Technical report, IDE0609, October 2008

ARQ PROTOCOLS SUPPORTING QOS IN EMBEDDED SYSTEMS

Master's Thesis in computer system Engineering

Aydin B.Mofrad

School of Information Science, Computer and Electrical Engineering
Halmstad University

Preface

I sincerely thank Professor Magnus Jonsson and Kristina Kunert for giving me the opportunity to work for my Master's thesis in CC-lab under their supervision and for their guidance all through the thesis. Their valuable suggestions and ideas were a great strength for the achievement of my goals in this project. I would also take the opportunity to thank my friend Praveen K. Sigirisetty who started the Thesis with me and inspired me to keep working.

I would also like to thank my parents, and friends for their support and guidance.

Aydin B.Mofrad
Halmstad University, October 2008

List of Figures

Table of contents

Abstract

Many efforts have been carried out to provide transmission reliability in the history of communication systems. As the demand for real-time applications increased, providing a reliable communication in a timely manner for such applications is strongly desired. Considering timing constraints makes the issue of achieving reliability more difficult. This thesis concentrates on providing reliability for real-time communication in embedded networks by achieving a timing analysis and using the ARQ concept. What is carried out in this thesis is providing retransmission in a real-time manner for embedded networks according to application request. The thesis work focuses on one packet retransmission over a point to point link, but the concept is rich and can be extended to cover application request in real-time embedded networks. Two methods have been fulfilled, and a simulation has been done on the timing analysis focusing on the performance in accepting real-time traffic in the form of separate channels for each application request. The protocol combines ARQ and a scheduling algorithm as a base to support retransmission for hard real-time applications in embedded networks.

1 Introduction

The value of time in communication systems has become more and more vivid and today's applications make the real-time communication an inevitable need and Distributed Real-time Embedded (DRE) Systems become more and more common. DRE systems have a variety of applications ranging from very important military applications to some ordinary daily applications in people's life. Examples of these systems can be automation systems, car engines, robotics, aviations, control systems, telecommunication, military systems, multimedia systems, interactive games and many more.

Meanwhile the performance that can be achieved in a timely constrained communication is of great importance. Methods concerning performance in real-time communications can be divided into two major groups: guaranteed schemes and best-effort schemes. As the names imply guaranteed schemes give guarantees of performance in a particular scope and best effort schemes try to improve performance the best they can in a specific scope. Depending on the application's criticality and different types of network characteristics and traffic patterns, different methods can be used to improve performance. In some sensitive applications, guarantees are needed whereas in some others a best-effort scheme can be more suitable regarding the complexity and cost that implementing guaranteed schemes may have. Some methods concerning real-time communication take advantage of real-time channels, so that the physical communication channel is divided into several logical real-time channels and different methods are used to establish real-time channels with different specified characteristics and Quality of Service (QoS) requirements.

The outline of the thesis is as follows: after motivation and problem statement, some background is described in chapter two, namely a brief description of reliable transmission, ARQ and a short introduction of real-time systems. In chapter 3 real-time communications is taken into account and packet format and features of these systems plus traffic pattern and channel establishment process in our target networks are described. The chapter also includes scheduling analysis and some information about scheduling algorithms. In chapter 4 some

related works are mentioned. More details about assumptions and methodologies can be found in chapter 5 and a thorough description of solution proposals in chapter 6 including schedulability analyses. A simulation part which includes conclusions made on the evaluation and simulation results follows in chapter 7.

1.1 Problem Statement

As the communication path is not always reliable and there can be erroneous packets, there is a need for retransmission to provide reliable communication. Depending on how critical the application is and how often erroneous packets appear, the need for retransmission increases. Retransmission in real-time systems is limited by the deadline constraint and in order to be able to do retransmission within the deadline a scheduling analysis needs to be made to analyze the worst-case situation.

The approach is to take advantage of ARQ in real-time embedded systems and make a worst-case scheduling analysis to support the application layer with a number of desired retransmissions. In this thesis one packet retransmission is assumed, but the scheme is extendable to support more than one retransmission, actually as the application request. It is clear that as the application request for the retransmission increases, the efficiency of the system drops, because the deadlines should be met and the system should not be overloaded.

Therefore a scheme is proposed for hard real-time systems, in which transmission and retransmission of the packets are done within the deadline and the scheme tries its best to retransmit one faulty packet for each message. In other words the problem is to make a scheduling analysis framework in real-time systems in which the best possible reliable transmission can be made.

2 Background

2.1 Reliable Transmission

Reliability is desirable in almost all sorts of communications in human life. This desire applies to Communication systems as well. As the backbone of the communication networks may always change in the case of technology or topology, it is preferable to have reliability in a layer that operates on an end-to-end basis between two or more communicating hosts. In communication networks layering, the general role of a transport layer protocol is to provide this end to end inter-communication between processes of two or more different computers abstractly. A good transport layer service provides the connection between communicating hosts on a variety of different networks without worrying about different network structures.

A transport service is reliable if there is no-loss, no-duplicates, data order and data-integrity. And the service is unreliable if any of the features are missing. No-loss means that all packets

should be delivered. Providing this feature is possible by retransmissions and error correction policies, as there are disturbances in all communication paths. No-duplicates means that packets retransmitted and original packets should be recognized by the service in the receiving media and no duplication is allowed. Ordered means that: if it is needed, packets should be organized and ordered in the receiving end. For example retransmission packets always need to be put in the original packet's order. By using a suitable ARQ strategy, packets would be arranged in order.

Data integrity means that the data received by the receiver is exactly the same data that has been transmitted; it is achieved by a suitable error detection strategy. For example a checksum is usually used, which detects errors and in case there was an uncorrectable error in the received data, a retransmission is done to provide this data integrity.

2.2 ARQ (Automatic Repeat reQquest)

Error control in communication systems consist of two major parts: error detection and error correction. Error detection is the process of checking the received data in the receiver's side to see if there exists any error in the received data, but error correction is the procedure in which the receiver tries to fix the possible errors in a received chunk of data. Generally error detection algorithms also have some ability to correct some failures in the received data and also act as an error-correction strategy. Simply there should be some kind of agreement between sender and receiver that they make sure that the data received in the receiver is the one which is sent by the sender, otherwise the data should be retransmitted. Error control is normally processed in the transport layer and in some cases it is done in the link layer as well.

One of the methods, by which communication systems control transmission errors, is called Automatic Repeat reQuest or Automatic Retransmission Query (ARQ) [i], and it functions as follows:

The sender transmits the data and starts a timer, the receiver checks the data by means of error-detection algorithms and replies with a positive acknowledgment if the data was correct or for some systems with a negative acknowledgement showing that the data was erroneous and could not be fixed.

The timer is there in case the receiver's reply is lost or damaged or in the case that the original message is lost (and the receiver never gets the message to acknowledge it). When the sender receives no acknowledgement it uses the timer to keep track of the time it has waited for the acknowledgement. Communication systems make a limit on waiting for the acknowledgement and when no acknowledgement is received during this specific time period, called time out, the sender assumes that the receiver did not receive the packet and therefore retransmit it. Specifying this time out is a very delicate manner in communication systems, because long time outs waste resources and need buffers and more calculation for the packets in the buffers, and especially in real-time systems timing constraints will not allow long time outs. And short time outs have the drawback of unnecessary retransmissions, because the acknowledgement can be slightly late. Therefore calculating time out plays a very important role in communication systems and it depends on the networks characteristics such as transmission and queuing delay, and it is mostly defined as a function of round trip time.

There are different ways of doing this procedure. Types of ARQ methods include Stop and Wait ARQ, Go Back N ARQ and Selective Repeat ARQ [ii]. There are also some more complex variations of ARQ which are known as Hybrid ARQ and are mostly used to improve performance.

Stop and Wait is a simple ARQ method in which the transmitter transmits a frame and waits for acknowledgement. The transmitter is supposed to wait and does not send any further information until the acknowledgement is received. It is one of the early ARQ methods, and as faults can always happen in the communication path, either the frame or the acknowledgement can be lost or be erroneous. There are some special cases which may cause problems. Consider, for example the case that a frame reaches the destination and it is correct. The receiver sends an ACK, and if the ACK is lost the sender time outs and retransmits the data but the receiver getting the retransmitted data may assume that the received frame is the next frame and new information. The same thing happens if the sender has a short time out and receives the ACK late, in which case it assumes that the ACK is for the retransmitted data but the receiver gets one frame twice. This is where a need for sequence number comes into existence. Sequence number is a way to mark frames of information and by means of these marks sender and receiver synchronize on frames of information. In case of Stop and Wait one bit sequence number would be enough and by numbering packets 0 or 1 receiver and sender can recognize a new frame from the old one.

The major problem with the stop and wait is that no data is transmitted while the transmitter waits for the acknowledgement and resources would be wasted.

In Go-Back-N method, transmitter would not wait for the ACK and it sends a series of information to the receiver, and if it receives a NACK it goes back and retransmits all the frames starting from the erroneous frame. It is obvious that more sequence numbers are needed and this scenario is played with the help of a concept called sliding window. Go back N has a better performance as it uses the bandwidth while waiting for the acknowledgements but it should also be mentioned that when a NACK is received or a time out occurs, it re-transmits a huge pile of received data.

However in Selective repeat ARQ just the faulty frame is retransmitted. Basically the receiver accepts frames after an erroneous frame (until the end of its window size) while waiting for the retransmission of the faulty frame. This way bandwidth is utilized in a much better way and the performance improves. More information on sliding windows and details of Selective Repeat ARQ can be found in [iii].

2.3 Real-Time Systems

Real time refers to sensing and responding to external events as they occur and within a limited time. Real-time systems are defined as those systems in which “The correctness of the system depends not only on the logical result of computations, but also on the time at which the results are produced [iv]. A typical real-time system monitors and controls external processes. The system responds to changes in the external process in a timely manner, usually on the order of milliseconds. To provide predictability to the system and to handle resources to meet the timing constraints of the physical system the scheduling algorithm and the scheduler of the operating system play a vital role in maintaining the job. These systems are called real-time systems,

because they need information to be delivered with a time constraint which is called deadline, and dependencies to time constraints are different in different systems.

Missing deadlines is not desirable in real-time systems and it is essential that the timing constraints of the system are met. Keeping up with timing behavior requires that the system is predictable.

Generally real-time systems are divided into two groups

1. Hard real-time systems
2. Soft real-time systems

Hard real-time systems: Some Systems are very sensitive to these time constraints and in case the constraints are neglected or a deadline is missed, the functionality of the system is disturbed or something catastrophic happens to the system or the environment the system is working on. These systems are called hard real-time systems and examples of these systems can be found in military applications or robotic systems.

Soft real-time systems: Some other systems are not very sensitive to time constraints and if a deadline is missed, it would have a lower effect on the system functionality and performance. These systems are called soft real-time systems and examples can be found in multimedia systems such as a video conferencing.

Systems which are not sensitive to time constraints are called non real-time systems and many of today's applications like a simple computer simulation, a simple file transfer or a backup system are non real-time.

When real-time features are added to a communication system, we expect the system to function in a timely fashion, and what disturbs the ideal timings expected from a system's behavior is delay. In general communication delays can be categorized in two main categories: transmission delays which occur while transmitting a message through the communication network (Transmission Path Delay) and the time it takes inside the communication node until the message reaches the application (Delivery Delay). Both these delays can vary because of the inter-modules of the networks, such as routers or switches, or the receiving devices. In both cases variation of the traffic can also cause a variance in delays. For example as the traffic is heavy in a switch or router buffers may overflow or the message's path to the receiver may change, meanwhile queuing delay is also a varying factor of delays in receiving node. Also in the transmitting nodes, there might be delays in a corresponding to the receiving side.

To provide the required timing for real-time systems, delays must be calculated and bounded from the sending module or application to the receiver one. In some cases early arrival of messages also leads to destruction. The properties of a communication protocol include communication medium, switching technique and network topology. To make a real-time communication protocol, scheduling is also an important issue, which decides the order of message transmission. Scheduling will be discussed in Chapter 3.3.

3 Real-Time Communication

What is brought up by today's sensitive applications and the need of accuracy and time dependency of these applications is a real-time concept to systems. It means the importance of timing constraints in the applications. An important feature of real-time communication is that the communication depends on the times at which messages are successfully delivered at their destination. The delivery time is bounded by a definitive maximum delay for each message over the network; consequently deadline is associated for each message. If any message arrives after the expiration of deadline to the destination host, it reduces the end applications value and may lead to loss of message.

Three basic approaches of communication are stated in [v]: circuit switching, packet switching and hybrid switching. Among these approaches authors have discussed packet switching in real-time communication to provide performance guarantees. As the target networks in this text are real-time embedded networks, the case discussed and studied here contains packet switched real-time communication.

3.1 Real-Time Traffic

The traffic pattern is an important parameter to consider in real-time communication in an embedded system. Real-time traffic is mostly periodic, meaning that it is generated at regular intervals. For example a sensor often sends some information about the system in a periodic manner at regular time intervals. This interval is called period (*P*). Real-time traffic can also be aperiodic or non-periodic, in which case it does not have a pattern or a certain regular interval. For example in a control unit, a thermostat will simply generate a signal as the temperature exceeds a limit. This signal does not have a special interval or appearance time and is therefore called an aperiodic signal.

General features of real-time traffic can be addressed by tree factors:

Transmission Rate
Packet Size
Delay Tolerance

As discussed in [vi], Periodic Traffic Sources can generate packets in one of the tree formats as explained below (see Figure 1 also):

Constant Bit Rate (CBR): Packets of a fixed size and a Fixed inter-arrival time. CBR is mostly generated by sensors that have a fixed size data and regularly these data are generated as the sensor senses, for example a whether forecast sensor.

Variable Bit Rate (VBR): On/Off Sources: Fixed packet size and a fixed interval of time which is active or inactive for a period of time. An example would be the voice-packet pattern in an ATM system.

Periodic with Variable Packet Sizes: Different Packet Sizes generated in a periodic pattern as a vision system in a robotic application can do. Depending on the encoding of the video frames, different packets will have different sizes. In case of aperiodic traffic any of the three features can be changed. This format is usually characterized by an inter-arrival time and an average

packet size, each of which can have different distribution, such as for example uniform or delta distribution.

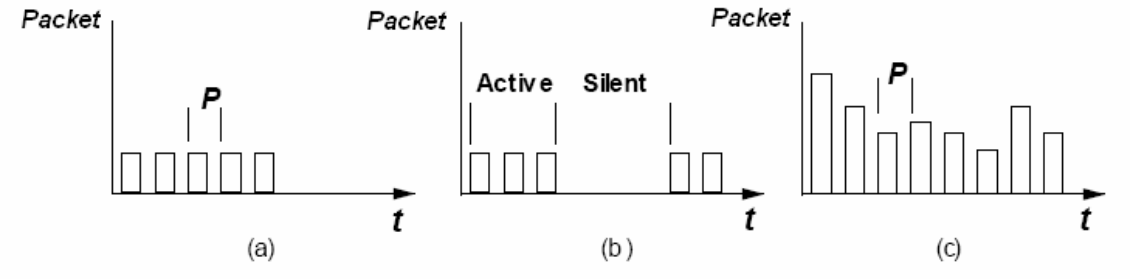

Figure 1: packet arrival forms
(a) CBR (b) VBR and (c) periodic with variable size (*packet*=packet size, *t*=time, *P*=period)[vi].

Both periodic and aperiodic packet streams can have hard or soft deadlines and would be treated differently by different control systems. Aperiodic requests which have a minimum inter arrival time between their different instances and have hard deadlines are called sporadic requests.

In real-time Embedded Networks the structure of the network is assumed to be known. This helps the calculation of transmissions and provides timing constraints more easily. It is assumed that the target networks have a packet format in which each packet has some extra fields in the header.

The data flow which is going to be transmitted from one node to the other is divided into messages. The beginning of each communication flow starts with a channel establishment process. During this process a communication path is setup from source to destination. This can be done for example by reserving some bandwidth for a special application. These data flows can be in terms of periodic or non periodic (aperiodic) messages of different or the same size. Channel establishment policies take care of the most efficient way of making these channels of periodic messages or closing or opening a new channel each time for an aperiodic flow. The actual channel-establishment process is discussed in the next sub-chapter.

Then for effective bandwidth allocation reasons and scheduling purposes every message in each channel is divided into some packets. Depending on scheduling and transmission strategies, the size of packets from messages can be different or the same for each channel. Each message is considered a task instance or a job. As mentioned before these messages can be periodic or non periodic.

3.2 Channel Establishment

Channel establishment is the logical connection between two nodes to provide real-time services. To send messages from one node to another node, a channel is established to control resources, communication path and other parameters that may be important for specific purposes of the communication system. Channel establishment process provides an agreement between communicating nodes to meet communication needs. This agreement is done according to a communication protocol. When a channel is made a logical end-to-end connection is made between the two nodes, which work under the special rules of that channel and communication

protocol. Thus, in order to synchronize communicating nodes and provide connection needs, a real-time channel is to be established to deliver real-time traffic through the network.

Channel establishment strategies are different in the case of assigning resources to nodes and policies determining how resources should be scheduled. Differences in these strategies are mostly dependant on the traffic flow and regularity of the messages. Sometimes it is better to close a connection channel which is used non-periodically and without a pattern than to waste resources.

Another important issue is that channel establishment has some overhead on the system, and to provide agreements between the sender and the receiver, the system needs to check the availability of the sender's request. This leads to some request and reply messages and some measurements are done by the system. It is important that the cost of channel establishment is low compared to the transmission flow. Consider a channel establishment process which takes some round trip times with some large messages to make agreements for a channel which is used for transmitting just a few packets; this makes an undesirable overhead on the system, which might be high if channels are set up and occupy a lot of bandwidth. The overhead can also be measured by time, considering a channel establishment process which would take some milliseconds and the related application will use the transmission channel for just a few microseconds.

3.2.1 Logical Connectivity between Two Layers

Logical connection is a logical link associated with the arrangement of a suitable type of services. It must be a peer-to-peer connection between two network nodes. Generally channel establishment is a procedure for establishing a channel in a logical link control level and an adaptation layer protocol helps to provide connection-oriented or connectionless data services for upper layer protocols.

To provide end-to-end performance in a packet-switched network, admission control and service discipline are the two levels of control. Admission control at connection establishment makes sure that the resources are available for the requested service, and service discipline at the packet level take care of QoS. The service discipline controls the order of serving packets and decides how the packets from different connections can interact with each [vii].

To provide real-time communication over packet-switched network the common approach is to establish an end-to-end bounded point-to-point connection and several works have been done in real-time communication for packet switched network,[v] ,[vi] and [vii].

3.2.2 Examples of Different Channel Establishment Methods

In a real-time MAC protocol [viii] a real-time channel is established by reserving the resources as per the designated traffic characteristics of the periodic message, if the admission of this connection does not threaten the guarantees given to the already admitted channels. While admitting a periodic message stream the protocol has to perform an associated admission test.

Ferrari and Verma [ix] describe a scheme for channel establishment with deterministic delay bounds in wide area networks. They devised a single round-trip procedure for establishing a channel. A message can go from a source node to a destination node passing through a number of intermediate nodes. At the time of channel establishment, clients declare their traffic characteristics and performance requirements to provide real-time services. The destination node is the end point along the path where decision of acceptance or rejection of channel requests is made. They perform several tests and reserve resources in each node visited by the request message for channel establishment. If the test fails at any particular node, the channel cannot be established along that path.

Han and Shin [x] presented a failure-recovery scheme for dependable real-time communication services in multi-hop networks. They exhibited a scheme to restore real-time channels, each with guaranteed deadline, from network component failures in multi hop networks. They set up back-up channels inclusion to the primary channels to guarantee successful rerouting. In the case of failure of primary channel; one of the backup channels is activated as a primary channel. Chou and Shin [xi] proposed a scheme to improve network utilization by using statistical real-time channels for the performance requirements specified in statistical terms, which is easy to implement and used for many applications.

In all the examples above, every scheme has some kind of routing investigation in the method and the schemes try to apply flow and congestion control in the network. In this thesis as a point to point link is discussed, no flow and congestion control is considered. The specialty of the case is to provide a special service.

3.3 Scheduling

3.3.1 Real-Time Scheduling

In real-time systems, real-time scheduling takes an important role in solving the problem of allocating resources to a task in a specific time interval to meet timing requirements. Resources are assigned to different tasks according to a predefined theory which is called *scheduling policy* and a set of rules specify the execution order of the tasks (e.g. which task can have resources at each time instance) and it is called a *scheduling algorithm*. A schedule is called feasible if all tasks can be completed according to a set of constraints (e.g. within their deadline). A task is guaranteed at its activation time when a feasible schedule can be found for all tasks which exist in the system or will come to the system. A system capable of guaranteeing and executing tasks with hard time constraints is called hard real-time system. In some hard real-time systems what is desirable is not only a feasible schedule but also a minimum in average response times [xiv].

In our case tasks are messages with real-time characteristics, and the execution of a task is equal to transmitting a message consisting of some packets.

Mainly Scheduling policies can be characterized as static or dynamic [xiv] but there are some common terms in real-time scheduling. These terms are defined as follows:

Static versus dynamic: Scheduling algorithms can be static or dynamic. In Static scheduling (fixed) all priorities are fixed and will never change during operation, whereas in dynamic scheduling tasks may be assigned new priorities during operation.

Online versus offline: If a scheduling decision is made whenever a new task arrives (during run-time) to the system, the algorithm is on-line and if timing analysis is made before the operation of the system, the scheduling is done off-line.

Pre-emptive versus non-pre-emptive: pre-emptive scheduling permits the suspension of a task before it completes, in order to allow another higher priority task to be executed. But in a non-pre-emptive scheduling once a task starts, it must execute until completion without any interruption.

Pre-emptive scheduling acquires more system overhead and requires more processing than the non-pre-emptive scheduling, but it is more desirable since scheduling can be done dynamically.

3.3.2 Scheduling Algorithm

In order to handle different real-time traffic transmissions a scheduling policy should be used in which a set of rules specify the transmission order of the packets, depending on their size, deadlines, delays, order, importance (as being soft or hard) etc.

There are two major optimal scheduling algorithms (see [xii] and [xiii] for examples): Rate Monotonic (RM) and Earliest Deadline First (EDF). Although RM is an optimal algorithm, it is static, and in the worst case the maximum processor utilization that can be achieved when using RM is about 69% [xiii]. Processor utilization can be increased by using dynamic scheduling algorithms, such as Earliest Deadline First (EDF) [iv]. EDF has been shown to be optimal, and can achieve full processor utilization and run with smaller overhead, if it is schedulable. The schedule is feasible if the total utilization is less than one ($U \leq 1$ is a sufficient and necessary condition for scheduling) for deadlines equal to or longer than the period [xiv].

Deadline scheduling: The EDF scheduling algorithm schedules messages in the order of their deadlines. Dynamically the tasks which have shorter deadline will execute before the others. In other words: EDF assigns highest priority to the task whose deadline is closest to the current time.

4 Related Projects

4.1 ARQ Based Retransmission Guaranteeing Hard RT Communication

Jonsson and Kunert [xv] has developed a framework in which they use ARQ combined with worst case scheduling analysis under EDF and guaranteed packet retransmission. The main idea in the protocol is that the transport layer delay bound is divided into one delay bound for ordinary transmission of the packets belonging to the message and one delay bound for the possible retransmission of one of the packets. The strength of the protocol is in the calculation of

the delay bounds to guarantee hard deadlines for both ordinary and retransmission packets. The real-time analysis maps the transport layer delay bounds to the network layer, while considering the division into ordinary packet transmissions and retransmissions.

The protocol is described and defined to support one packet retransmission for each message over a point to point link. An extension is even enhanced to advance the protocol with more than one packet retransmission support. A separate link is considered for acknowledgements so that it builds a full duplex link between the transmission parties: one for RT data transmission and the other for acknowledgements.

Another interesting point in the protocol is that messages are divided into full size packets and one last packet which is not necessarily full size. All delay bounds both in ordinary packet transmission and retransmission channels are calculated considering the actual packet size, which in case of the last packet can be less than the full size. What is noticeable is that this last packet (which can be less than full size) does not affect the retransmission channel to any great extent, since the retransmission channel is used for both full size packets and the last packet. In other words, the period of the retransmission channel is always a function of full packet size as long as at least one RT channel has a message length of at least one full size packet [xv]. Exact timing calculation helps the protocol to guarantee hard deadlines and to have a better performance.

The protocol described in paper [xv] can be considered as the most related work to this thesis work and it is quite similar to the second proposal described in the solution part. The main difference is that the real-time analysis discussed in Jonsson and Kunert's work is a guaranteed scheme whereas our scheme is a best-effort scheme. The difference becomes more clear when sending retransmissions and accepting RT channels. Because both deadlines and one packet retransmission are guaranteed, a tighter timing is needed and the scheme has to consider shorter delay bounds for ordinary packet transmission. This is because of the time needed to control reception of all ordinary packet transmissions. Retransmission starts after knowledge has been gained about the reception of all ordinary packets, but in our scheme we have more free timing and the delay bound for ordinary packet transmissions are wider. This fact would give the approach the potential to accept more RT channels and provide a best effort on retransmission of one possible erroneous packet. Furthermore, it can be used for more relaxed timing constraints in which a best effort would be enough and hard real-time guarantees would not necessarily be needed.

4.2 A Scheme for RT Channel in WAN's

Ferrari and Verma [ix] describes a method for guaranteeing delays in a packet-switching wide area network, and presents an evaluation of some of its most important characteristics. Real-time service guarantees performance by taking the form of lower bounds on the bandwidth allocated to a channel and upper bounds on the delays to be experienced by a packet on the channel. Basically, it has the feasibility of providing real-time services on a packet-switched store and forwards wide-area network with general topology.

The two types of delay bounds used are both deterministic and statistical. Delay bounds are valid only for packets that reach the destination. As it is very difficult to build real-time channels on top of a connectionless service this scheme works in conjunction with specific scheduling, flow-control policies and connection-oriented packet switching as the basis of a real-time

service. A single round-trip procedure for establishing channels has been devised. Performance can be guaranteed because of the scheduling and distributed rate control policies. Simulation experiments have failed to disprove the correctness of the scheme even in worst-case situations.

4.3 Real-Time Channel Establishment for Packet Switched Networks.

Zheng and Shin [xvi] define a real-time channel as a unidirectional connection between two nodes in a network that guarantees every packet to be delivered before a user-defined, end-to-end deadline. They presented solutions to two fundamental problems associated with the establishment of real-time channels: checking the schedulability of channels and computing the minimum delay bound over a link. They had given a mathematical basis for the problem and time schedulers in the switches make decisions which packets are transferred first over the links.

The channel-establishment procedure includes routing, performance verification and connection confirm or denial. The end-to-end packet delivery delay is the summation of delays over links and nodes along the selected route. They have not addressed the routing problem, which is the first step in establishing a real-time channel.

5 Methodologies and Assumptions

This part is going to review the main assumptions which support the real-time transmission protocols and which provide a best effort in retransmitting one packet of a message. It should be noted that the protocol is very general and can be extended to support more than one packet retransmission and provide the number of retransmissions desired by the application for each channel. Giving a best effort on reliable transmissions by retransmitting one packet, all timing constraints and deadlines of ordinary transmissions are guaranteed. EDF is assumed as the scheduling algorithm.

5.1 Terms and Definitions

This part defines important terms used in the text. These terms are used mainly in calculations, timing analysis and solution descriptions. Basically almost the same terms are used in other texts and books, but they are exclusively defined here to avoid ambiguity. In the text each message is denoted as m_i and characterized by some parameters in order to be applied in a scheduling algorithm.

The extra parameters defining a message's real-time features are as follows, and they are characterized in Figure 2 , to provide a more descriptive view.

Period P_i : is the regular interval in which a task is activated repeatedly. As discussed before this parameter is an exclusive feature of periodic tasks.

Arrival time a_i : is the time at which a task/message becomes ready for transmission; it is also referred to as request time or release time.

Length of the message L_i : is the time necessary to transmit the message through the network. In other words it can be defined as the length of the message.

Absolute Deadline d_i : is the time before which a message should be transmitted to avoid damage or performance degradation depending on the task being hard or soft. After this time it may be of less importance or completely useless to receive the message.

Relative Deadline D_i : is the difference between the absolute deadline and the arrival time:

$$D_i = d_i - a_i$$

Start time s_i : is the time at which a message transmission starts.

Finishing time f_i : is the time at which a message is sent and transmission finishes. Note that it can be later than the deadline.

Response time R_i : is the difference between the finishing time and the arrival time:

$$R_i = f_i - a_i$$

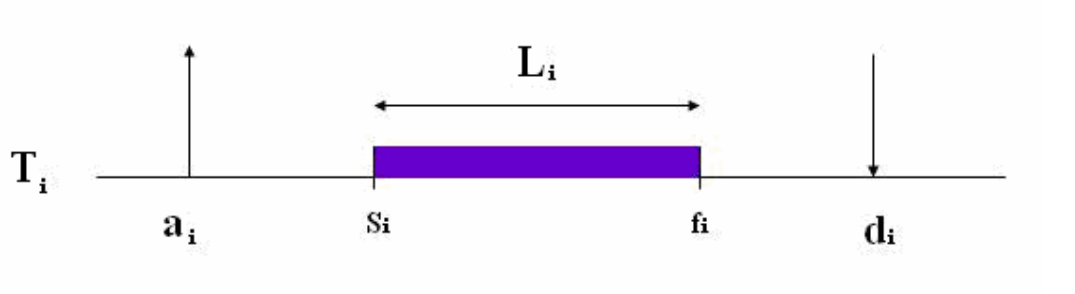

Figure 2: Typical parameters of a real-time message [xiv]

Definitions have been carried out with the help of [xiv] which is a major resource in real-time systems.

5.2 Channel Establishment

In this approach, the application layer sends requests to the transport layer. Each request for a channel consisting periodic message T_i includes the following parameters: Length of the message (L_i), period (P_i), Relative deadline (D_i), Arrival time (a_i) Maximum packet size (L_{max}), and number of retransmissions. In this thesis the number of retransmissions is assumed to be one, for simplicity in the calculations, but the scheme is extendable to meet the application request of more than one retransmission. Parameters in the application request are assumed to apply for one periodic message, for which the channel is made.

$$T_{n,i} = (L_i, P_i, D_i, a_i, L_{\max})$$

The lower layer, which in this case is the transport layer, checks the request of the application layer to see if it can provide the service for the application layer or not. The protocol should make a schedulability test with all the known network parameters and if the task set Q consisting of requests (T_i) passes the test, it means that the transport layer can be able to provide the service

in which case it grants the request of the application layer, otherwise it rejects the request. When the request is accepted, it means that the channel is established. When an application is done with the channel and the transmission of the messages is over, the channel is closed to free resources for the other applications.

In this approach no resource is strictly assigned to a special application, but resources are scheduled so that in a special time, the application can use the resources, in this case the transmission path, to send data. Note that this special time may vary depending on the scheduling pattern.

Almost all related work has some kind of routing investigation in the method and they also try to apply flow and congestion control in the network. As the target is a point to point link, the path and connection specification is known in our case. The specialty of our case is that because we know the information about the network we can provide a special service. Although our service does best effort in retransmission, it guarantees that communication is done within the deadline and all timing constraints are guaranteed and fixed. In this special case with guaranteed timing constraints we try to give the best possible communication and the channel establishment process accepts as many channels as possible with all deadlines guaranteed.

5.3 Packet Format and Scheduling

The Scheduling algorithm which is going to be used in the proposed solutions to the problem is defined as a fully pre-emptive algorithm. As described in EDF [xiii] and [xvii] the algorithm assumes that tasks are pre-emptive in order to be schedulable. In our case, however, messages are not fully pre-emptive. We therefore suggest dividing the messages to some packets in order to make the messages pre-emptive by the end of each packet. In this way a packet itself can not be pre-empted but, messages become pre-emptive in the way that at the end of one packet, the message can be pre-empted by a packet from another channel.

There is a trade off in making the size of packets, between feasibility (pre-emption) and overhead of messages. Each packet needs some overhead and if a message is divided to more packets, more overhead is clearly needed and therefore more bandwidth is wasted on the headers, but messages become more preemptive.

It is therefore desirable to make these packets as small as possible unless they do not push an unbearable overhead. Messages of different channels can be of different sizes but for reasons of simplicity it is assumed that all packets have the same size equal to a value called maximum packet size (L_{max}). This maximum packet size is equal to pure data per packet plus the header and is decided by the system designer. In the time domain it is equal to one transmission time unit.

If the maximum packet size is known, the number of packets for each channel can be calculated as follows:

$$n_{packets,i} = \left\lceil \frac{L_i}{L_{\max}} \right\rceil$$

It is notable that when dividing a message to packets it is possible that the size of the last packet is less than the maximum packet size. In this case, it is assumed that a packet can also be extended by stuffing redundant bits, if it is not a full-size packet, so full-sized packets are made at all times. This assumption simplifies calculations.

5.4 Timings

The time it takes to transmit one packet into one channel ($T_{x,i}$) is equal to the length of the packet in bits divided by the bit rate of the channel:

$$T_{x,i} = \frac{L_{\max}}{BTR}$$

The time it takes to transmit a message is equal to the number of packets multiplied by the transmission time of one packet. So the original message transmission (without considering the retransmission) would have a transmission time of:

$$T_{x\_msg,i} = n_{packets,i} \cdot T_{x,i}$$

The transmission time of the Acknowledgement (T_{ACK}) is calculated in the same way, taking into consideration that the size of the acknowledgement packets can be different in different implementations and that this is a parameter decided by the system designer. It is assumed that a separate link has been developed for the acknowledgements, but that what is needed to be calculated is the time it takes for the acknowledgements to reach the destination. It has been used later on to calculate the time out.

While the packets are not fully preemptive and preemption can just take place at the end of a packet, a preemption delay should nevertheless be calculated. The preemption time is equal to the delay time a packet may experience until it gets the bandwidth. In the worst case this pre-emption time ($T_{preemption}$) is equal to ($T_{x,i}$), the transmission delay of one packet passing the physical link. In addition, there is another time that should be taken into account, namely the propagation delay. The time it takes for a packet to be transmitted over the physical link is showed by (T_{prop}) which is dependant on the physical link. In embedded networks, it is assumed that it is a known parameter to the system manager.

Therefore the round trip time of a packet, that is the time it takes for a packet to be transmitted to the receiver, and for the the receiver to acknowledge the packet and the sender to get the acknowledgement, can be calculated as below:

$$RTT_i = T_{preemption} + T_{x,i} + T_{prop} + T_{ACK} + T_{prop}$$

As there is a separate link for retransmission packets and the transmission time of packets are much larger in relation to the acknowledgements, there should be no pre-emption time for acknowledgement packets. It would simply never happen that an acknowledgement packet would wait for another acknowledgement to be transmitted.

After calculating required timings, a time out can be calculated for packet transmission. Since all packets have the same size, the timeout would be the same for all original and retransmission packets. This time out is used in the ARQ method ticking to inform the scheduling algorithm of a retransmission packet, if no acknowledgements show up.

Timeout calculation is an important issue in communication networks and as it is pointed out in the ARQ section: Short timeouts result in unnecessary retransmissions and longer ones add on delay before retransmission occurs. In case of real-time embedded networks when considering timing constraints, it is desired to have them short. Meanwhile system designers have full knowledge of the network parameters. Regarding this thesis work, where a point to point link is analyzed, it is important to consider the time it takes to get the acknowledgement plus a safety margin (T_{margin}) which is decided by the system designer between the expected ACK reception time and the time out moment, since networks with different characteristics and topologies can have different timing analysis which result in different delays and jitter. Studying the network's characteristics and its behavior shows that this safety margin can avoid unnecessary retransmissions. So the time out is defined as below:

$$TimeOut_i = RTT_i + T_{margin}$$

In this approach the time out timer starts as the scheduler choose a packet for transmission and the resource is assigned to the packet.

A detailed demonstration of the timing analysis can be seen in the Figure 3. Packet transmission timings are demonstrated for packet number two, which has a so called preemption delay. since the figure demonstrates one channel this packet needs to wait for packet number one to be transmitted first, but basically this packet can be any packet in any channel.

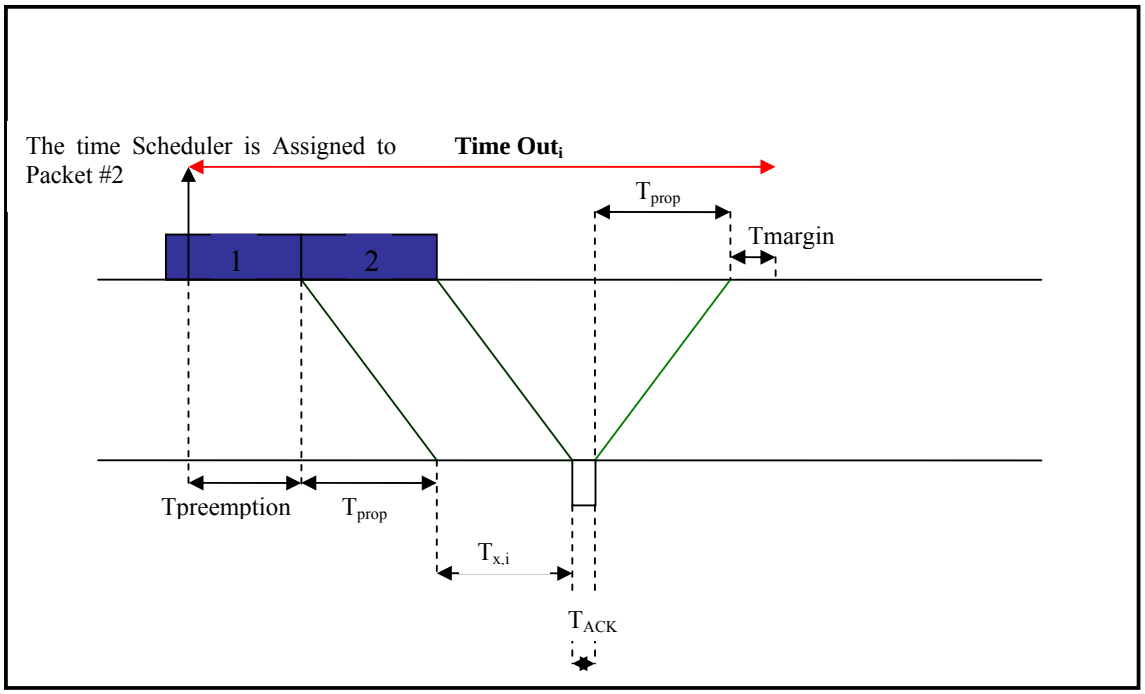

Figure 3: Timing analysis diagram for Packet transmission

6 Solutions

In this thesis two different protocols are suggested which both give best-effort service on providing reliable communication by retransmitting one packet per message. It is noticeable that by best effort is meant that in some cases the protocol may retransmit an unnecessary retransmission or in some cases there is the probability of retransmitting a packet which is not going to be erroneous. What is guaranteed by the algorithms here is the retransmission of one packet and as described before it can simply be extended to guarantee more according to the application's request. It is also guaranteed that deadlines will not be missed neither for the original message, nor for the retransmission packet. Although it can happen that the retransmission of one erroneous packet misses, the one which is retransmitted would not miss the deadline.

For reasons of simplicity, a point to point link is considered between two nodes and involvement in the multi-hop networks complexities is avoided in this approach, but the scheme is rich and can easily be extended to apply for embedded networks. We have also assumed that the link used for acknowledgements is a separate link just for this purpose and by adding one extra line which has a very light traffic (just ACKs instead of a link with a high bandwidth), the efficiency of the system improves to a very large extent.

In the ARQ method a positive selective repeat ACKS is assumed, but it is very open and the system designer can use another selective repeat method. What is important is that it is not going to change the worst case timing analysis and therefore the protocol would function with the same performance.

A simulation is followed on channel acceptability for each solution proposal. Which show using each of these algorithms how it affects acceptance of RT channels.

6.1 Method #1

The first suggestion to solve the problem of giving a best effort scheme for providing one packet retransmission is to schedule the retransmission packet with the original message's packets under EDF. This way, all packets would be transmitted under EDF, but the retransmission packet would be scheduled as the last packet to be sent. As it is seen in Figure 4, the last packet is reserved for the retransmission packet. By knowing about previously transmitted packets the possible erroneous packet can be sent with a higher probability on the latest chance. The method is suggested as follows:

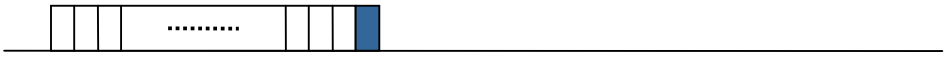

Figure 4: Method #1 retransmission's queuing pattern

It has been tried to overview all possible happenings and describe how the algorithm would act in each instance.

If an error occurs during transmission the error can be detected by the error detection algorithm in two ways. One is when the receiver gets the erroneous packet and informs the sender by a NACK packet and the other way is that when no acknowledgement is received by the sender showing either a lost ACK or a lost packet. In this situation the Timeout timer would ring to inform the sender of the error. In both ways (Negative ACK or Timeout) the special packet which is detected by the error detection algorithm and ARQ system would be scheduled as the last packet to be retransmitted, that is the packet which is reserved for retransmission purpose. This situation is showed in Figure 5.

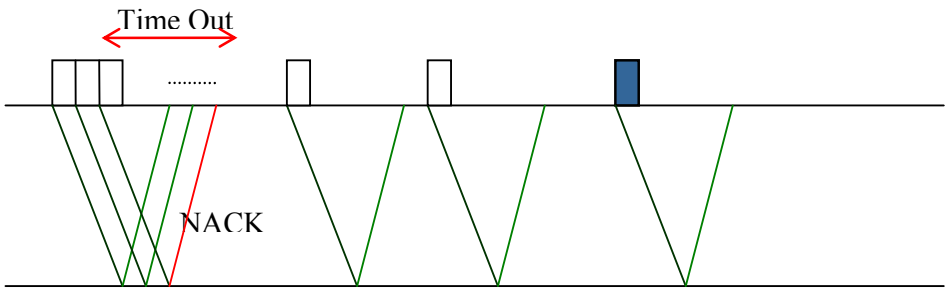

Figure 5: Method #1 retransmission decision

The worst-case scenario in this algorithm is when the scheduler reaches the reserved packet and still has some unacknowledged packets that have not timed out. At this point the scheduler needs to schedule a packet among those packets which have not been acknowledged and timed out yet and the algorithm is unsure of their health. That is, if the scheduler waits for acknowledgements it cannot keep the promise of being in time and not missing the deadline, but as the number of these wondering packets is likely to be small the scheduler can choose the erroneous one with a high probability. This is what makes this algorithm a best effort as the retransmission of the packet in the worst-case situation is based on a random or probability function.

The (worst-case) situation is demonstrated in Figure 6, showing the point that the algorithm decides to use a packet with an unknown entity. What can happen is that none of the packets are erroneous and all of them reach the destination safe and sound. In this case an unnecessary retransmission is carried out. If, on the other hand, one or more of the packets are erroneous, either the erroneous packet with a high probability is transmitted or retransmission of the real erroneous packet fails. Among all situations this one is tricky and it is still a weakness of the algorithm and makes the algorithm a best-effort algorithm.

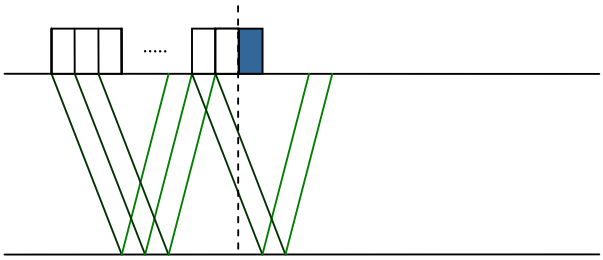

Figure 6: Method #1 retransmission decision on unknown packets

What is notable and worth investigating is how many packets could exist that the scheduler would be unsure about their safe arrival in the worst-case situation. Or in other words by the time the retransmission packet has to be scheduled, how many packets with no confirmed acknowledgment may exist. By knowing about the traffic pattern, a relative assumption of the mistake rate of the proposed algorithm may be achieved.

As the erroneous packet may not always happen in all messages, it can be suggested that if no ARQ occurs until the last packet transmission and all acknowledgments of the previous packets have been received, meaning that all packets of the message have been transmitted safe and sound, the reserved bandwidth can be freed and used by the scheduler for another channel. This would improve the performance of the algorithm.Figure 7 tries to show the situation and how the scheduler may face the freeing-up of the retransmission packet. This approach may have some implementation difficulties but results in a better performance.

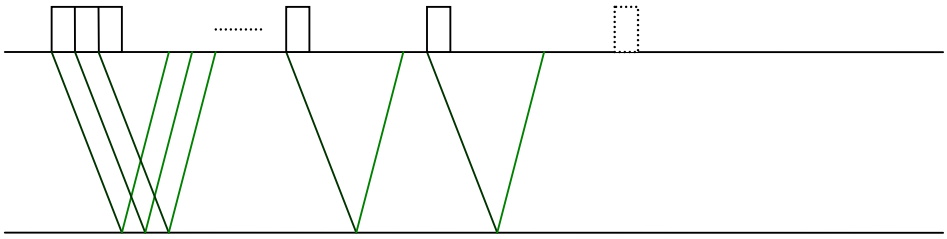

Figure 7: Method #1 freeing up retransmission's resources

6.2 Method #2

The second suggestion to solve the problem of giving a best effort scheme for providing one packet retransmission is similar to the first method and they therefore share the same weaknesses. However, the second method has a better performance because it suggests dividing transmission into two separate parts.

In the first part the original messages' packets can be scheduled with a shorter deadline and in the second part the retransmission packet/s can be scheduled. So for each channel there would be two periodic messages, as showed in Figure 8. The first message is the original message with a shorter deadline D'_i, and the second message is the retransmission packet with the remaining time ($D'_i = D_i - T_{Last\_Chance,i}$). As showed in the Figure 8, the original Deadline D_i is reduced to D'_i and in a small part of the channel's period, the retransmission packet is inserted with a period equal to the reduced time which is called $T_{Last\_Chance,i}$ here, since this is the last possible time that the retransmission packet can be transmitted in order not to miss the deadline.

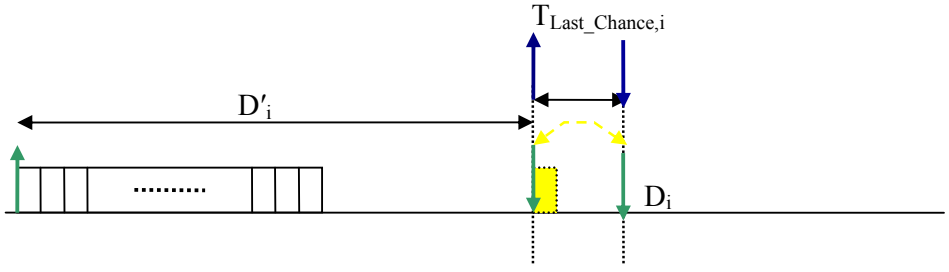

Figure 8: Method #2 The Idea of dividing transmissions

In fact, $T_{Last\_Chance,i}$ is the minimum time needed to send the retransmission packet. Then all packets of both sub-channels would be scheduled under EDF with the other RT channels. It means that for each logical RT channel, two sub-channels are made as shown in Figure 9: one for the original message and one for the retransmission.

$T_{Last\_Chance,i}$ is calculated as follow:

$$T_{Last\_Chanse,i} = T_{prop} + T_{Preemption} + T_{x,i}$$

$T_{x,i}$ is equal to the transmission time of one packet, T_{prop} is the propagation delay and $T_{preemption}$ is also one packet size transmission in time, which as previously discussed is applied to provide preemption for the scheduling algorithm.

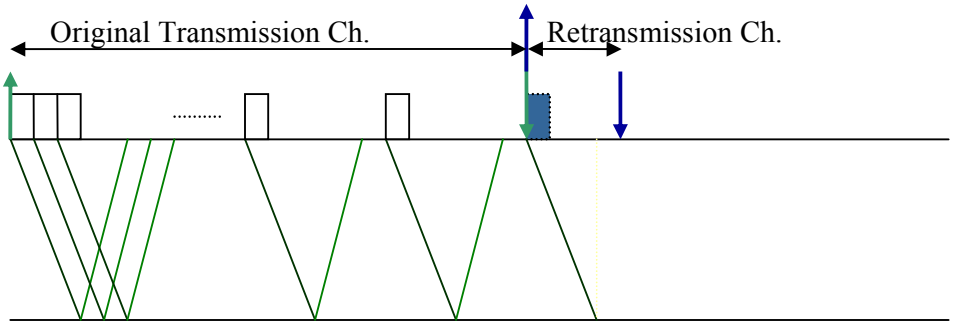

Figure 9: Method #2 separated Channels for transmission and retransmission

When having multiple channels, in the worst case of EDF algorithm, which is when all tasks start at the same time, it can happen that tasks have the same period and then it is the case that channels with equal parameters of period and start time would have the same deadlines both on the original message part and the retransmission sub-channel. Since these two channels are separated and EDF is applied on a task set which is composed of two sub-channels, the period and deadline of both sub channels would be the same. EDF would take care of schedulability of the task set, but what is clear is that since the deadline is minimal for retransmission sub-channel, one of them can not be scheduled. To avoid this situation in the channel establishment process, the time $T_{Last\_Chance,i}$ would be revised and the schedulability test would run once more for the accepted channels plus the new channel trying to establish a channel with the revised $T_{Last\_Chance,i}$. This last scenario is shown in Figure 10.

The process of channel establishment and revision of the times is as follow: for each channel establishment, the schedulability test is done once with the setting above (D'_i and $T_{Last\_Chance,i}$). Then, if the task set is schedulable the setting is assigned to the channel and channel is accepted, otherwise D'_i is to be reduced by one $T_{preemption}$ and $T_{Last\_Chance,i}$ is increased by one $T_{preemption}$ (i.e. $T_{Last\_Chance,i} = T_{Last\_Chance,i} + T_{preemption}$ and $D'_i = D'_i - T_{preemption}$), see Figure 10. Next, the schedulability test tests the set once more. This is done until the original task itself is no longer feasible. This is simply because if the original task is not feasible there is no need for the retransmission to be scheduled. A pseudo code is also given in the end of the simulation part,

which describes the channel-establishment process in more details and helps to understand the time revisions.

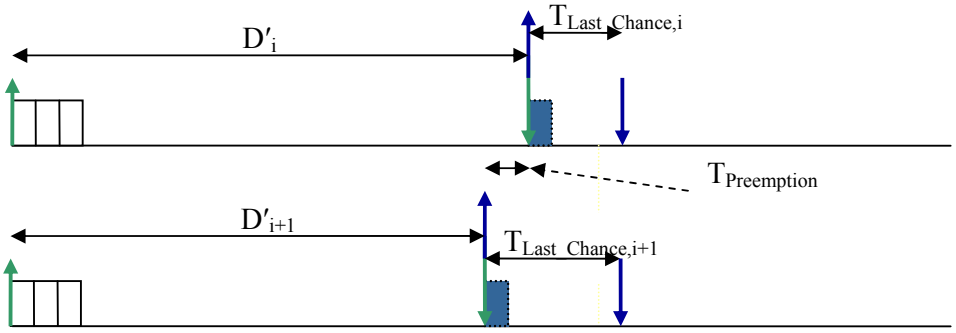

Figure 10: Method #2 deadline revision, revising $T_{Last\_Chance,i}$ and D'

In this method the chances of having un-received ACKs are fewer and a better performance can be achieved compared to the first proposed algorithm. Note that not all ordinary packets transmissions can be known by the retransmission time and still there may be un-received acknowledgments. Here there is also a possible chance of investigating how many unknown ACKs may appear. But the method would have a better ratio since the decision on the retransmission packet is made at a later time in the period interval of the message. As described later, the reliability of this method is higher than in the previous one.

Since Scheduling is done under EDF there is no need to worry about missing deadlines as long as the system is not overloaded. When deadlines are shortened a bandwidth equal to $T_{Last\_Chance,i}$ is reserved for each message and it has to be considered in the process of accepting a channel. When the transport protocol wants to establish a logical channel while accepting a request, it has to perform a feasibility test for EDF under the existing parameter which is shorter deadline (D'_i) for the message and the new periodic retransmission task.

6.3 Schedulability Aanalysis

Using EDF a necessary, but not sufficient, condition of feasibility for the traffic flow is that the utilization should not exceed ONE in all time intervals. In this case traffic consists of original message transmission and the possible retransmission packet. Considering the EDF scheduling algorithm [xiii], the utilization of a task set Q including retransmission that consists of tasks T_i and one packet retransmission for each message is calculated as below:

$$U = \sum_{i=1}^{Q} \left(\frac{T_{x\_msg,i} + T_{x,i}}{P_i} \right)$$

The schedulability analysis for task sets with deadlines less than periods can be performed by a concept named Processor Demand. This Method was first described in [xviii], and later used by [xix] to achieve the bringing down of the interrupt handling costs while using EDF.

In general, the processor demand of a task set Q which consists of tasks T_i in an interval (α,β) is the transmission time requested by the task/message activated instances in that interval that should be completed in that interval (α,β) In other words, it is the sum of all packets that their arrival times (a_i) and their deadlines are within the interval (α,β). In order to make the concept more clear, Figure 11 can help to get a better view of processor demand by showing an example.

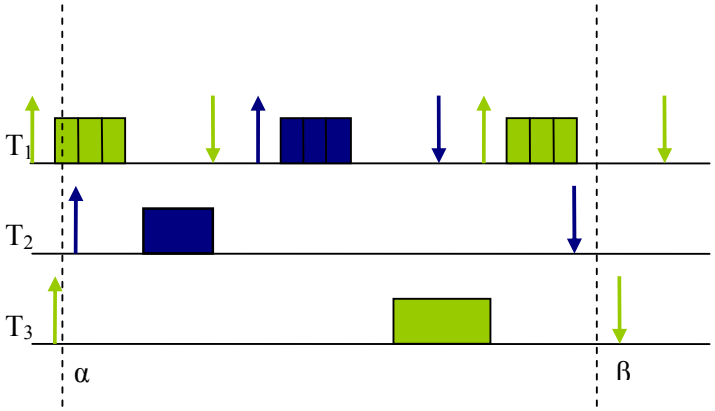

Figure 11: Processor Demand Declaration

A task set is guaranteed to be feasible if and only if in any interval of time the processor demand is less than or equal to the time available and the utilization of the task set is not greater than one. So the Processor demand function *f(s)* is calculated as the sum of all transmission times of all messages and the retransmission packets for all channels which have a deadline shorter than or equal to the point s (considering worst case in which all tasks start at time 0, the processor demand is simply calculated in the interval *(0,s)* in time)[xviii].

$$f(s) = \sum_{i=1}^{n} \left\lfloor \frac{S + P_i - D_i}{P_i} \right\rfloor \left(T_{x\_msg,i}\right)$$

Considering any interval of time the feasibility test can seem as a huge load of computation and therefore highly complex, but actually the processor demand function is a step function (if we consider the worst case that all tasks start at time zero), meaning that the value of the processor demand function in an interval is constant between two deadlines. Considering the interval *(0,s)* the value of the function increases when s crosses deadline d_j and remains constant until the next deadline d_{j+1}. This way the number of calculation can be reduced to some instances of time, which are the deadlines and it is proven that the schedule of tasks is repeated every hyper period. The hyper period is the least common multiple of all periods. Therefore if the feasibility test passes in a hyper period it can be guaranteed in all intervals because it is a repetition afterwards. So the feasibility test needs to be performed for some time instances (i.e. deadlines) in the hyper period.

There is also a hyper-period-reduction technique which is used in simulations. Using the technique the number of instances for which the feasibility test should be performed falls dramatically. In general, this reduced hyper period can be calculated as follows:

$$S^* = \frac{\sum_{i=1}^{n} (P_i - D_i)U_i}{1 - U}$$

A similar schedulability analysis is suggested in [xv] and [xx], with the difference that in these papers busy period is used to calculate processor demand and hyper period reduction. Notation and description used in [xv] and [xx], is more suitable for communication issues and it is intimately related to the concept of this thesis, therefore the suggested feasibility analysis is used in a similar way.

A task set including periodic tasks is schedulable if and only if the utilization of the task set is below one and:

$$\forall S \in \omega \qquad f(s) = \sum_{i=1}^{n} \left\lfloor \frac{S + P_i - D_i}{P_i} \right\rfloor \left(\left(n_{packets,i} + 1 \right) \cdot T_{x,i} \right) \leq S$$

Where

$$\omega = \left\{ d_j \middle| d_j \leq \min(S^*, Hyper\_Period) \right\}$$

In method number one the retransmission packet has the same period and deadline as the original message. However, if the original message is in the processor demand so is the retransmission packet and it is included in the feasibility test. In case of method number two for which the formula is re-defined as below, the second part belongs to the retransmission packet and if it is in interval $(0, \omega)$, it is by definition considered in the calculation. Otherwise, it is not calculated.

$$\forall S \in \omega \qquad \sum_{i=1}^{n} \left\lfloor \frac{S + P_i - D_i'}{P_i} \right\rfloor \cdot \left(n_{packets,i} \cdot T_x \right) + \left\lfloor \frac{S + P_i - T_{Last\_Chanse,i}}{P_i} \right\rfloor \cdot T_{x,i} \leq S$$

7 Simulation Analysis

In this thesis work the performance of the two methods in case of channel establishment is simulated. The ratio of number of real-time channels acceptance to the utilization is the main outcome of the simulations and at the end a comparison is made between two methods. The process of building a real-time channel is so that given a specific Task Set; the question is how many channels can be established successfully. In other words: how many of these real-time periodic messages can be sent.

For simulation simplicity, it is assumed that all applications generate periodic messages. The interesting point is that channels can be released at a specific moment and new channels can be established. Furthermore, because non-periodic messages are considered as periodic messages with a short period, even aperiodic messages can be covered in this scheme. It is obvious that these channels are dynamic and after the transmission is complete the channel is free and other communication can take place with the freed resource.

Task sets has been generated randomly with the following parameters:

A time period of 1000 have been assumed for all tasks and deadlines are randomly varied in different ranges, between:
500-1000: deadlines shorter than the period
800-1000: deadlines shorter than the period
1000 : deadlines equal to the period
500-1500: deadlines greater than the period
800-1200: deadlines greater than the period

All the task sets above have been tested for messages of different lengths: 10, 35 and 80 timeslots.

In the simulation of the second-proposal scheme all sub-channels scheduled together, and as EDF [xiv] schedulability analysis, consider the worst case in which all tasks start at the same time. The timing of the second sub channel was always changed with the method discussed in the simulation part. EDF assumes the same start time for all the retransmission packets in the second sub channel. To get a better understanding of the task sets schedulability, a pseudo code is presented below.

If all packets are acknowledged, there was no need for retransmission and the reserved packet can be freed by the scheduler. However, in simulation the worst case is considered (i.e. when all messages use the retransmission of one packet).

Feasibility Test Pseudo Code

Assume that we have a message set which consists of some periodic tasks (T_j), and then it is extended to a task set which is twice as big as the first one. In the new tasks set (T_i) (in which $i=2j$) all odd tasks are the original messages with the deadline D' and tasks with an even indices are retransmission packets with the deadline equal to T_{Last_Chance}. Accepted array is a list of accepted channels. It is notable that the end would increment the index of i and if it is decremented during the program reaching the end would increment it again and this way the same iteration can be repeated. The code would be as below:

```
Accepted = empty;
i:=1;

While (i < length of Task set)
    If  T(i) is schedulable
        If T(i+1) is schedulable
            Add T(i) to Accepted
        Else
            T(i).deadline= T(i).deadline - T_preemption;
            T(i+1).deadline= T(i+1).deadline + T_preemption;
        Endif
    Else
        i=i+2;
    Endif
End
```

A full range of simulation results have been done on the proposed algorithms. Some samples of simulation results are shown below in, Figure 12, Figure 13 and Figure 14.

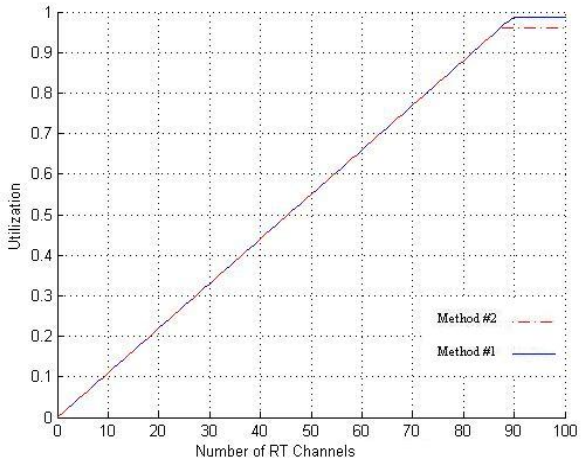

Figure 12: Simulation sample figure, L=10+1 packet retransmission, P=1000, D= 500-1000

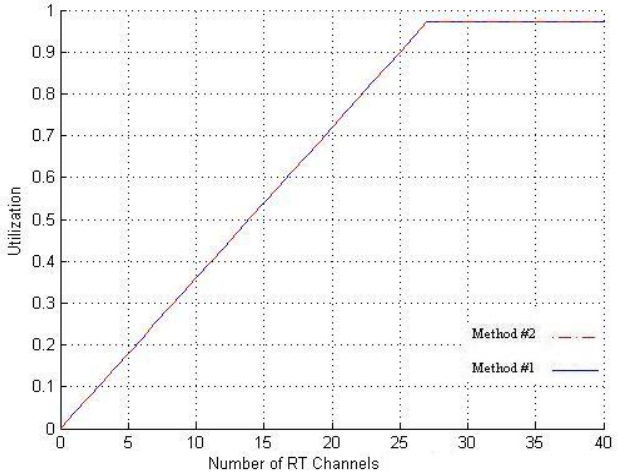

Figure 13: Simulation sample figure, L=35 +1 packet retransmission, P=1000, D= 1000

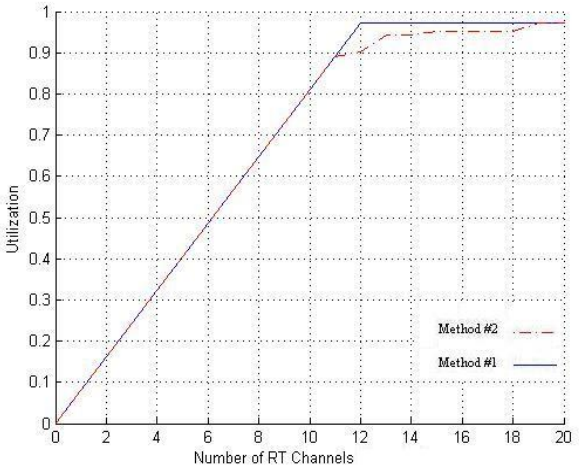

Figure 14: Simulation sample figure, L=80+1 packet retransmission, P=1000, D= 500-1500

As the simulation results show, In cases that the deadlines are shorter than the period (i.e. deadlines between 500-1000 and 800-1000), Method #1 can accept more RT channels than Method #2, and, as expected Method #1 can utilize the channel more than Method #2, which is quite reasonable. In order to give a visual reasoning the matter is verified as below:

In the case that deadlines are equal to the period, both methods have the same performance and the same ratio of channel acceptance in relation with utilization. In the case that they are randomly spread over tasks and can have shorter or longer deadlines (i.e. deadlines between 500-1500 and 800-1200) although the performance is almost alike, channel establishment process can differ somewhat for the two methods. The amount of accepted channels is the same and both methods utilize the channel. However, like the case of deadlines shorter than the period, method number one utilizes the channel faster.

Generally method number two is more reliable and chances of failed retransmission are lower. As the retransmission packet is sent as late as possible, there is a higher probability that all acknowledgements have been received by the sender. That means that even the last original packet of a message is acknowledged with a much higher probability and the retransmission is done under an almost full cover of all channels and the protocol has more information on transmitted packets. In other words, the retransmission is done more accurately, meaning that method number two is more reliable and therefore a better performance would be achieved when this method is implemented. Also, the overlaps between retransmission sub-channels are actually less than what is assumed in the worst case by EDF schedulability analysis. Worst case refers to a situation when all channels start at the same time, but in reality when channels are released and new ones established it is not always the worst case.

8 Conclusion

In this thesis, a framework has been studied to provide reliable transmission using an ARQ scheme in a real-time manner for embedded networks. The framework is demonstrated through two different methods over a point to point link. Using EDF scheduling, the framework inserts retransmission packets calculating the delay bounds of the original transmission. Strategies have been used to give a best effort on attempt to retransmit erroneous packets, trying to guarantee the deadlines of original and retransmission packets.

The methods presented would lead to a performance utilization of the real-time communication by providing retransmissions and a better quality of service. Basically a lower message error rate is provided by adding one retransmission packet per message while the promises of real-time deadlines are kept. Calculating the delay bounds of the messages, a retransmission packet is scheduled in the channel. This way, a notable reduction of message error rate is achievable with the cost of a reasonable overhead. The possibility and schedulability of the presented methods have been evaluated and some simulations have been done to verify the strength of the methods in accepting real-time channels. The evaluation shows the performance of the methods in accepting real-time channels in reality, and at last a comparison has been made between the two methods. The future works may be to extend the idea of providing retransmissions and QoS for multi-hop networks using an ARQ scheme.

9 References

[i] Network Working Group, G Fairhurst, Request for Comments (RFC) 3366, Aug . 2002.

[ii] L. L. Peterson and B. S. Davie, Computer Networks: A Systems Approach, 3rd Edition, 2003, ISBN 0-13-978-1558605145.

[iii] A. S. Tanenbaum, Computer Networks, 4th Edition 2003, ISBN 0-13-038488-7.

[iv] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, "Deadline scheduling for real-time systems – EDF and related algorithms", 1998, ISBN-13: 978-0792382690.

[v] D. Ferrari and D. Verma, "Real-time communication in a packet-switching network", proceeding of Second International Workshop on Protocols for High-Speed Networks, Palo Alto, Nov. 1990.

[vi] C. M. Aras, J. F. Kurose, D. S. Reeves and H. Schulzrinne, "Real-time communication in packet-switched networks", Proceedings of the IEEE, vol. 82(11), pp.122-139, Jan. 1994.

[vii] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks", Proceedings of the IEEE, vol. 83(10), pp.1374-1396, Oct. 1995.

[viii] J.F. Kurose, M. Schwartz and Y. Yemini, "Multiple-access protocols and time-constrained communication", ACM computing surveys, vol.16, pp.43-70, Mar. 1984.

[ix] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks". IEEE Journal on selected areas in communication, vol. 8(3), pp.368-379, Apr. 1990.

[x] S. Han and K. G. Shin, “A primary-backup channel approach to dependable real-time communication in multi-hop networks”, IEEE Transaction on Computers, vol. 47(1) , pp.46-61, Jan. 1998.

[xi] C. Chou and K.G. Shin, “Statical real-time channels on multi-access bus networks”, IEEE Transaction on parallel and distributed systems, vol. 8(8) , pp.769-780 Aug. 1997.

[xii] U. Zapata and P. Alvarez, “EDF and RM multiprocessor scheduling algorithms: survey and performance evaluation”, Technical Report CINVESTAV-IPN, Sección de Computación. Oct 2005.

[xiii] C.L Liu and J.W Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment”, Journal of the ACM, vol. 20, pp.46-61, Jan. 1973.

[xiv] G. C. Buttazo, Hard real-rime systems predictable scheduling and applications, 2nd Edition 2003, ISBN 0-387-23137-4.

[xv] M. Jonsson and K. Kunert “Reliable hard real-time communication in industrial and embedded systems”, Priliminary Version, Proceedings of the Third International Symposium on Industrial Embedded Systems (SIES2008), Jun. 2008.

[xvi] Q. Zheng and K.G. Shin. “On the ability of establishing real-time channels in Point-to-point packet-switched networks”, IEEE Transactions on Communications, vol. 42(2/3/4), pp.1096-1105, Feb./Mar./Apr. 1994.

[xvii] M. Spuri, G.C. Buttazzo, and F. Sensini. “Robust aperiodic scheduling under dynamic priority systems”, IEEE real-time systems symposium (RTSS ‘95), pp.210-219, Dec. 1995.

[xviii] S. K. Baruah, L.E. Roiser and R.R. Howell, “Algorithms and complexity concerning the preemptive schesuling of periodic , real-time tasks on one processor”, Journal of Real-Time Systems, vol. 2(4), pp.301–324, Nov. 1990.

[xix] K. Jeffay and D.L. Stone, “Accounting for interrupt handling costs in dynamic priority task systems”, IEEE real-time systems symposium (RTSS ‘93), pp.212-221, Dec. 1993.

[xx] M. Jonsson and H. Hoang “Switched real-time Ethernet in industrial applications – deadline partitioning”, The 9th Asia-Pacific Conference on communication (APCC2003), vol. 1, pp.76–81, Sept. 2008.