EECS 571 Principles of Real-Time Embedded Systems
Lecture Note #18:

Controller Area Network (CAN)
Non-Preemptive Scheduling of Messages on CAN

- The Controller Area Network (CAN) protocol: contention-based multi-master network (up to 1 Mbps) with high schedulable utilization and reliability, prioritized bus access, and good reconfigurability.

- Scheduling messages on CAN
  - Workload characteristics.
  - MTS algorithm.
Real-Time Control Systems

- Main devices are controllers (CPUs), sensors, and actuators (drives).
- Detect events and respond to them.
- Real-world events are aperiodic in nature.
- But if they occur frequently enough, can use periodic polling of sensors, e.g., servo control of drives.
- But if event occurs *sporadically*, periodic polling is a waste of bandwidth, e.g., temperature-too-high event.
- For such events, *smart sensors* are appropriate.
  So, ∃ two requirements to meet:
  - Fast response to sporadic messages.
  - Efficient handling of both periodic and sporadic messages.
- CAN satisfies both requirements.
The CAN Protocol

- Contention-based multimaster bus with a special bus acquisition algorithm.

- Wired-OR (or wired-AND) bus.

- Each message has an identifier (ID) – reflects message priority.

- When bus becomes free, each node transmits (bit-by-bit) the ID of its highest-priority message.

- After writing each bit, each node reads the bus.

- If bit read different from bit written, node drops out of contention.

- In the end, only one winner which transmits its message.

- Two CAN message formats: standard (11-bit ID) and extended (29-bit ID).

- This algorithm implemented in low-cost bus interface chip.

Kang Shin (kgshin@eecs.umich.edu)
CAN Message Format

<table>
<thead>
<tr>
<th>SOF</th>
<th>Identifier</th>
<th>Control</th>
<th>Data</th>
<th>CRC</th>
<th>Ack</th>
<th>EOF</th>
</tr>
</thead>
</table>

SOF: Start of Frame  
CRC: Cyclic Redundancy Code  
EOF: End of Frame

**ID field:** Serves two purposes:

- Controls bus arbitration.
- Describes the meaning of the data (message routing).

**Routing:**

- Suppose ID is %xxxxxx10110.
- All nodes desirous of receiving this message, will set *filters* in their interface chips.
Advantages of CAN

- Global bus arbitration despite being distributed.
- Fast bus access to highest priority message.
- Prioritized bus access.
- Minimal priority inversion.
- Easily implemented bus acquisition algorithm – cheap interface chips.
- Reconfiguration flexibility – nodes can be easily added or removed.
Workload Characteristics and Message Scheduling

**Hard periodics:** Commonly found in servo control of drives. Deadline may be less than the period.

**Hard sporadics:** Used for notification of events. Each has a *minimum interarrival time* (MIT).

**Non-real-time aperiodics:** Examples include diagnostic information, device status, device setup, etc.

**Message scheduling:**

- For CAN: message scheduling = design of ID.
- Non-preemptive scheduling under release-time and deadline constraints is NP-hard ⇒ *Mixed Traffic Scheduler* (MTS) based on ED.
Fixed-Priority Scheduling on CAN

- CAN nodes send messages as follows:
  - Node assigns an ID to a message and transfers the message (with ID) to bus interface chip.
  - Chip will contend for the bus autonomously.
  - Once message transferred to chip, its ID will remain fixed unless processor changes it.

- Fixed-priority scheduling is a natural choice.
  - Use *deadline monotonic* scheduling.
  - Each message has a fixed priority according to the tightness of its deadline.
  - This fixed priority forms the message ID. It also uniquely identifies message for reception purposes.

- But to get greater utilizations, try dynamic scheduling . . .
Earliest-Deadline Scheduling on CAN

- Design of ID:

<table>
<thead>
<tr>
<th>priority</th>
<th>deadline</th>
<th>uniqueness</th>
</tr>
</thead>
</table>

  - *Deadline* is logical inverse of message deadline.
  - *Uniqueness* (unique code assigned to each node) distinguishes messages with same deadlines.
  - *Priority* is a 1-bit field: 1 for real-time messages, 0 otherwise.

- Problem: absolute deadlines keep increasing.
- One solution: *slack time* (time to deadline). But,

  P1. Slack time changes with every clock tick, so must be updated before each arbitration round ⇒ too time-consuming.
  P2. Messages in typical workloads may have a wide range of laxities – but not enough bits in ID.

- P2 makes ED impractical for CAN:
  - Laxities can range from 100’s of microseconds (high-speed drives) to several seconds (temperature readings) ⇒ need ~20 bits (µs granularity).
  - Can use CAN extended format, but this wastes 20–30% bandwidth.

Kang Shin (kgshin@eecs.umich.edu)
MTS

- MTS gives high utilization (like ED) while using 11-bit IDs (like DM).

- Observation: 11 bits are too many for DM — a few bits will remain unused.

- Use these bits to enhance schedulability by using ED.
MTS – Solution to P1: Time Epochs

- Use actual deadlines (instead of slack time), but express them relative to a periodically increasing reference called the start of epoch (SOE).
  - Deadline values stay fixed for duration of epoch.
- A periodic process wakes up every $\ell$ seconds (length of epoch) and updates IDs.
- Deadline field for message $i$
  - $d_i - \text{SOE}$
  - $d_i - \lfloor \frac{t}{\ell} \rfloor \cdot \ell$
  - If want $x\%$ of CPU time spent on updates, $\ell = \frac{n}{xM \times 10^6}$.
- Disadvantage: need more bits in deadline field: $m = \log_2(\ell + D)$, where $D$ is the largest $(d - r)$.

---

Kang Shin (kgshin@eecs.umich.edu)
MTS – Solution to P2

• Define two classes of messages:
  
  **High-speed:** The tightest deadline messages in the workload.
  
  **Low-speed:** The remaining messages.

• *Goal:* improve schedulability of high-speed messages since they tend to use several times more BW than low-speed messages.

• Use ED for high-speed and DM for low-speed messages.

• High-speed messages:

<table>
<thead>
<tr>
<th>1</th>
<th>deadline</th>
<th>DM priority</th>
</tr>
</thead>
</table>

  - MSB is 1.
  - *Uniqueness* field is 5 bits (32 high-speed messages).
  - remaining 5 bits not enough to encode the deadlines (relative to the latest SOE) . . .
MTS – Solution to P2: Quantized Deadlines

- Quantize time into *regions*.
- Encode deadlines according to which region they fall in.

\[
\begin{align*}
\text{SOE} & \quad \text{end of epoch} \\
\hline
\end{align*}
\]

\[
00 \quad 01 \quad 10 \quad 11
\]

\* Length of a region is \( l_r = \frac{\ell}{2^m - 1} \), where \( m \) is the length of the deadline field (5 in this case).
- What if two deadlines fall in same region (and quantize to same value)?
  - Use DM-priority of a message as its uniqueness code.
  - This makes MTS a hierarchical scheduler.
• Use DM scheduling for low-speed messages.
• Use fixed-priority for non-real-time messages; priorities assigned arbitrarily.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM priority</td>
<td></td>
</tr>
</tbody>
</table>

(Low-speed)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed priority</td>
<td></td>
</tr>
</tbody>
</table>

(Non-real-time)

• So MTS allows:
  o 32 high-speed messages (periodic or sporadic).
  o 512 low-speed messages (periodic or sporadic).
  o 512 non-real-time messages.
Schedulability Conditions – ED and DM

• Non-preemptive ED (Zheng and Shin ’94; modified for relative phase offsets):

1. \( \sum_{j=1}^{n} C_j/T_j \leq 1. \)

2. \( \forall t \in S, \sum_{i=1}^{n} \left[ \left( t - d_i - \phi_i \right)/T_i \right]^+ C_i + C_p \leq t, \)
   where \( S = \bigcup_{i=1}^{n} S_i, S_i = \{d_i + nT_i : n = 0, 1, \ldots, \}
   \left( t_{max} - d_i - \phi_i \right)/T_i \}, \) and \( t_{max} = \max\{d_1, \ldots, d_n, \)
   \( C_p + \sum_{i=1}^{n} (1 - d_i/T_i) C_i \}/(1 - \sum_{i=1}^{n} C_i/T_i) \}.\n
   - \( T_i, C_i, d_i \) are the period, length, and deadline of message \( i. \)
   - \( C_p \) is the length of the longest possible packet.
   - \( \left[ x \right]^+ = n \) if \( n - 1 \leq x < n, n = 1, 2, \ldots, \) and \( \left[ x \right]^+ = 0 \) for \( x < 0.\)

• Non-preemptive DM. Message \( i \) schedulable if

\( \exists t \in S, \sum_{j=1}^{i-1} \left[ \left( t - \phi_j \right)/T_j \right] C_j + C_p \leq t, \)
where \( S = \{\) set of all release times of messages \( 0, 1, \ldots, i - 1 \) through time \( d_i - C_i \} \cup \{d_i - C_i \}, \)
and \( \phi_j \) are the \textit{relative} phase offsets.
• **High-speed** messages – worst-case situation created when

  1. worst possible traffic congestion.
     - release all messages at same time.
  2. worst possible deadline encoding.
     - maximize number of messages which get priority over message $i$.
     - occurs when deadline-to-start of $i$ coincides with the start of a region.

\[
d_k - C_k \quad d_i - C_i \quad d_j - C_j
\]

(a) start of region end of region

\[
d_k - C_k \quad d_i - C_i \quad d_j - C_j
\]

(b) start of region end of region
Schedulability Conditions – MTS

- Message invocations $j$ will have priority over $i$ if:
  1. $(d_i - C_i) > (d_j - C_j)$, or
  2. (a) $(d_i - C_i) < (d_j - C_j) \leq (d_i - C_i + l_r)$, and
     (b) DM priority of $j$ is greater than that of $i$, and
     (c) $j$ is released before $d_i - C_i$.

- So a high-speed message is schedulable if its first invocation $i$ satisfies the condition:
  $$\exists t \in S, \sum (\text{lengths of all “qualified” } j \text{ released before } t) + C_p \leq t,$$
  where $S = \{\text{set of release times of each } j\} \cup \{d_i - C_i\}$,
  $C_p$ is the size of a longest possible packet.

- Low-speed messages – just check DM schedulability for each low-speed message.
  o Since high-speed messages have shorter deadlines than low-speed ones, they will automatically have higher DM priority (which is exactly what we want).
Simulation

- Let ED* be an ideal (imaginary) scheduling policy:
  - Works same as ED (no quantization), but . . .
  - Requires only an 11-bit ID.

- We expect MTS’s performance to be
  - Better than that of DM.
  - Close to that of ED*.
Simulation Workload Model

Consider an industrial drill with attached robot arm.

**High-speed periodics:** Needed for servo control of high-speed drives.

- Two messages per drive (feedback and command).
  \[ r_1 = t_0, \quad r_2 = t_0 + 0.5c, \quad d_1 = d_2 = 0.4c. \]

**High-speed sporadics:** Needed for contact sensors in robotic grippers.

- Assume that deadline of contact sensor message is one-fourth of drive cycle time.

<table>
<thead>
<tr>
<th>High-speed messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Fingers</td>
</tr>
<tr>
<td>Joints</td>
</tr>
<tr>
<td>Carriage</td>
</tr>
<tr>
<td>Drill</td>
</tr>
<tr>
<td>Sensors</td>
</tr>
</tbody>
</table>

**Low-speed messages:** Examples: servo control of low-speed drives (periodic) and smart temperature monitoring sensor (sporadic).

<table>
<thead>
<tr>
<th>Low-speed messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
<tr>
<td>Periodic</td>
</tr>
<tr>
<td>Sporadic</td>
</tr>
</tbody>
</table>
Simulation Workload Model (cont’d)

Message sizes: Standard CAN format has 47 framing bits.

• Servo control typically needs 32-bit command and feedback values. So periodic messages are 79 bits long. (Use for $C_p$ as well).

• Sporadic messages only notify of an event; so the ID is enough – 0 data bytes needed.

Length of epoch: $M = 10$ MIPS, $n = 1000$ instructions, $x = 5\%$.

\[ \ell = \frac{1000}{(0.05)(20 \times 10^6)} = 1\text{ms} \]

Length of region:

\[ l_r = \frac{\ell}{2^m - 1} = \frac{1\text{ms}}{2^5 - 1} = 32.3\mu\text{s} \]

• Use this workload to evaluate MTS on 10 Mb/s CAN.
Simulation Results

Vary number of high-speed periodics (6kHz messages):
DM can handle only 5 ($U = 58.5\%$); MTS and ED* can handle 8 each ($U = 72.7\%$).

Vary deadlines of high-speed periodics (with six 6kHz messages and DM fails at $99.9\mu s$ or less; MTS and ED* can handle even $56.8\mu s$ (min. possible).
Simulation Results (cont’d)

Vary number of high-speed sporadics with $U = 63.2\%$: DM can handle only 1 sporadic message; MTS and ED* can handle 4 each.

Vary deadlines of high-speed sporadics with number fixed at 2: DM fails at $104.1\mu s$ or less; MTS and ED* can handle even $17.3\mu s$ (min. possible).
Simulation Results (cont’d)

Vary deadlines of high-speed sporadics (cont’d): To better compare MTS and ED*, increase load. Use 10 6kHz messages ($U = 82.2\%$).

MTS fails at 151.5$\mu$s or less; ED* fails at 72.5$\mu$s or less;

Low-speed messages: Even 10% of the bandwidth is enough to accommodate about a hundred low-speed messages. Thus, the schedulability of low-speed messages is not a problem. Our simulations showed no real difference between DM, MTS, or ED* in scheduling low-speed messages for a fixed load of high-speed messages.