Modeling and Verification of Transmission Protocols:
A Case Study on CSMACD Protocol

SHI Ling and LIU Yan
Outline

• Motivation
• Background
• Model for CSMA/CD Protocol
• Verification Properties and Experimental Results
• Conclusion & Future Works
Motivation

- Real-time systems are mission critical;
- Potential causes to real-time systems:
  - Environmental conditions, human errors
  - Design errors

Verification Methods:
- Human inspection, Simulation, Testing
- Model Checking and PAT

~ Potential guarantee correctness
Outline

• Motivation
• Background
  - Timed extension for CSP#
  - Timed refinement checking
  - The CSMA/CD protocol
• Model for CSMA/CD Protocol
• Verification & Results
• Conclusion & Future Works
Background(1) – Timed CSP#

P = Stop | Skip
| e -> P
| P [] Q | P<>Q
| P; Q
| P ||Q
| Wait[d]
| P timeout[d] Q
| P interrupt[d] Q
| P within[d]
| P waituntil[d]
| P deadline[d]

– primitives
– event prefixing
– general choice
– sequential composition
– parallel composition
– delay
– timeout
– timed interrupt
– react within some time
– wait until
– deadline

SSIRI 2010
Background(2) – Timed Refinement

- Timed safety property can be proved by

  \[
  \text{#assert implementation refines}<T> \text{ specification;}
  \]

- For example: a model \( I \) contains two events start and end, a specification \( S = \text{start} \rightarrow ((\text{end} \rightarrow S) \quad \text{within}[5]) \)

  \[
  \text{#assert } I \text{ refines}<T> S ;
  \]
Background(3) –
The CSMA/CD Protocol

Abstract algorithm of CSMA/CD Protocol:

Agent Ready to Send

Sense Bus

Start Transmitting

Collision Detected

Abort Transmission

Wait and Retry

Busy

Idle

No Collision Informed

Transmission Completed
Outline

• Motivation
• Background
• Model for CSMA/CD Protocol
• Verification Properties and Experimental Results
• Conclusion & Future Works
Model for CSMA/CD Protocol

- Assumptions
  - Agents communicate in the 10Mbps Ethernet with a worst case for absence signal travel of 26 $\mu$sec
  - Messages have a fixed length of 1024 bytes
  - Time for transmitting a complete message is assumed to be a constant time 808 $\mu$sec, including propagation time
  - Backoff strategy for agent retrying is not modeled
# Model for CSMA/CD Protocol

## Components

<table>
<thead>
<tr>
<th>Components</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Definition</strong></td>
<td><strong>N</strong></td>
<td>Constant: number of senders</td>
</tr>
<tr>
<td></td>
<td><strong>channel newMess 0</strong></td>
<td>Sender gets messages to send</td>
</tr>
<tr>
<td></td>
<td><strong>channel begin 0</strong></td>
<td>Sender starts sending message</td>
</tr>
<tr>
<td></td>
<td><strong>channel busy 0</strong></td>
<td>Sender senses a busy bus</td>
</tr>
<tr>
<td></td>
<td><strong>channel cd 0</strong></td>
<td>Sender detects a collision</td>
</tr>
<tr>
<td></td>
<td><strong>channel end 0</strong></td>
<td>Sender completes its transmission</td>
</tr>
<tr>
<td><strong>Sender Behavior</strong></td>
<td><strong>WaitFor(i)</strong></td>
<td>Sender i is waiting for a message from the upper level</td>
</tr>
<tr>
<td></td>
<td><strong>Trans(i)</strong></td>
<td>Sender i is sending a message</td>
</tr>
<tr>
<td></td>
<td><strong>Retry(i)</strong></td>
<td>Sender i is waiting to retry after detecting a collision or a busy bus</td>
</tr>
<tr>
<td><strong>Bus Behavior</strong></td>
<td><strong>Idle</strong></td>
<td>Bus is free, no sender is transmitting</td>
</tr>
<tr>
<td></td>
<td><strong>Active</strong></td>
<td>One sender starts transmitting and is detecting collision</td>
</tr>
<tr>
<td></td>
<td><strong>Active1</strong></td>
<td>One sender is transmitting messages, bus is busy</td>
</tr>
<tr>
<td></td>
<td><strong>Collision</strong></td>
<td>Collision occurs and bus broadcasts the collision information to all senders</td>
</tr>
</tbody>
</table>
Model for CSMA/CD Protocol

• Sender Behavior

\[ \text{WaitFor}(i) = (\text{cd}?i \rightarrow \text{WaitFor}(i)) \]
\[ \quad \text{[] (newMess!i} \rightarrow ((\text{begin!i} \rightarrow \text{Trans}(i)) \]
\[ \quad \quad \quad \text{[] (busy?i} \rightarrow \text{Retry}(i)) \]
\[ \quad \quad \quad \text{[] (cd?i} \rightarrow \text{Retry}(i))))); \]

\[ \text{Trans}(i) = (\text{cd}?i \rightarrow \text{Retry}(i) \text{ within}[0,52]) \]
\[ \quad \text{[] (atomic\{end!i} \rightarrow \text{Skip}\} \text{ within}[808,808];} \]
\[ \quad \text{WaitFor}(i)); \]

\[ \text{Retry}(i) = \text{newMess!i} \rightarrow ((\text{begin!i} \rightarrow \text{Trans}(i) \text{ within}[0, 52]) \]
\[ \quad \text{[] (busy?i} \rightarrow \text{Retry}(i) \text{ within } [0, 52]) \]
\[ \quad \text{[] (cd?i} \rightarrow \text{Retry}(i) \text{ within}[0, 52])); \]
Model for CSMA/CD Protocol (Cont.)

• Bus Behavior

Idle = newMess?i -> begin?i -> Active;

Active = (end?i -> Idle)
[] (newMess?i ->
((begin?i -> Collision) timeout[26] (busy!i -> Active1)));

Active1 = (end?i -> Idle)
[] (newMess?i -> busy!i -> Active1);

Collision = atomic{BroadcastCD(0)} within[0, 26]; Idle;
Model for CSMA/CD Protocol (Cont.)

• **BroadcastCD process**

\[
\text{BroadcastCD}(x) = \text{if } (x < N) \{ \\
\text{cd!}x \rightarrow \text{BroadcastCD}(x+1) \\
[] \\
(\text{newMess?[i==x]i} \rightarrow \text{cd!}x \rightarrow \text{BroadcastCD}(x+1)) \\
\} \\
\text{else } \{ \\
\text{Skip} \\
\};
\]

• **CSMACD Process**

\[
\text{CSMACD} = (\| | | x :\{0..N-1\}@\text{WaitFor}(x)) | | | \text{Idle};
\]
Outline

• Motivation
• Background
• Model for CSMA/CD Protocol
• Verification Properties and Experimental Results
• Conclusion & Future Works
Verification Properties

• Deadlock Freeness (P0)
• Timed Divergence-free (P1)
• Collision detection in a given bounded delay (P2)
  ✓ Use refinement model checking techniques
  ✓ Build a model Spec which satisfies the property, then check whether CSMACD model satisfies Spec or not
Verification Properties (Cont.)

• Spec Model

\[ \text{Spec} = (\text{newMess.0} \rightarrow \text{begin.0} \rightarrow \text{Constrained1}) \]
\[ \quad \left[ \right] (\text{newMess.1} \rightarrow \text{begin.1} \rightarrow \text{Constrained2}) \]
\[ \quad \left[ \right] \text{Relaxed}; \]

\[ \text{Constrained1} = ((\text{newMess.1} \rightarrow \text{begin.1} \rightarrow) \]
\[ \quad \left( (\text{cd.0} \rightarrow \text{Skip} \left[ \right] \text{cd.1} \rightarrow \text{Skip}) \right) \text{deadline}[52])); \text{Spec} \]
\[ \quad \left[ \right] \text{Relaxed}; \]

\[ \text{Constrained2} = ((\text{newMess.0} \rightarrow \text{begin.0} \rightarrow) \]
\[ \quad \left( (\text{cd.0} \rightarrow \text{Skip} \left[ \right] \text{cd.1} \rightarrow \text{Skip}) \right) \text{deadline}[52])); \text{Spec} \]
\[ \quad \left[ \right] \text{Relaxed}; \]

\[ \text{Relaxed} = (\left[ \right] x: \{2..N-1\} @ (\text{newMess.x} \rightarrow \text{begin.x} \rightarrow \text{Spec})) \]
\[ \quad \left[ \right] \left[ \right] x: \{0..N-1\} @ ((\text{newMess.x} \rightarrow (\text{busy.x} \rightarrow \text{Spec} \left[ \right] \text{cd.x} \rightarrow \text{Spec})) \]
\[ \quad \left[ \right] \left( \right) (\text{cd.x} \rightarrow \text{Spec}) \]
\[ \quad \left[ \right] (\text{end.x} \rightarrow \text{Spec})); \]
Experimental Results

Testbed is a computer with 2.33GHz Intel(R) core(TM)2 Duo CPU and 3.25GB memory.

<table>
<thead>
<tr>
<th>Property</th>
<th>No. of Senders</th>
<th>Result</th>
<th>#States</th>
<th>#Transitions</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>4</td>
<td>Yes</td>
<td>787</td>
<td>1075</td>
<td>0.20</td>
</tr>
<tr>
<td>P0</td>
<td>5</td>
<td>Yes</td>
<td>2789</td>
<td>3847</td>
<td>0.60</td>
</tr>
<tr>
<td>P0</td>
<td>6</td>
<td>Yes</td>
<td>8851</td>
<td>12227</td>
<td>2.28</td>
</tr>
<tr>
<td>P0</td>
<td>7</td>
<td>Yes</td>
<td>26109</td>
<td>35991</td>
<td>8.43</td>
</tr>
<tr>
<td>P0</td>
<td>8</td>
<td>Yes</td>
<td>73123</td>
<td>100419</td>
<td>31.03</td>
</tr>
<tr>
<td>P0</td>
<td>9</td>
<td>Yes</td>
<td>196997</td>
<td>269319</td>
<td>108.69</td>
</tr>
<tr>
<td>P0</td>
<td>10</td>
<td>Yes</td>
<td>514915</td>
<td>700611</td>
<td>361.58</td>
</tr>
<tr>
<td>P1</td>
<td>4</td>
<td>Yes</td>
<td>787</td>
<td>1075</td>
<td>0.17</td>
</tr>
<tr>
<td>P1</td>
<td>5</td>
<td>Yes</td>
<td>2789</td>
<td>3847</td>
<td>0.66</td>
</tr>
<tr>
<td>P1</td>
<td>6</td>
<td>Yes</td>
<td>8851</td>
<td>12227</td>
<td>2.53</td>
</tr>
<tr>
<td>P1</td>
<td>7</td>
<td>Yes</td>
<td>26109</td>
<td>35991</td>
<td>9.79</td>
</tr>
<tr>
<td>P1</td>
<td>8</td>
<td>Yes</td>
<td>73123</td>
<td>100419</td>
<td>35.69</td>
</tr>
<tr>
<td>P1</td>
<td>9</td>
<td>Yes</td>
<td>196997</td>
<td>269319</td>
<td>123.24</td>
</tr>
<tr>
<td>P1</td>
<td>10</td>
<td>Yes</td>
<td>514915</td>
<td>700611</td>
<td>407.12</td>
</tr>
<tr>
<td>P2</td>
<td>4</td>
<td>Yes</td>
<td>787</td>
<td>1075</td>
<td>0.20</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>Yes</td>
<td>2789</td>
<td>3847</td>
<td>0.90</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>Yes</td>
<td>8851</td>
<td>12227</td>
<td>3.69</td>
</tr>
<tr>
<td>P2</td>
<td>7</td>
<td>Yes</td>
<td>26109</td>
<td>35991</td>
<td>14.74</td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
<td>Yes</td>
<td>73123</td>
<td>100419</td>
<td>55.38</td>
</tr>
<tr>
<td>P2</td>
<td>9</td>
<td>Yes</td>
<td>196997</td>
<td>269319</td>
<td>196.35</td>
</tr>
<tr>
<td>P2</td>
<td>10</td>
<td>Yes</td>
<td>514915</td>
<td>700611</td>
<td>655.3</td>
</tr>
</tbody>
</table>
Outline

• Motivation
• Background
• Model for CSMA/CD Protocol
• Verification Properties and Experimental Results
• Conclusion & Future Works
Conclusion

• Specify a formal model for CSMA/CD protocol
• Verify the properties using PAT
On-going and Future Works

• Model back off strategy for agent retrying of CSMA/CD protocol
• Apply probabilistic model checking techniques to model more richer properties of the protocol
• Improve PAT to efficiently deal with state explosion problems
Thanks & QA!

National University of Singapore
School of Computing