Formal Specification and Analysis of AFDX Redundancy Management Algorithms

Jan Täubrich

9. Bieleschweig-Workshop zum Systems Engineering
15-05-2007

Outline

Outline

Motivation

- Modern Airliners use the *all-electronic-fly-by-wire* technology
- Increased demand for bandwidth and reliability required new avionic bus
- For economic reasons *off-the-shelf-technologies* shall be explored
- Research resulted in AFDX founding on *IEEE 803.2 Ethernet*

Facts about AFDX

- Profiled network with star topology of maximum 24 end systems
- Full duplex to overcome unpredictable delay of ethernet
- Deterministic *point-to-point* communication through *Virtual Links*
- Allocated bandtwith for each *Virtual Link*
- AFDX can be run with 10 Mbps or 100 Mbps
- **Redundant network scheme increases reliability and availability**

Redundant network scheme

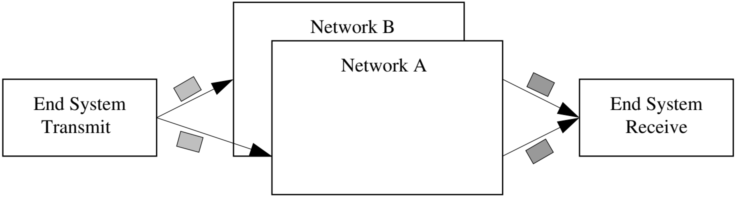

Figure: Concept of redundant networks

Redundancy Management Task

Defining the task for the redundancy management.

1. The redundancy management shall not submit redundant frames to the application layer.
2. Furthermore, the redundancy management shall preserve the order of the delivered frames. Hence, if the network is perfectly preserving the order the redundancy management shall do this as well.

Location of Redundancy Management

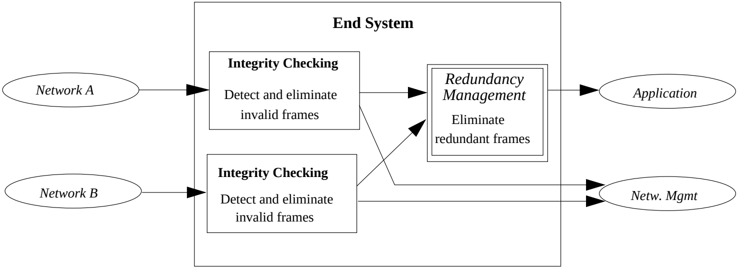

Figure: Placement of redundancy management.

Sequence Numbers

Each frame contains a sequence number. For these sequence numbers we define:

- $SN\_CNT =_{def} 2^8$, to be the number of sequence numbers
- $SN\_MAX =_{def} SN\_CNT - 1$, to be the maximum sequence number
- $SN\_HALF =_{def} SN\_CNT/2$, to be the mid sequence number

Consecutive frames have sequence numbers as follows:
$SN(f_{i+1}) =_{def} SN(f_i) + 1 \text{ mod } SN\_CNT$

Operations on Sequence Numbers

To sort out redundant and outdated frames, one needs to determine the order of sending. Thus subtraction of sequence numbers is defined as:

$$s_1 -_{SN} s_2 =_{def} ((s_1 - s_2 + SN\_HALF) \bmod SN\_CNT) - SN\_HALF$$

The comparison operators are defined in the obvious way, using the above defined subtraction.

Building the Algorithms

With the definition of the *Sequence Number Skew* (SNS) and the *Sequence Number Offset* SNO alone 6 algorithms were proposed. The *Sequence Number Skew* of frame f is

$$SNS(f) =_{def} RSN(f) -_{SN} RSN(PTN(f)).$$

Respectively the *Sequence Number Offset* is

$$SNO(f) =_{def} RSN(f) -_{SN} PASN(f)$$

Algorithm Examples

Although quite similar the following algorithms differ in their behaviour for a remarkable number of scenarios.

1. Accept frame f if $SNS(f) > 0$
2. Accept frame f if $SNO(f) > 0$
3. Accept frame f if $max(SNS(f), SNO(f)) > 0$
4. Reject frame f if $SNS\_MIN \leq SNS(f) \leq 0$
5. Reject frame f if $SNS\_MIN \leq SNO(f) \leq 0$
6. Reject frame f if last two points are satisfied together

Outline

Non-Functional Requirements

Each proposed algorithm shall satisfy the following properties:

1. The algorithm shall be easy to understand.
2. It shall allow certification.
3. It shall allow verification with acceptable effort.
4. And finally shall enable cost effective implementation.

Functional Requirements

There were 18 functional properties devided into 4 parts.

- Saftey - requirements on handling of redundant and outdated frames
- Liveness - requirements on the advance of the system
- Quality - requirements on the systems's availabilty in case of two operating networks
- Availability - requirements on the system's availability in case of one faulty network

Requirement Example

The following example shows the liveness formula

$$
\begin{aligned}
\mathit{Liveness} \triangleq\ & \forall \langle id, pos \rangle \in \mathit{deliverable} : \\
& \quad \lor \text{ENABLED } \langle \mathit{extAcceptFrame}(id, \mathit{env.frames}[id][pos][SN], pos) \rangle_v \\
& \quad \lor \text{ENABLED } \langle \mathit{extRejectFrame}(id, \mathit{env.frames}[id][pos][SN], pos) \rangle_v
\end{aligned}
$$

Figure: Specification of liveness in TLA^+.

Specifying the Algorithms

Each algorithm may perform one of the following tasks:

- Accept the incoming frame
- Reject the incoming frame
- Optionally wait until a timeout occurs

Example

Accept frame IF frames are available AND $(SNS(f) > 0$ OR $SNO(f) > 0)$

$$acceptFrame(id, sn) \triangleq$$
$$\land \lor snSkew[id, sn] > 0 \lor snOffset[sn] > 0$$
$$\land rm' = [rm \text{ EXCEPT } !.rsn = sn, !.paf = sn, !.ptn[id] = sn]$$

Reject frame IF frames are available AND $SNS(f) < = 0$ AND $SNO(f) < = 0$

$$rejectFrame(id, sn) \triangleq$$
$$\land snSkew[id, sn] \leq 0 \land snOffset[sn] \leq 0$$
$$\land rm' = [rm \text{ EXCEPT } !.rsn = sn, !.ptn[id] = sn]$$

Figure: Decision specification in TLA$^+$

Next Step Definition

Step of Redundancy Management

$$RM\_Next \triangleq \exists \langle id, sn \rangle \in networks \times (0 \, .. \, SN\_MAX) : acceptFrame(id, sn) \lor rejectFrame(id, sn)$$

The RM shall react on each frame

$$RM\_Fairness \triangleq \land \mathrm{WF}_{\langle rm \rangle}(RM\_Next)$$

$$RM\_Spec \triangleq InitRM \land \Box[RM\_Next]_{\langle rm \rangle} \land RM\_Fairness$$

Figure: Composed specification formula

Outline

Write Things Once

TLA$^+$ provides module instantiation to allow single specification of the environment.

MODULE *ENV*

```
CONSTANTS
networks,                       set of networks
SN_CNT, SN_MAX, SN_HALF,        maximum sequence number
MCFL,                           maximum number of consecutive frame loss
MTF,                            maximumb number of transient frames
A, B, SN, TAG                   just for convenience

VARIABLES
rm,                             Redundancy Management
env,                            Environment including the redundant networks
out,                            forwarded "frames"
status                          debugging

INSTANCE RMA11 WITH SNS_MIN <- MTF    load instance of RMA
```

Environment Send and Receive

accept frame:

$$
\begin{array}{l}
extAcceptFrame(id, sn, pos) \triangleq \\
\quad \land acceptFrame(id, sn) \\
\quad \land env' = [env \text{ EXCEPT} \\
\qquad !.frames[id] = SubSeq(@, pos + 1, Len(@)), \\
\qquad !.frames[TNid[id]] = \\
\qquad\ \text{IF } Len(@) \geq Len(SubSeq(env.frames[id], pos, Len(env.frames[id]))) \\
\qquad\quad \text{THEN } tag[env.frames[TNid[id]], \\
\qquad\qquad\qquad SubSeq(env.frames[id], pos, Len(env.frames[id])), \\
\qquad\qquad\qquad env.frames[id][pos][SN], id] \\
\qquad\ \ \text{ELSE } @] \\
\quad \land out' = out \cup \{env.frames[id][pos]\} \\
\quad \land status' = \text{“accept”}
\end{array}
$$

reject frame:

$$
\begin{array}{l}
extRejectFrame(id, sn, pos) \triangleq \\
\quad \land rejectFrame(id, sn) \\
\quad \land env' = [env \text{ EXCEPT} \\
\qquad !.frames[id] = SubSeq(@, pos + 1, Len(@)), \\
\qquad !.frames[TNid[id]] = \\
\qquad\ \text{IF } Len(@) \geq Len(SubSeq(env.frames[id], pos, Len(env.frames[id]))) \\
\qquad\quad \text{THEN } tag[env.frames[TNid[id]], \\
\qquad\qquad\qquad SubSeq(env.frames[id], pos, Len(env.frames[id])), \\
\qquad\qquad\qquad env.frames[id][pos][SN], id] \\
\qquad\ \ \text{ELSE } @] \\
\quad \land \text{UNCHANGED } \langle out \rangle \\
\quad \land status' = \text{“reject”}
\end{array}
$$

Marking of Frames I

Frames must be marked as redundant or old.
Let N_1 be a non-empty network and f a frame in N_1, located at $n \in \{1, \dots, Len(N_1)\}$. It can be deduced that the twin frame of f is still pending on the second network N_2 if and only if

$$Len(SubSeq(N_1, n, Len(N_1))) \leq Len(N_2). \qquad (1)$$

More precisely, given networks N_1 and N_2 with $Len(N_1) \leq Len(N_2)$, we know that, selecting frame at position $0 < k \leq Len(N_1)$ from network N_1, its twin frame on network N_2 is located at position l for which holds:

$$Len(SubSeq(N_1, k, Len(N_1))) = Len(SubSeq(N_2, l, Len(N_2))) \quad (2)$$

Marking of Frames II

$$
\begin{array}{l}
tag[seq1 \in Seq([sn : (0 \,..\, SN\_MAX),\ tag : \{\text{“n”}, \text{“r”}, \text{“o”}\}]), \\
\quad seq2 \in Seq([sn : (0 \,..\, SN\_MAX),\ tag : \{\text{“n”}, \text{“r”}, \text{“o”}\}]), \\
\quad val \in (0 \,..\, SN\_CNT),\ id \in networks] \triangleq \\
\quad\quad \text{IF } Len(seq1) > Len(seq2) \text{ THEN} \\
\quad\quad\quad \text{IF } Head(seq1)[TAG] = \text{“r”} \text{ THEN } \langle Head(seq1) \rangle \circ tag[Tail(seq1),\ seq2,\ val,\ id] \\
\quad\quad\quad \text{ELSE } \langle [sn \mapsto Head(seq1)[SN],\ tag \mapsto \text{“o”}] \rangle \circ tag[Tail(seq1),\ seq2,\ val,\ id] \\
\quad\quad \text{ELSE } \langle [sn \mapsto Head(seq1)[SN],\ tag \mapsto \text{“r”}] \rangle \circ Tail(seq1)
\end{array}
$$

Figure: Marking algorithm in TLA$^+$

Outline

About the Algorithms

- The originally proposed requirements did not help very well to distinguish the algorithms
- The most trivial algorithm had the best performance concerning the safety properties
- Scenario driven algorithm evolution seem to be not effecient enough

About TLA$^+$

- Compact notations keep specifications small, but are not easy to understand for „newbies “
- Instantiation supports „Write things once“principle
- Untyped variables are very flexible and type invariance properties help to keep them in the desired value ranges
- Nice LaTeX export
- Working with strings is tiresome

About TLC

- Temporal formulas must be of the form $\Box\Diamond A$ or $\Diamond\Box A$
- The „Contraint Problem“increases state size
- Instantiation does not map constants to the instantiated functions
- Poor handling of views
- Experienced to be slow for more than 10^6 states

Thank's for your Attention!

Your Questions?