

A Real-Time Control Network Protocol for Embedded Systems Using Controller Area Network (CAN)

Paul Richardson[†], Larry Seih[‡], Peter Haniak, [‡]

Abstract

Controller Area Network (CAN) is a priority driven network protocol that can be found in a wide range of embedded systems (*e.g.: automotive, military, aerospace, manufacturing,...*). The networks in these applications are inherently real-time local area networks (LANs) whose correct performance is specified in terms of time constrained message transmissions. The messages in these types of systems have inter-arrival rates that are either periodic or sporadic. A significant issue for this type of network is the need to provide quality of service (QoS) support for long periods of time in harsh environments. Under these conditions, the network is prone to transient surges in network loading that arise from the *retransmission of corrupt data, sporadic message arrivals, and inaccuracies in electro-mechanical devices*. In this effort we propose a method to determine the feasibility of sporadic messages whose deadlines do not equal their inter-arrival rates using the CAN protocol in conjunction with the Earliest Deadline First (EDF) scheduling algorithm. Our approach is based on the extension of the determination of worst-case task execution in a multitasking system. A distributed mechanism that efficiently detects the occurrence of transient surges in network loading based on knowledge of worst case message transmission time is also described and several possible responses to transient surges are mentioned.

[†]Paul Richardson is an Assistant Professor in Department of Electrical and Computer Engineering, University of Michigan-Dearborn, Dearborn, MI 48139

[‡] Peter Haniak and Larry Seih are Research Associates at the U.S. Army Vetricons Tank-Automotive, Research, Development, and Engineering Center, Warren MI, 48397

1. Introduction

Control networks are a class of local area networks (LANs) found in many embedded real-time applications such as *automobiles, military platforms, manufacturing, and aircraft*. The primary role of these networks is to provide the data communications infrastructure for controlling the various devices and sub-systems in an embedded application. Control networks tend to have a limited geographic span (*e.g. several hundred meters or less*), strictly controlled inter-network access (*e.g. radio interface*), a fixed number of nodes, and a well-defined message set. These networks are often expected to provide long periods of uninterrupted service in an environment that is harsh and unpredictable.

Control networks are inherently real-time networks that must provide services for the reliable and timely delivery of messages. Critical messages are defined as those whose untimely delivery can lead to costly damage or serious injury (*e.g. messages concerned with vehicle braking, weapons control,...*). In order to avoid catastrophe, the network must continuously provide for the timely and reliable delivery of critical messages. Many of the messages in a control application have periodic inter-arrival rates (*e.g. digital control loops*). Other messages such as those generated by *users* or *events* occurring in the external environment have sporadic inter-arrival rates. It is often possible to bound the minimum inter-arrival rates of sporadic messages based on *apriori* knowledge available during the development cycle.

The requirement for continuous operations in harsh and non-deterministic surroundings gives rise to *transient surges* in network loading. Inaccuracies in electro-mechanical devices cause jitter in message inter-arrival rates. Extremes in *temperature, shock, vibration, EMI, power surges, etc.*, result in the bursty retransmission of corrupt data frames. If left undetected, transient surges can cause serious degradation in performance.

In this effort we propose a protocol that efficiently guarantees the time constraints for the transmission of both periodic and sporadic real-time messages. Our approach also realizes a distributed mechanism to detect the occurrence of transient surges occurring in the network. This mechanism is effective in that it does not add to the computational complexity of the protocol and provides a *necessary and sufficient* condition for transient surge detection during a message's critical time zone.

There are several alternatives for responding to a transient surge once it is detected. Two approaches originally developed for responding to transient faults on a uni-processor show particular promise for managing transient surges. The *slack-stealing* approach (*Ghosh, Davis, Thuel, etal*) relies on temporal redundancy to schedule tasks during transient faults. Value based scheduling (*Jenson, Locke, etal*) can consider multiple time-varying parameters such as *tasks value, system state* and *network loading* in arriving at scheduling decisions. The best approach to handling transient surges is a complex issue that depends on the severity of the surge and is beyond the scope of this effort.

1.1 Real-Time LANs – Previous Work

The ability to guarantee the time constraints of real-time message transmissions is fundamental to the success of a control network. A real-time message set $\{m_n\}$ is said to be feasible for a network if all of the time constraints for message transmissions can be met by the network. The characteristics of the media access control (MAC) protocol have a significant impact on feasibility. The MAC protocol is responsible for resolving contention between two or more nodes that simultaneously wish to transmit a data frame. A data frame is an indivisible (*non-preemptive*) unit of data transmission that consists of control information and data payload. Application messages are fragmented into one or more data frames for transmission and reassembled upon reception. In order to preserve QoS support, the QoS specified for different messages must be mapped to individual data frames.

Priority driven MAC protocols arbitrate control of the network based on a priority mechanism. Before the transmission of each data frame, a priority arbitration cycle determines which node will be allowed to transmit data. In this approach, message blocking, or the time that a high priority message must wait while messages of lower priority are transmitted is limited to one data frame[21]. Earlier studies have extended classical real-time scheduling algorithms such as *rate-monotonic scheduling* (RMS) and *earliest deadline first* (EDF) to priority driven network protocols. RMS assigns priorities based on *inter-arrival rates* and has been shown to be optimal amongst fixed-priority schedulers with respect to its feasible utilization

bound [16]. RMS is a special case of deadline monotonic scheduling (DMS) that assumes all tasks have periodic inter-arrival rates and deadlines are equal to inter-arrival rates. EDF prioritizes based on *impending deadlines* and has been shown to optimal amongst dynamic priority schedulers with respect to its utilization bound[16]. These algorithms can be applied to a priority driven network by using them to schedule messages locally (within a node). Each node then uses its local highest priority to arbitrate for control of the network. This results in the highest priority data frame pending in the entire network being transmitted. The use of RMS on the IEEE 802.5 protocol is described in [22] and on CAN in [24]. The use of EDF on IEEE 802.5 was proposed in [21] and on CAN in [26]. In [21] it was shown that priority driven protocols provide a bandwidth efficient solution for real-time networks with limited geographic span. A drawback back of the methods described in [21,22, 26] is the assumption that message deadlines are equal to their inter-arrival rates. In the case of messages with sporadic inter-arrival rates, the deadline is often significantly earlier than the minimum inter-arrival time. Applying the techniques used in the referenced approaches to this case results in an overly pessimistic feasibility analysis.

The interval during network arbitration cycles has been called *dead-space*, because the transmission of data is prohibited during this time. For networks with large geographic spans and high data rates, the signal propagation delay can be much larger than the time taken to generate a data frame. In this case, *dead-space* can result in poor bandwidth utilization efficiency[13]. For many embedded applications, the network's geographic span is typically on the order of hundreds of meters and *dead-space* is not significant [21]. In this case, the combination of classical scheduling techniques and priority driven MAC protocols provides optimal bandwidth utilization while guaranteeing all message time constraints.

Time division multiplexed (TDM) protocols have been studied in [13,26]. In this approach, network capacity is divided into time intervals which are allocated to individual nodes. A node voluntarily surrenders control of the network at the end of its time slice, eliminating the need for the network arbitration and the associated issue of dead-space. A drawback to this approach is that the worst-case message blocking time depends on the sum of allocated time slices. For the case when *dead-space* is negligible, this results in a feasible network utilization bound that is significantly lower than for the priority driven case.

A third class of MAC protocols exists that is based on random access to the network. These protocols can provide only probabilistic guarantees at pessimistically low network loading.

Practical methods of handling sporadic message arrivals and transient surges have included keeping network loading pessimistically low to minimize contention and the use dedicated data links instead of shared media networks to eliminate contention. The former method is based on the likelihood that enough network capacity exists to absorb any transient surges. This method results in inefficient bandwidth utilization and is naïve in that excess capacity does not guarantee timely message delivery. The later approach raises practical system issues with respect to fault tolerance, cost, and complexity. For example, distributed algorithms, including those dealing with fault tolerance, rely on data being available to multiple processors. Connecting multiple processors with dedicated data links means increasing the number of computer interfaces and the amount of cabling used. Increasing the number of computer interfaces increases system complexity and the risks associated with complexity. Increasing the amount of cabling is significant to applications that are sensitive to the cost, weight, and space claim of additional cabling (*e.g.: automobiles, aircraft, military vehicles...*)

1.2. Objective and Description of Approach

In this effort we propose the use of the earliest deadline first scheduling algorithm (EDF) in conjunction with the CAN protocol to guarantee time constraints of sporadic and periodic real-time messages.

In this scheme, the EDF algorithm is used on individual nodes to schedule messages. Each node uses its local highest priority message to arbitrate for media access, resulting in network-wide scheduling based on EDF. This approach relies on the determination of the worst-case transmission time for messages using EDF. This method is able to efficiently schedule sporadic messages whose deadlines do not equal their inter-arrival rates. The determination of worst-case message transmission time using EDF is derived the determination of worst-case task response time for multi-tasking on a uni-processor [14]. We also show how distributed detection of transient surges can be effected once the worst-case transmission times of each pending frame is known.

In section 2 we present the network model, section 3 describe the conditions for feasibility and section 4 describes the detection of transient surges. Section 5 summarizes this work and discusses future work in this area.

2. Control Network Model

Control networks are typically used to interconnect sensors, *motors*, *processors*, and other devices in an embedded system. The message set and topology of a control network are typically fixed at some point during the system design. As a result, it is possible to leverage *a priori* knowledge in order to bound network demand for a given real-time message set. The message set for a control network is fixed in terms of the number of messages and individual message *lengths*, *deadlines*, and *importance*. The topology of a control network is fixed in terms of the *number of nodes*, and *number of users*. Inter-network access is often strictly limited and typically handled by one or more specially designated nodes (*e.g. radio interface*). In the case of CAN, the restriction on geographic span eliminates *dead-space* as an issue. If the above conditions hold, we can limit our consideration of real-time message transmissions to the case of an isolated LAN with a limited geographic span and a fixed topology [2].

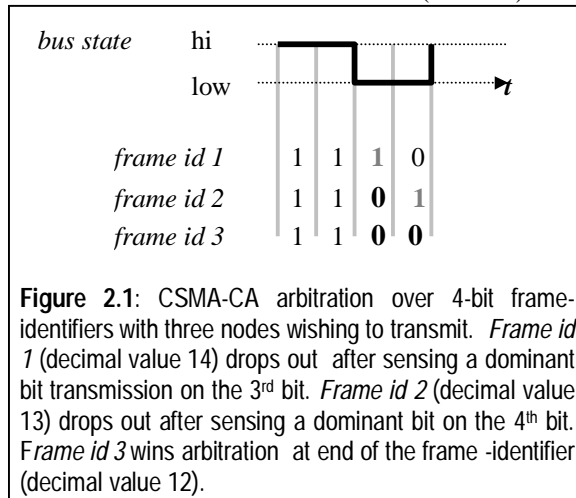
In order to guarantee the time constraints of a message set, it is necessary to have knowledge of the inter-arrival rates for each message stream. Periodic messages have regular or periodic inter-arrival rates while sporadic messages have irregular inter-arrival rates. Often, there is enough *a priori* information available during system design to bound their minimum inter-arrival rates. In our approach, we assume that the minimum inter-arrival rate for sporadic messages can be bound and that this can be used to assess feasibility. If a sporadic message(s) exceeds its specified inter-arrival rate then this results in a transient surge.

2.1 CAN Protocol

The CAN protocol specifies a bus topology, variable length payload of 1-8 bytes, and a maximum data rate of 1M bits/second [7]. The first segment of a CAN data-frame, the *frame-identifier* is a 29 bit (optionally an 11 bit) field that serves a dual-role as the *frame's address* and *priority*. In a system with a fixed message set, frame-identifiers can be used to uniquely identify the source and destination. For example, in an application involving *elevation control* of some device *X*, the *elevation* data frame would have a unique identifier that would imply the *elevation-sensor* for *X* as its source and one or more *digital controllers* as the destination. Thus, a system can have approximately 2^{29} or 2^{11} unique messages in its message set, depending on which version of CAN is used. This is only approximate because several identifiers are reserved for control purposes.

The CSMA-CA media arbitration also uses the frame-identifier to ensure that the pending data frame with highest priority is always transmitted first. In this approach a node senses the transmission media for any activity before transmitting a data frame (*carrier-sense*). If an active signal is sensed then another node is transmitting and the current node must wait until the transmission is complete. If no activity is sensed then the node is free to begin transmission of a data frame. Contention occurs when two (or more) nodes wish to transmit simultaneously. This most often occurs when two nodes have been waiting for a third node to complete a frame transmission. To avoid a collision of signals and lost data, contention is resolved using a *collision-avoidance* mechanism.

In order to describe collision avoidance and priority arbitration in CAN we first clarify (i) a restriction concerning *bit-length* and (ii) the physical realization of a logical bit transmission. The CAN protocol requires that a signal representing a single bit must be available to all nodes simultaneously. This implies that the *bit-length* must be greater than the geographic span of the network. At $1M\text{ bits/second}$, the *bit-time*, or the time to transmit a single bit is $1\mu s$. Assuming



the signal propagation speed through copper media is approximately 2×10^8 meters/second, the resultant ‘bit length’ is approximately 200 meters:

$$(2 \times 10^8 \text{ meters/second})(1 \times 10^{-6} \text{ seconds/bit}) \approx 200 \text{ meters/bit} . \quad (2-1)$$

This restricts the theoretical maximum geographic span of the network to less than 200 meters. In practice this is taken to be 100 meters or less. The span of the network can be increased by specifying a lower data rate. For a good number of embedded systems, spans of this magnitude are adequate.

CAN uses binary signaling with a *high* and *low* signal state and an idle signal-state that is defined as *high*. To transmit a logical ‘0’ bit, a node sinks the bus state to low for one bit time, this is called a *dominant* bit. To transmit a logical ‘1’ bit, the state of the line is left high for one bit time, this is called a *recessive* bit. Collision-avoidance begins when two or more nodes simultaneously begin to transmit the first bit of their frame-identifier. Bit-wise priority arbitration then proceeds over the length of the frame-identifier as follows: At any time during priority arbitration, a node transmitting a dominant bit (logical ‘0’) has a higher priority than any node transmitting a recessive bit (logical ‘1’). A node transmitting a recessive bit effectively monitors the bus state for one bit time. Upon detection of a dominant bit transmission, this node recognizes a higher priority frame and drops out of contention. This process is repeated over the length of the identifier. Given that the frame-identifiers are unique, only one node can be left in contention at the end of the bit-wise arbitration. This effectively realizes a priority arbitration mechanism wherein the identifier with the lowest numeric value has the highest priority. Figure 2.1 illustrates this concept with a simple example.

2.2 Dynamic Priorities on CAN

Scheduling messages using EDF requires the use of dynamic priorities whereas CAN inherently uses static priorities. It is straightforward to implement dynamic priorities on CAN by partitioning the frame identifier into two segments. One segment of the frame identifier is used to encode a frame’s dynamic priority and the other to preserve uniqueness. Since the priority of a frame is inversely proportional to its numeric value, we select the most significant bits in the frame to represent the priority and the least significant bits to provide uniqueness. This will result in all frames with unique priorities being ordered strictly by their priority segment. If two or more frames have equivalent priority segments then the bit-wise arbitration will use the remainder of the frame identifier to determine the final priority. The actual partition of the frame identifier depends on the number of messages in the message set and the resolution needed to attain feasibility. For example, using the 29-bit identifier, if the first 9 bits are used to encode dynamic priorities and the last 20 bits are used to preserve uniqueness then 512 priorities could be used to support over 1 million unique messages. In [15] *Lehoczky et al.*, describe the encoding of continuous values onto discrete priorities and its impact on schedulability. The example, figure 2.2 shows an 11-bit identifier partitioned into 4 and 7 bit segments to provide approximately 16 priorities and 127 unique messages types. Using the 29-bit identifiers is far less restrictive. For example 256 priorities could be used to support over 2 million unique messages.

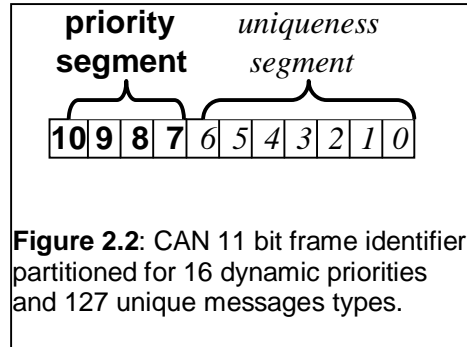


Figure 2.2: CAN 11 bit frame identifier partitioned for 16 dynamic priorities and 127 unique messages types.

3 Feasibility of Real-Time Messages

A real-time message set $\{m_n\}$, is said to be *feasible* for a network if all of the message time constraints can be guaranteed by the network protocol. Feasibility for a network depends on a number of factors such as:

- (i) the demand that a message set places on the network in terms of *message lengths* and *inter-arrival rates*,
- (ii) the capacity of the network, which is a function of the *data rate*, *signal propagation delay*, and MAC protocol,
- (iii) and the *local scheduling* of message transmissions on individual nodes.

For every type of message in a control network’s message set, $m_i \in \{m_n\}$, we assume that the length in bits, C_i^b , deadline, D_i , and inter-arrival rate, T_i are known. The inter-arrival rates used for sporadic messages

are estimates of their *minimum* inter-arrival rate. In the absence of transient surges, the deadlines, D_i , are assumed to be less or equal to the inter-arrival rates, T_i 's. A message is divided into one or more frames, depending on the length of its payload. A message's length in bits, C_i^b , is assumed to include the payload data, C_i^{b-data} and fixed overhead fields such as *frame identifier*, *frame check sequence*,.... In CAN, the overhead bits per frame, $F_{ovhd}^b = 66$ bits and the maximum frame payload is $F_{max}^b = 64$ bits. Thus, message length in bits is given by:

$$C_i^b = C_i^{b-data} + \left\lceil \frac{C_i^{b-data}}{F_{max}^b} \right\rceil F_{ovhd}^b \quad (3-1)$$

The selection of CAN as the underlying network establishes the MAC protocol as priority-driven CSMA-CA, and signal propagation delay to be negligible. If the data rate is set at $\gamma = 1M \text{ bit/second}$ then the bit time, $\gamma^{-1} = 1\mu s$.

The mapping of the demand approach to a priority driven network is straightforward. Abstractly, the transmission of a message is analogous to the execution of a task, with *message transmission time* correlating to *task execution time*. Message deadlines and inter-arrival rates map directly to task deadlines and inter-arrival rates. In the case of a message transmission, preemption is only allowed at frame boundaries. The transmission time of a message is its length in bits multiplied by the bit time, γ^{-1} , plus the propagation delay. Since propagation delay is negligible, the transmission time for a message type m_i is given by:

$$C_i = \gamma^{-1} C_i^b \quad (3-2)$$

The transmission time of the longest frame is given by:

$$F_{max} = (F_{max}^b + F_{ovhd}^b) \gamma^{-1} \quad (3-3)$$

Another important consideration is the worst case blocking time B , which is the longest time the transmission of a message can be delayed while a message of lower priority is transmitted. In CAN, this is equal to the transmission time of the longest frame, F_{max} :

$$B = F_{max} \quad (3-4)$$

Theorem 3.1: The worst case blocking time for any message using the CAN protocol is $B = F_{max}$.

By definition of the CAN priority arbitration, once a message begins to transmit, the only message that will be allowed to preempt it at a frame boundary, is a message of higher priority. Thus, the only time a message can be blocked is on its arrival, if a lower priority message is already transmitting. After the transmission of the current frame, this lower priority message will be preempted and the highest priority message pending will begin transmission. In general, the worst-case scenario for blocking is that the lower priority message has just started transmitting a frame of maximum length as the higher priority message arrived, thus $B = F_{max}$. ■

Using EDF, we determine the feasibility of a message set by extending the demand-based approach for multi-tasking on a uni-processor, described in [22]. The demand of a task (or message) at any time is the ratio of its response time to its deadline. If the worst-case response time of each task can be determined then an assessment of feasibility can be made. A task's worst-case response time, d_i , is recursively derived then compared to its deadline. If for each task τ_i , $d_i \leq D_i$, then the set of tasks is feasible, otherwise it is not. By substituting the message transmission times, C_i , and worst case blocking time, B , for CAN into the demand calculation, we can determine d_i for each message stream. The procedure for determining the worst-case response time for a message transmission is as follows:

(i) Determine the length of the synchronous busy period preceding the 1st processor idle time.

Let L represent the length of synchronous busy period preceding 1st processor idle time. We compute L iteratively in the following manner:

$$\text{The initial estimate of } L \text{ is determined as } L^0 = \sum_{i=1}^n C_i \quad (3-5)$$

$$\text{Continue computing } L^m \text{ iteratively as } L^{m+1} = \sum_{i=1}^n \left\lceil \frac{L^m}{T_i} \right\rceil C_i, \quad (3-6)$$

until $L^m = L^{m+1}$, when $L^m = L^{m+1}$ then set $L = L^m$.

(ii) Determine the network demand of each message in message stream m_i at intervals of their deadline for all time intervals up to L .

Let m_i represent the i^{th} message stream. Then for m_{ij} , the j^{th} arrival of m_i , its deadline will occur at:

$$e_{ij} = jT_i + D_i \quad (3-7)$$

Let d_{ij} represent the network demand at e_{ij} , then from [22], we have:

$$d_{ij} = B + \sum_{D_i \leq t} \left(1 + \left\lceil \frac{e_{ij} - D_i}{T_i} \right\rceil \right) C_i \quad (3-8)$$

Then while $e_{ij} < L$, determine d_{ij+1} . If $d_{ij} > e_{ij}$ then the message stream is not feasible.

The worst-case transmission time for a message in m_i occurs at $d_i = \max(d_{ij})$. If $\forall i: d_i \leq D_i$, then the set of messages, $\{m_n\}$ is feasible, otherwise for any $d_i > D_i$, the message stream m_i is not feasible. The network demand for m_i is X_i , the ratio of its worst-case response time to its deadline:

$$X_i = d_i / D_i. \quad (3-9)$$

The nominal network demand is given by:

$$X = \max_i \left(\frac{d_i}{D_i} \right). \quad (3-10)$$

In the absence of transient surges, if $X \leq 1$ then the message set is feasible.

4. Detecting Transient Surges

A transient surge in network demand, $TS(t_0, t_f)$ is a temporary increase in *nominal network demand* that occurs at some time t_0 and dissipates at time t_f . Nominal network demand, $X_{nom} = X_{max}$, is determined by the message transmission times, C_i , deadlines, D_i , and inter-arrival rates, T_i specified for the message set $\{m_n\}$ as described earlier in this section. A surge in network demand can occur as the result of messages that exceed their *specified* or *estimated* inter-arrival rates and/or data frames that have to be re-transmitted because of corruption. If a message of type m_i exceeds its inter-arrival rate at time t_0 then the impact to network demand at t_0 can be determined by adding a message whose arrival time is t_0 , transmission time is C_i and deadline is D_i . If several messages simultaneously exceed their inter-arrival rates then the impact to network demand is determined by repeating this step for each message. If the k^{th} frame of message type m_i has to be re-transmitted, then its transmission time is augmented by the transmission time of the frame, $F_i(k)$:

$$C_i = C_i + F_i(k). \quad (4-1)$$

The deadline for the message remains the same. The impact to network demand for multiple simultaneous frame retransmissions is determined by repeating (2-10).

The transient surge $TS(t_0, t_f)$ is said to have dissipated at time t'_f when network demand returns to nominal levels. This will occur given that the network has some level of spare capacity which can be used to transmit the additional network loading. If the effective network demand, X_{eff} , during a transient surge exceeds 100% then a transient overload has occurred and some message(s) will unavoidably be late. If an additional surge in network loading occurs at time t_b , where $t_0 < t_b < t_f$, then the resultant increase in network demand is treated as a single transient surge $TS(t_0, t'_f)$, where t'_f is the time that the surge dissipates.

4.1 Detection of Transient Surges

If we assume that a network is operating nominally without any permanent network faults, then the occurrence of a transient surge can be detected in the following way. The critical instant is defined as the

moment when a message will have its worst-case transmission time. At the critical instant, let a transient surge occur as a message m_{ij} that either (i) exceeds its specified inter-arrival rate or (ii) retransmits a corrupt frame. At the critical instant, this will cause m_{ij} to exceed its worst-case response time, d_i . Conversely, the violation of d_i for m_{ij} at the critical instant is an indicator that a transient surge has occurred. Since scheduling on a network is distributed, the violation of d_i will only be noticed locally. However a message, $m_{x,y}$, on any node whose priority is less than m_{ij} 's will also violate its worst case response time, as a result of m_{ij} 's extended response time. Thus, any nodes with pending messages of lower priority than m_{ij} can detect the occurrence of the transient surge. Any messages that have a higher priority than m_{ij} will not be affected by the increase in m_{ij} 's response time. A node whose pending messages all have priorities greater than m_{ij} will not detect the transient surge. However, none of the response times of messages pending on this node will be affected by the surge. Using similar reasoning, a violation of d_i at any time indicates that a transient surge has occurred. These ideas are formalized below.

Assume that the network is operating normally and that permanent network faults do not occur. Let the message set $\{m_n\}$ be ordered in descending order by message deadlines and let p_{ij} denote the priority of message m_{ij} .

Theorem 1: If the EDF scheduling algorithm is used to schedule the message set $\{m_n\}$, then at the critical instant the message $m_{x,y}$, where $p_{i,j} < p_{x,y}$ can cause a transient surge can if and only if d_i is violated.

→ Let $m_{x,y}$ cause a transient surge at the critical instant, either by exceeding its specified inter-arrival time or by retransmission of a corrupt frame. Since $p_{ij} < p_{xy}$, the extra transmission time of m_{xy} will delay the transmission of m_{ij} . If the k^{th} frame of $m_{x,y}$ is retransmitted, the additional frame transmission implies that the modified worst case response time for m_{ij} is:

$$d'_i \geq d_i + f_{xy}(k). \quad (4-1)$$

In the case of an unspecified arrival of $m_{x,y}$ the effective network demand is modified to account for its arrival as described earlier. The modified worst-case response time for m_{ij} will be:

$$d'_i \geq d_i + C_x. \quad (4-2)$$

In both cases, the worst-case response time of m_{ij} will be violated.

← If the network is operating normally then d_i is the worst-case response time for any message m_{ij} . If no permanent faults have occurred, then a violation of d_i indicates that a transient surge has occurred. Since EDF is used, the source of the violation must be a message $m_{x,y}$ where $p_{x,y} > p_{i,j}$. ■

Corollary 1: For any message m_{ij} , the violation of d_i at any time indicates that a transient surge has occurred.

The proof follows from the sufficiency condition of theorem 1. ■

Corollary 2: If a transient surge occurs as a result of an unspecified arrival or frame retransmission of $m_{x,y}$ then d_i for any message m_{ij} , with priority $p_{x,y} < p_{i,j}$ will not be affected.

This result follows from the use of a priority driven protocol. Since $p_{x,y} < p_{i,j}$ then the transmission of m_{ij} will proceed unaffected by a transient surge occurring for $m_{x,y}$. Then the d_i for m_{ij} will not be violated. ■

The slack stealing approach as described in [8,9,19] provides an effective means of managing a transient surge that does not cause the system to overload. If the transient surge results in an overload, then a scheduling mode change to value based scheduling [10,17] would help to ensure that critical messages are transmitted in a timely manner. The optimal response to a transient surge depends on its severity which implies some type of prediction. The estimation of the severity of a transient surge and enacting a response to a transient surge is beyond the scope of this effort.

5. Summary

Local area networks are gaining increased popularity in many embedded real-time systems because they can help to improve a system's scalability and reliability. Often these systems have both sporadic and aperiodic messages. This efforts propose an efficient means to guarantee the time constraints of these messages for the case when deadlines do not equal arrival rates. This approach relies on the determination of the worst-case transmission time using an extension of the demand-based approach for determining the

worst-case task response time[22]. We also describe a distributed mechanism for detecting transient surges in network loading based on knowledge of worst-case message transmission time. This mechanism provides a necessary and sufficient condition for detecting transient surges and is efficient in that it does not increase computational complexity of EDF.

In follow on efforts we are studying the response to transient surges using the slack stealing and value based scheduling mechanisms referenced earlier. The optimal response to a transient surge depends on its severity and the resultant effective network loading. A key challenge of this work is devising a dynamic and computationally efficient mechanism to determine the severity of a transient surge.

Acknowledgement: This work was funded by the U.S. Army Vetronics Institute, a U.S. Army initiative to promote research collaboration in vehicle electronic systems.

References

- [1] S. A. Aldarmi and A. Burns, "Time-Cognizant Value Functions for Scheduling Real-Time Systems", *YCS 306*, Department of Computer Science, University of York, October 1998
- [2] K. Arvind, J Stankovic, *A LAN Architecture for Communications In Distributed Real Time Systems*, IEEE Real Time Systems, Vol 3, No. 2, May 1991
- [3] Audsley, N.C.; Burns, A.; Richardson, M.F.; Wellings, A.J.; "Hard Real-Time Scheduling: The Deadline Monotonic Approach", *Proceedings of the 8th IEEE Workshop on Real-Time Operating Systems and Software*, Oxford, UK, May 1991
- [4] Burns, A., Davis, R., Punnekkat, S., "Feasibility Analysis of Fault-Tolerant Real-Time Task Sets", 8th Euromicro Real-Time Systems Workshop, June 1996
- [5] A. Burns, D. Prasad, "Value Based Scheduling of Flexible Real-Time Systems for Intelligent Autonomous Vehicle Control", Proc. of 3rd IFAC Symposium on Intelligent Vehicles, March 1998
- [6] G. Butazzo, "Hard Real-Time Computer Systems: Predictable Scheduling Algorithms and Applications", Kluwer Academic Publishers, 1997
- [7] CAN Specification, 2.0, Phillips Semiconductors, 1991
- [8] Davis, R. I., Tindell, K., Burns, A. "Scheduling Slack Time in Fixed Priority Preemptive Systems" IEEE Real Time Systems Symposium, December, 1993
- [9] Goshe, S., Melhem, R., Mosse', D., "Fault Tolerant Rate Monotonic Scheduling" *Journal of Real Time Systems*, 15 (2): 149-181, Sept 1998
- [10] Jenson, E.D., C.D. Locke, and H. Tokuda, "A Time-Value Driven Scheduling Model for Real-Time Operating Systems", *Proc. Symp. on Real-Time Systems*, November 1985, IEEE Computer Society
- [11] D. Jenson, "Eliminating the 'Hard'/'Soft' Real-Time Dichotomy", *Embedded System Conference*, Apr 94, pp84-96.
- [12] Kamat, S., Malcom, N., Zhao, W., *The Probability of Guaranteeing Synchronous Real-Time Messages with Arbitrary Deadlines in FDDI Network*, Proc. of IEEE Real-Time Symposium, Dec 1993.
- [13] S. Kamat, W. Zhao, *Real Time Performance of Two Token Ring Protocols*, Advances In Real Time Systems, Prentice Hall, 1996
- [14] Mark Klien, Thomas Ralya, Bill Pollak, Ray Obenza, Michael Harbour, 1994, *A Practitioners Handbook for Real-time Analysis*, Kluwer Academic Publishers
- [15] Lehoczky, J.; Sha, L; Strosnider, J.K.; Tokuda, H.; "Fixed Priority Scheduling Theory for Hard Real-Time Systems", *Foundations of Real-Time Computing: Scheduling and Resource Management*, Boston, MA, Kluwer Academic Publishers, 1991
- [16] C. Liu, J. Layland, *Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment*, ACM 20(1):46-61, 1973
- [17] Locke, *Best-Effort Decision Making for Real-Time Scheduling*, PhD thesis, Carnegie-Mellon University, 1986

- [18] Mejia-Alvarez, Melhem, Mosse, “*An Incremental Approach to Scheduling during Overloads in Real-Time Systems*”, IEEE Real-Time Systems Symposium 2000,
- [19] Ramos-Thuel, S. “Enhancing Fault Tolerance of Real-Time Systems Through Time Redundancy”, Ph.D. Thesis, Carnegie Mellon University, May 1993
- [20] Ramos-Thuel, S., Stosnider, J., “Scheduling Fault Recovery Operations for Time-Critical Applications”, 4th IFIP Conference on Dependable Computing for Critical Applications, Jan 1995
- [21] Richardson, *Seih*, “Real-Time LANs in Combat Vehicles: Feasibility Criteria For Non-Preemptive Messages and Multiple Message Streams Originating From Individual Nodes”, *IEEE ICCCN '99*, Boston, MA, Oct 99
- [22] Stankovic, Spuri, Ramamritham, Buttasso, “Deadline Scheduling for Real-Time Systems” , Kluwer Academic Publishers, 1998
- [23] Strosnider, Lehoczky, and Sha, *Advanced Real Time Scheduling Using IEEE802.5 Token Ring*, Proc IEEE Real-Time Symposium, Dec 1988
- [24] K. Tindell, H. Hansson, A.J. Wellings, “Analyzing Real-Time Communications, Controller Area Network”, Proc. Real-Time Systems Symposium, pp 259-263, Dec. 1994
- [25] K. Tindell, A. Burns, A.J. Wellings, “Calculating Controller Area Network Message Response Times”, *Control Engineering Practice*, vol.3, no. 8, pp1163-1169, 1995
- [26] S. Zhang , A. Burns, *An Optimal Synchronous Bandwidth Allocation Scheme for Guaranteeing Synchronous Messages with Deadlines with the Timed Token MAC Protocol*, IEEE/ACM Transaction on Networking, Vol 3, No. 6, pp. 729-741, Dec. 1995
- [27] K. Zuberi, K. Shin, “Non-Preemptive Scheduling of Messages on Controller Area Networks for Real-Time Control Applications”, Proc. Real-Time Technology and Applications Symposium, pp240-249, May 1995
- [28] K. Zuberi, K. Shin, “Scheduling Messages on Controller Area Network for Real-Time CIM Applications”, IEEE Trans. Robotics and Automation, pp 310-314, April 1997