Combining static priority and weighted round-robin like packet scheduling in AFDX for incremental certification and mixed-criticality support

Marc Boyer\textsuperscript{1}, Nicolas Navet\textsuperscript{2}, Marc Fume\textsuperscript{3}, Jörn Migge\textsuperscript{4}, Lionel Havet\textsuperscript{4}

\textsuperscript{1} ONERA – The French aerospace
\textsuperscript{2} University of Luxembourg, Luxembourg
\textsuperscript{3} Thales Avionics, Toulouse, France
\textsuperscript{4} RealTime-at-Work, Villers-lès-Nancy, France

5th EUropean Conference for Aeronautics and Qpace Sciences (EUCASS 2013)
Outline

AFDX backbone

Network partitioning
Deficit Round Robin
RTaW-PEGASE AFDX network analyser
Applicability of DRR for AFDX network
Applicability of SP/DRR for AFDX network
Conclusion
Real-Time distributed systems:
- AFDX $\approx$ Ethernet technology for avionics
- $\approx$ hundred of computers
- 8 switches
- $\approx$ thousands of data flows
Requirement: Worst Case Traversal Time

Requirement: bound on network latency

- Input contract traffic: Virtual Link
  - Static routing
  - Maximal frame size $I^m$
  - Minimal interval between two frames $BAG$

- Knowledge on switches:
  - bandwidth
  - scheduling policy: FIFO, Static Priority

- Analyse method
  - Network calculus
  - Trajectorial approach
  - Ad-hoc methods
Outline

AFDX backbone

Network partitioning

Deficit Round Robin

RTaW-PEGASE AFDX network analyser

Applicability of DRR for AFDX network

Applicability of SP/DRR for AFDX network

Conclusion
Design challenge: system independence

Network: shared resource
- performance of one flow depends on all other flows
- hundred of systems (and system providers)
- need of early network design
- avoid “frequent” re-design / provisioning

Network independence:
- Virtual Link: a mid-grained independence
  - allows frame content change
  - allows frame size reduction
  - allows period change ($\geq BAG$)
  - $\approx 10^4$ VLs
- A more coarse-grained partitioning?
Network partitioning

Challenge: cut the net into virtual independent partitions
- 4-10 partitions (coarse-grained)
- independent performances

One physical 100Mb/s network

Two virtual 20Mb/s 80Mb/s networks
Generalised Processor Sharing

GPS: Generalised Processor Sharing

- cut the service into $n$ classes
- allocate a quantum $q_i$ to each class
- each class receives fraction $\frac{q_i}{\sum_{i=1}^{n} q_i}$ of service

- ideal policy
- Packetised-GPS (P-GPS):
  - practical implementation of GPS
  - implement GPS “up to one packet size”
  - can hardly be implemented in real-time

- Deficit Round Robin (DRR)
  - other GPS implementation
  - efficient implementation
Outline

AFDX backbone

Network partitioning

Deficit Round Robin

RTaW-PEGASE AFDX network analyser

Applicability of DRR for AFDX network

Applicability of SP/DRR for AFDX network

Conclusion
Deficit Round Robin:

- GPS implementation
- $O(1)$ implementation
- allocate one quantum $Q_i$ per class
- one queue per class
- infinite loop: for each class/queue
  - increment credit by $Q_i$ if active
  - sending packet of size $s$ decrease credit by $s$
  - send packets as long as non null credit
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

<table>
<thead>
<tr>
<th></th>
<th>DC</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```
   3  4
  ___
   2 1 5
  ___
   2
```

DC Q
5  5
0  7
0  3
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```
3
1 5
2 1 5 0 7
2 0 3
```
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```
+-----------------+   +----+
| 3              |   | 1  |
+-----------------+   +----+
| 2   1  5        |   | 0  |
+-----------------+   +----+
| 2              |   | 0  |
```

DC  Q
---+---+---
1  5
0  7
0  3
3 classes
Quanta Q: 5, 7, 3
CD: Deficit Counter
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

<table>
<thead>
<tr>
<th>DC</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```
3

2

2

DC  Q
1  5
1  7
0  3
```
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```
  3  1  5
  2  1  7
  2  0  3
```
3 classes
Quanta Q: 5, 7, 3
CD: Deficit Counter
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```

<table>
<thead>
<tr>
<th>DC</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Boyer, Navet, Fumey, Migge, Havet
DRR run example

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

```
 3
 Q: 1
 5
 2
 1
 7
 1
 3
```
**DRR run example**

- 3 classes
- Quanta Q: 5, 7, 3
- CD: Deficit Counter

<table>
<thead>
<tr>
<th>DC</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
DRR property

- DRR allows per class WCTT bound
- DRR introduces some latency
- previous contribution: latency computation
- this latency depends on quanta $Q_i$ and maximal frame sizes $l_i^m$

$$ DRRT \ latency = \frac{Q_i(L - l_i^m) + (F - Q_i)(Q_i + l_i^m)}{F} $$

with $F = \sum_{i=1}^{n} Q_i$, $L = \sum_{i=1}^{n} l_i^m$. 
Outline

AFDX backbone

Network partitioning

Deficit Round Robin

RTaW-PEGASE AFDX network analyser

Applicability of DRR for AFDX network

Applicability of SP/DRR for AFDX network

Conclusion
RTaW-PEGASE AFDX network analyser

PEGASE:
- french-founded project
- 2009-2012
- partners: 4 academics (ONERA, ENS, ENS Lyon, Inria), 1 SME (RealTime-at-Work), 1 major company (Thales)
- Network Calculus for avionics networks

RTaW-PEGASE:
- prototype from PEGASE become product
- developed by RealTime-at-Work
- augmented with DRR analyse
Outline

AFDX backbone

Network partitioning

Deficit Round Robin

RTaW-PEGASE AFDX network analyser

Applicability of DRR for AFDX network

Applicability of SP/DRR for AFDX network

Conclusion
DRR pro and cons:

+ network partitioning
- additional latency
DRR for AFDX

- DRR pro and cons:
  - network partitioning
    - additional latency
  - Bandwidth should increase
DRR for AFDX

- DRR pro and cons:
  + network partitioning
  - additional latency
- Bandwidth should increase
  ⇒ use bandwidth increase to “hide” latency increase
DRR pro and cons:
  + network partitioning
    - additional latency
  Bandwidth should increase
  ⇒ use bandwidth increase to “hide” latency increase
  does it works?
DRR for AFDX experiment

- AFDX realistic configuration
- assume $n$ classes
- random allocation of VL to classes
- quanta: proportional to class load
- what bandwidth to get same performance as 100Mb/s FIFO network?
- 100 configurations for each $n$
The quantum allocated to a class is directly proportional to its bandwidth requirement, with the property that the class with the least bandwidth requirement is still allocated a quantum sufficiently large to transmit any frame. Let $M$ denote the maximum packet size over the system, the quantum of DRR class $k$, denoted by $Q_k$, is given as follows:

$$\text{Load}(M) = \text{Min} \left( \sum_{i \in \{ \text{Bag} \}} \text{CVLi}_i \cdot k_i \right)$$

Additional bandwidth requirements under DRR.

The aim of this first set of experiments is to explore the minimum network speed increase that is required to keep all the VLs under their latency constraint when several DRR classes are introduced. Using the reference configuration described in §4.1 that is feasible (i.e., all the VLs are under their latency constraints) with FIFO scheduling and all links at 100Mbit/s, we generate 100 random DRR configurations according to the process described in the previous paragraph. Then, for each configuration, we search for the minimum network data rate that leads to a feasible system. During the process, like in the initial configuration, it is assumed that the speed is the same over all links, be they links from end-systems to routers or links between routers. The computation performed with RTaW4Pegase takes a few seconds for a given data rate, and the search is speed up by a dichotomic search of various granularity (first 100Mbit/s then 10 Mbit/s).

The results shown in Figure 4 show that the average additional bandwidth required by DRR increases almost linearly with the number of DRR classes. In our experiments, the average bandwidth needed under DRR (up to 8 classes) is approximately the number of DRR classes times the minimum bandwidth needed under FIFO. The curve of the minimum value over the set of experiments is significantly below the average value and less smooth. The latter phenomenon can probably be explained by the variability among the set of candidate DRR configurations that are partially generated at random. The maximum data rates needed among the set of experiments (not shown in Figure 4) exhibit the same behaviour and, ranging from 270Mbit/s to 4450MBit/s, they are much larger than the average values (from a factor 1.3 up to a factor 10).
AFDX/DRR conclusion

- DRR latency costs
AFDX/DRR conclusion

- DRR latency costs
- but bandwidth can compensate it
AFDX/DRR conclusion

- DRR latency costs
- but bandwidth can compensate it
- 1Gb/s network allows 6-8 independent classes/partitions
Outline

AFDX backbone

Network partitioning

Deficit Round Robin

RTaW-PEGASE AFDX network analyser

Applicability of DRR for AFDX network

Applicability of SP/DRR for AFDX network

Conclusion
- a few flows with very high constraints
- keep these flows in high priority queue
same methodology as for DRR
all VLs with latency constraints $\leq 2$ms in high priority
We now consider the use of 1Gbit/s network, and if the minimum data rate needed for feasibility exceeds 1Gbit/s, the DRR configuration is deemed unfeasible. The results in Table 3 show that a 1Gbit/s network can ensure feasibility for almost all randomly generated configurations up to 6 DRR classes. The experiments shown in Figure 4 and Table 3 suggest to us that DRR can be a practical solution on 1Gbit/s networks with a traffic that is similar to the one from our reference configuration for a limited number of DRR classes.

Table 3: Percentage of DRR configurations feasible at 1Gbit/s for a number classes ranging from 2 to 10.

<table>
<thead>
<tr>
<th>Number of DRR Classes</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>100</td>
<td>99</td>
<td>99</td>
<td>97</td>
<td>96</td>
<td>92</td>
<td>78</td>
<td>71</td>
<td>50</td>
</tr>
</tbody>
</table>

4.4 Performance of SP/DRR scheduling

In the first series of experiment, all VLs were assigned to DRR classes. We now explore the combined use of Static Priority and DRR (SP/DRR) with two priority levels as discussed in §2.2. For a given VL, the choice among the two policies is based on the latency constraint: if a VL has a latency constraint that is below (or equal to) a certain threshold $l_c$ for at least one receiving end-system, it will be given the higher priority and will not be assigned to a DRR class. The VLs with their latencies above $l_c$ are scheduled under DRR at the lower priority level. In the latter case, the DRR class to which the VL belongs is assigned at random. The experiments conducted with RTaW4Pegase share the same setup as the ones from §4.3 except for the higher priority traffic scheduled under Static Priority, with the threshold $l_c$ set to 2ms.

The results of this experiment are presented in figure 5, along with the results obtained with DRR alone for the purpose of comparison. With SP/DRR, the same trend as with DRR alone can be observed: as the number of DRR classes grows, the bandwidth needed to obtain feasibility becomes larger. However, using priorities helps to offset the detrimental impact of a larger number of DRR classes: with 2 classes SP/DRR needs on average 14% less bandwidth than DRR while with 10 classes the gain of using SP/DRR reaches 39%.
AFDX/DRR conclusion

- SP/DRR improves number of possible partition
SP/DRR improves number of possible partition
no independence in high priority class
AFDX/DRR conclusion

- SP/DRR improves number of possible partition
- no independence in high priority class
- 1Gb/s network allows 10 independent classes/partitions
Outline

AFDX backbone

Network partitioning

Deficit Round Robin

RTaW-PEGASE AFDX network analyser

Applicability of DRR for AFDX network

Applicability of SP/DRR for AFDX network

Conclusion
Conclusion

- successful AFDX
Conclusion

- successful AFDX
- AFDX evolutions:
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
  - Implementation in RTaW-PEGASE
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
  - Implementation in RTaW-PEGASE
  - Is pure DRR acceptable?
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
  - Implementation in RTaW-PEGASE
  - Is pure DRR acceptable?
  - DRR-like solutions:

Boyer, Navet, Fumey, Migge, Havet
SP/DRR for AFDX

ONERA
THE FRENCH AEROSPACE LAB
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
  - Implementation in RTaW-PEGASE
  - Is pure DRR acceptable?
  - DRR-like solutions:
    - SP/DRR
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
  - Implementation in RTaW-PEGASE
  - Is pure DRR acceptable?
  - DRR-like solutions:
    - SP/DRR
    - Modified DRR (cf Cisco, Juniper)
Conclusion

- successful AFDX
- AFDX evolutions:
  - bandwidth increase
  - network coarse-grained partitioning
- First experiment with DRR
  - Implementation in RTaW-PEGASE
  - Is pure DRR acceptable?
- DRR-like solutions:
  - SP/DRR
  - Modified DRR (cf Cisco, Juniper)
  - Avionic-specific DRR?