, application deadline was always satisfied. In addition, the deviation in application cycle time in the case of non-greedy synchronization was smaller than that in the case of greedy synchronization. The minimum cycle time, in contrast, was much smaller in the case of greedy synchronization. This is because non-greedy synchronization slows down those applications which use more resources than their allocated share and therefore allows more even distribution of resources among applications.

However, such more even resource distribution is achieved at the expense of slowing down not only individual applications, but also of the whole system. Figure 5.4 shows the average resource utilization in the simulated system. The utilization in the case of non-greedy synchronization was con-

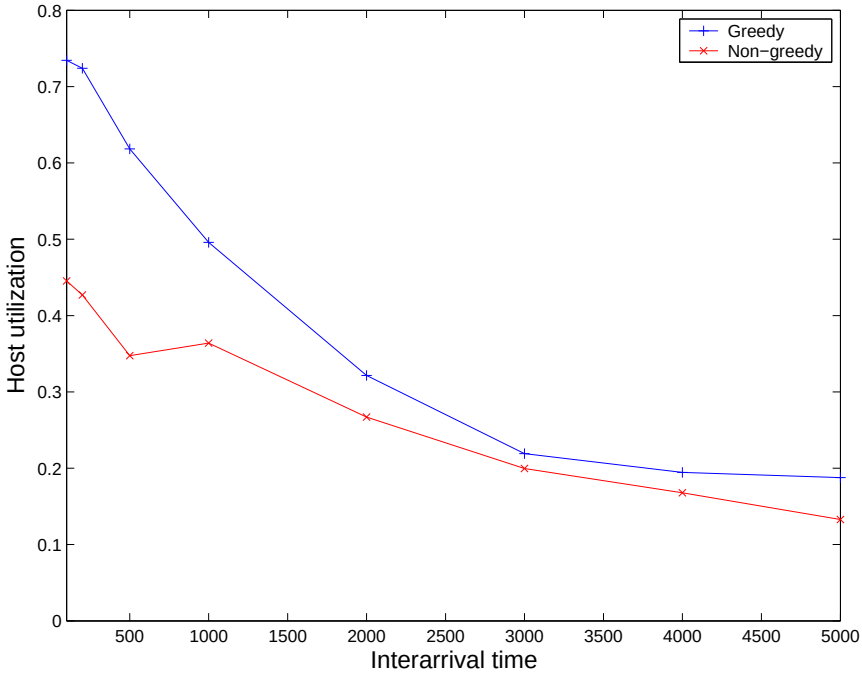

Figure 5.4: Simulation results: Average host utilization

sistently lower that that in the case of greedy synchronization. A possible explanation is that in the case of greedy synchronization when some applications do not fully utilize their share of resources, other applications may reclaim unused resources by achieving a shorter average cycle time. However, in the case of non-greedy synchronization this application speed-up is limited by the assigned minimum cycle time.

5.5 Applicability and limitations

The main advantage of the resource control scheme described in this chapter lies in the fact that all decisions on scheduling and delaying are made on the basis of information available to a scheduler and therefore there is no need for a scheduler to have communication with other resources. Communication with resources happens only during application deployment and resource reservation phase. This property makes the scheme suitable for use in loosely coupled environment.

Another advantage of the scheme is that it is able to give a tight estimation on the minimum cycle time. This is achieved by using Time Petri net model. An alternative approach is to use directed acyclic graph (DAG)

model, as suggested by Bettati [BL92]. However, DAG model can only give an upper bound on the minimum cycle time based on the critical path in the DAG. This upper bound may be loose for an application that has several input and several output nodes in its DAG model, because in this case the minimum cycle time value is a function of critical paths between different pairs of input and output nodes.

In our scheme we used a synchronization protocol similar to the *modified phase-modification protocol*. The use of this specific protocol allows us to control explicitly the time that a task spends on a resource and ensures that a task is never delayed beyond its deadline. Actually, some other synchronization protocol, for example *Release Guard* [Sun97] protocol, can be used. However, in this case we need to prove that either individual task deadlines or application end-to-end deadline is indeed satisfied. Whichever protocol is used, we still have to compute the possible range of values of guaranteed minimum cycle time. And we can obtain this range using Petri net model.

However, this scheme also has two limitations.

First, it is applicable only to applications that do not have conditional branching on the level of tasks. Although it is can be extended to include conditional branches, the complexity of the algorithm that finds the minimum cycle time in this case is exponential in the number of conditional branches. This limitation means that tasks represent high-level units of work and all units of work have to be done, which in many cases is a reasonable assumption. Tasks still may have branching inside and this branching could be represented by a variable time to execute a particular task. In fact, our Petri net application model is derived from a Directed Acyclic Graph application model, which is commonly used and, similarly to our Petri net model, does not contain branching on the level of tasks.

Another limitation of the scheme is that the resource demand is limited by a simple function similar to the token bucket regulator [Cru91]. In this case the depth of the bucket is the "weight" of all application tasks assigned to a resource and bucket rate is this weight per minimum cycle time. However, in many QoS architectures the token bucket regulator is considered sufficient for providing required functionality.

The disadvantage of the scheme is that it lowers the overall system efficiency by lowering resource utilization. This effect is commonly observed in quality of service architectures [GGPR96]. In general, since the effect is caused by applications not fully utilizing the reserved resources, this effect can be minimized in two ways. First, we may try to propose tighter admission control, which is able to consider the fact that not all application tasks can run simultaneously. Second, we may use "not claimed" resources to execute other applications. These applications generally include low priority or

best effort applications. However, if there is an economical reason to provide high priority applications with not only performance guarantees, but also with overall better service, resource providers would prefer to give these "not claimed" resources to other guaranteed application.

Chapter 6

Conclusion and future work

This thesis presents a research effort in the area of the application-level quality of service in the distributed loosely coupled environment. Two distributed computing environment were considered - sensor networks and more general loosely coupled distributed systems. These two environments have one important feature in common: communication and coordination between application entities deployed on separate resources is very limited.

Two techniques for the application-level QoS were proposed which address this lack of coordination. The third technique addresses the specific property of the sensor network application, namely, the need to take account the observed phenomena state in deciding which resources to use.

The first is a method to compute approximated delay and loss distribution for aggregated data of a sensor network query. By using this distribution, data-level Information Quality metrics can be enforced for sensor query. The significant difference of this method is in two facts. First, the structure of a sensor network query is taken into account and the probabilistic performance of a whole query is used as an admission control parameter. Second, the probability distribution for a query performance is obtained using statistical parameters measured locally on sensor network nodes thus eliminating the need for complex sensor network control.

The second application-level QoS technique is conceptually a form of a *leaky bucket* regulator, but implemented in the distributed fashion for a complex cyclic computation in loosely coupled distributed system environment, so that no additional communication is required for coordination of execution in different administrative domains and yet the regulation is achieved without unnecessary slowing down of application.

The last technique is the algorithm for a resource optimization problem formulated to guarantee the Information Quality obtained by a sensor network data-fusion application. It has a combination of properties not found

in other systems. First, it considers the general notion of phenomena tracked and takes into account the dynamic state of the phenomena to provide IQ guarantees. Second, it provides the way of taking into account the information loss and resource constraints existing in sensor network. The Dynamic Bayesian Network model is used to derive the dependency between the resources used and information quality obtained.

The general approach used in this work is based on application modelling and consists of three stages. In the first stage we analyze an application. In the second we identify the specifics of the environment which may prevent a application from obtaining the required level of service. In the third we choose a model of application and a method of using this model which can overcome the environment specifics.

In addition, we proposed a comprehensive list of the Information Quality metrics relevant to the sensor network environment. Some of the IQ metrics discussed were addressed in the sensor network query admission control and phenomena-aware resource management described here.

The further development of the work described here will mostly be in the area of phenomena-aware resource management, with more emphasis on the three aspects:

1. Other models of the information fusion have to be considered, other than Bayesian estimation, for example, grammar-based techniques.

2. Efficient computational techniques for the Dynamic Bayesian Network based resource management have to be developed.

3. Comprehensive optimization decomposition framework which addresses the resource constraints has to be developed.

Appendix A

List of publications arising from the thesis

1. Andrei Tolstikov, Jit Biswas, Wendong Xiao, Chen Khong Tham, Information Quality driven Resource Management for Human Activity Tracking, to be submitted to IEEE Transactions on Automation Science and Engineering

2. Tolstikov A., Biswas J., Tham C. K., Yap P., Eating Activity Primitives Detection - a Step Towards ADL Recognition, In proceedings of the 10th International Conference on e-Health Networking, Applications and Services (Healthcom 2008), 7-9 July 2008

3. Tolstikov, A.; Wendong Xiao; Biswas, J.; Sen Zhang; Chen-Khong Tham Information Quality Management in Sensor Networks based on the Dynamic Bayesian Network model In proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2007), 3-6 Dec. 2007, Page(s): 751-756

4. Tolstikov, A.; Chen-Khong Tham; Wendong Xiao; Biswas, J. Information quality mapping in resource-constrained multi-modal data fusion system over wireless sensor network with losses, In proceedings of the 6th International conference on Information, Communication and Signal Processing (ICICS 2007), 10-13 Dec. 2007

5. Tolstikov A.; Tham C. K., Biswas J., Quality of Information assurance using phenomena-aware resource management in sensor networks, In proceedings of the 14th IEEE International Conference on Networks (ICON 2006), Sept. 2006

6. Tolstikov, A.; Biswas, J.; Chen-Khong Tham "Data loss regulation to ensure information quality in sensor networks". In proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 5-8 Dec. 2005, Page(s): 133- 138

7. Tolstikov, A.; Biswas, J.; Tham, C.-K. "Providing end-to-end QoS in distributed computation using nongreedy task synchronization". In Proceedings. 12th IEEE International Conference on Networks, 2004. (ICON 2004). Volume 1, 16-19 Nov. 2004 Page(s):397 - 402 vol.1

8. Tolstikov A., Tham C. K., Biswas J., "Resource Load Control for Timing Guarantees in Cyclic Grid Ccomputations", In Proceedings of International Conference on Scientific & Engineering computation (IC-SEC 2004), Singapore, June 30 - July 02 2004.

Bibliography

[AKS04] Hitha Alex, Mohan Kumar, and Behrooz Shirazi. Midfusion: middleware for information fusion in sensor network applications. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 617–622, 2004.

[ANS99] ANSI/IEEE. *IEEE Std. 802.11, Wireless LAN Medium Access Control (MAC) and Physical (PHY) specifications*, 1999.

[ANS03] ANSI/IEEE. *IEEE Std. 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2003.

[ASB02] Tarek F. Abdelzaher, Kang G. Shin, and Nina Bhatti. Performance guarantees for Web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):80–96, 2002.

[Bay] Bayes net toolbox for matlab. http://bnt.sourceforge.net/.

[BDQ+05] J. Biswas, S.K. Das, Q. Qiu, V. S. Chava, and P.V. Thang. Quality aware elderly people monitoring using ultrasonic sensors. In *to appear in the Proceedings of the 3rd International Conference On Smart homes and health Telematics, ICOST2005*, 2005.

[BHA+01] Isabelle Bloch, Anthony Hunter, Alain Appriou, André Ayoun, Salem Benferhat, Philippe Besnard, Laurence Cholvy, Roger Cooke, Frédéric Cuppens, Didier Dubois, Hélène Fargier, Michel Grabisch, Rudolf Kruse, Jérôme Lang, Serafín Moral, Henri Prade, Alessandro Saffiotti, Philippe Smets, and Claudio Sossai. Fusion: General concepts and characteristics. *Int. J. of Intelligent Systems*, 16(10):1107–1134, 2001.

[BL92] Riccardo Bettati and Jane W.-S. Liu. End-to-end scheduling to meet deadlines in distributed systems. In *International Conference on Distributed Computing Systems*, pages 452–459, 1992.

[BNQP05] Jit Biswas, Felix Naumann, Qiang Qiu, and Hwee Hwa Pang. Assessing the completeness of sensor data. Manuscript submitted for publication, June 2005.

[BT01] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the internet.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[BW01] Shivnath Babu and Jennifer Widom. Continuous queries over data streams. In *SIGMOD Record*, volume 30, 2001.

[CHZ02] Maurice Chu, Horst Haussecker, and Feng Zhao. Scalable Information-Driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks. *International Journal of High Performance Computing Applications*, 16(3):293–313, 2002.

[CLCD07] Mung Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, Jan. 2007.

[Cro] Crossbow technology. http://www.xbow.com/.

[Cru91] Rene L. Cruz. A calculus for network delay, part i: Network elements in isolation. In *IEEE Transactions on Information Theory*, pages 114–131, 1991.

[CSS97] S. Chatterjee, J. Sydir, and B. Sabata. Modeling application for adaptive qos-based resource management. In *Procceding of the 2nd IEEE High Assurance Systems Engineering Workshop*, August 1997.

[DGM+04] Amol Deshpande, Carlos Guestrin, Samuel Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[FJ93] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[FK99] Ian Foster and Carl Kesselman. Computational grids. In *Grid: Blueprint for a New Computing Infrastructure*, chapter 2. Morgan-Kaufman, 1999.

[FKL+99] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proceedings of the International Workshop on Quality of Service*, 1999.

[FRS00] Ian Foster, Alain J. Roy, and Volker Sander. A quality of service architecture that combines resource reservation and application adaptation, 2000.

[GGPR96] L. Georgiadis, R. Guérin, V. Peris, and R. Rajan. Efficient support of delay and rate guarantees in an internet. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 106–116, New York, NY, USA, 1996. ACM.

[Gha97] Zoubin Ghahramani. Learning dynamic Bayesian networks. *Lecture Notes in Computer Science*, 1387:168–197, 1997.

[GN02] Xiaohui Gu and Klara Nahrstedt. A scalable qos-aware service aggregation model for peer-to-peer computing grids. In *HPDC*, pages 73–82, 2002.

[GT98] G. Ghinea and J. P. Thomas. Qos impact on user perception and understanding of multimedia video clips. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 49–54, New York, NY, USA, 1998. ACM.

[HdV05] Namwon Hyung and Casper G. de Vries. Portfolio selection with heavy tails. Technical Report 05-009/2, Tinbergen Institute, January 2005. available at http://ideas.repec.org/p/dgr/uvatin/20050009.html.

[HMCP04] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *IEEE Network Mag.*, 18(1):6–14, 2004.

[IP94] K. Ito and K.K. Parhi. Determining the iteration bounds of single-rate and multi-rate data-flow graphs. *Circuits and Systems, 1994. APCCAS '94., 1994 IEEE Asia-Pacific Conference on*, pages 163–168, 5-8 Dec 1994.

[Jen01] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[JNR05] Philippe Jacquet, Amina Meraihi Naimi, and Georgios Rodolakis. Performance of binary exponential backoff csma in wifi and optimal routing in mobile ad hoc networks. In Conrado Martínez, editor, *2005 International Conference on Analysis of Algorithms*, volume AD of *DMTCS Proceedings*, pages 365–370. Discrete Mathematics and Theoretical Computer Science, 2005.

[Lee95] Kam Lee. Performance bounds in communication networks with variable-rate links. In *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 126–136, New York, NY, USA, 1995. ACM Press.

[LHY+04] I. Lazaridis, Q. Han, X. Yu, S. Mehrotra, Nalini Venkatasubramanian, Dmitri V. Kalashnikov, and W. Yang. QUASAR: Quality aware sensing architecture. *ACM SIGMOD Record*, 33(1):26–31, March 2004.

[LL73] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, 1973.

[LN00] Ulf Leser and Felix Naumann. Query planning with information quality bounds. In *In Proc. of the Int. Conf. on Flexible Query Answering Systems (FQAS)*, 2000.

[LWF96] Jörg Liebeherr, Dallas E. Wrege, and Domenico Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking*, 4(6):885–901, 1996.

[MFHH05] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[MPI] Message passing interface forum. http://www.mpi-forum.org/.

[MPS+02] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor*

Networks and Applications (WSNA'02), Atlanta, GA, September 2002.

[MSB] Microsoft bayesian network editor and tool kit. http://research.microsoft.com/adapt/MSBNx/.

[MSFC02] Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 49, Washington, DC, USA, 2002. IEEE Computer Society.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, pages 541–580, April 1989.

[Mur02] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[NJZ99] K. Nichols, V. Jacobson, and L. Zhang. Rfc2638: A two-bit differentiated services architecture for the internet, 1999.

[NR00] Felix Naumann and Claudia Rolker. Assessment methods for information quality criteria. In *Fifth Conference on Information Quality (IQ 2000)*, pages 148–162. MIT, 2000.

[PHC04] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM.

[PSB04] Walter Perry, David Signori, and John Boon. *Exploring Information Superiority: A Methodology for Measuring the Quality of Information and Its Impact on Shared Awareness*. RAND, 2004.

[RFG+00] Alain J. Roy, Ian Foster, William Gropp, Brian Toonen, Nicholas Karonis, and Volker Sander. Mpich-gq: quality-of-service for message passing programs, 2000.

[Sim] Simgrid. http://simgrid.gforge.inria.fr/.

[SR05] Jens B. Schmitt and Utz Roedig. Sensor network calculus - a framework for worst case analysis. In *Lecture Notes in Computer Science*, volume 3560, pages 141 – 154, 7 2005.

[SS99] David Starobinski and Moshe Sidi. Stochastically bounded burstiness for communication networks. In *INFOCOM (1)*, pages 36–42, 1999.

[Sun97] Jun Sun. Fixed-priority end-to-end scheduling in distributed real-time systems. Technical report, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1997.

[TBT05] Andrei Tolstikov, Jit Biswas, and Chen-Khong Tham. Data loss regulation to ensure information quality in sensor networks. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing ISSNIP*, pages 133–138, 2005.

[TGEO06] Yuan Tian, Yaoyao Gu, Eylem Ekici, and Fusun Ozguner. Dynamic critical-path task mapping and scheduling for collaborative in-network processing in multi-hop wireless sensor networks. In *ICPPW '06: Proceedings of the 2006 International Conference Workshops on Parallel Processing*, pages 215–222, Washington, DC, USA, 2006. IEEE Computer Society.

[Wan98] Jiacun Wang. *Timed Petri Nets: Theory and Application*. Kluwer Academic Publishers, Boston, 1998.

[WC01] Alec Woo and David E. Culler. A transmission control scheme for media access in sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, New York, NY, USA, 2001. ACM Press.

[WS96] Richard Y. Wang and Diane M. Strong. Beyond accuracy: what data quality means to data consumers. *J. Manage. Inf. Syst.*, 12(4):5–33, 1996.

[XNVW00] Dongyan Xu, Klara Nahrstedt, Arun Viswanathan, and Duangdao Wichadakul. Qos and contention-aware multi-resource reservation. In *HPDC '00: Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing*, page 3, Washington, DC, USA, 2000. IEEE Computer Society.

[XRC+04] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM.

[YHE02a] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. 2002.

[YHE02b] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2002*, New York, USA, 2002.

[ZF94] Hui Zhang and Domenico Ferrari. Rate-controlled service disciplines. 1994.

[Zha95] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks, October 1995.

[ZJ06] Yongmian Zhang and Qiang Ji. Active and dynamic information fusion for multisensor systems with dynamic bayesian networks. *IEEE transactions on Systems, Man and Cybernetics, Part B*, 36(2):467–472, April 2006.

[ZLL+03] Feng Zhao, Jie Liu, Juan Liu, Leonidas Guibas, and James Reich. Collaborative signal and information processing: an information-directed approach. *Proceedings of the IEEE*, 91(8):1199– 1209, August 2003.

[ZPB02] J. Zhang, K. Premaratne, and Peter H. Bauer. Resource allocation and congestion control in distributed sensor networks - a network calculus approach. In *Proceedings of 15th International Symposium on Mathematical Theory of Networks and Systems*, University of Notre Dame, August 2002.