EVALUATION ASSESSMENT AND PROTOTYPING OF SPACEWIRE PROTOCOLS (SPW-D, RMAP)

(CCN3 TO ESTEC CONTRACT 22256/09/NL/CBI)

ANALYSIS AND ASSESSMENT OF THE SPACEWIRE-D DRAFT B SPECIFICATION

Prepared for the:
European Space Agency

Prepared by:
TELETEL S.A.

REVISION HISTORY

Version	***Date***	***Responsible***	***Comment***
V0.1	31 August 2010	Nikos Pogkas	*INITIAL VERSION*
V0.2	10 September 2010	Nikos Pogkas	*ADDED SECTIONS 3, 4*
V0.3	20 September 2010	Nikos Pogkas	*ADDED SECTIONS 6*
V0.4	20 September 2010	Antonis Tavoularis	*ADDED SECTIONS 5.1 – 5.4, 6.3*
V0.5	27 September 2010	Nikos Pogkas	*PRE-FINAL VERSION*
V1.0	5 October 2010	Nikos Pogkas, Antonis Tavoularis, Vangelis Kollias	*FINAL VERSION*

TABLE OF CONTENTS

1 Scope

1.1 Scope of the document

The scope of this document is to provide an initial analysis of the current SpaceWire-D Draft B [AD,2] protocol specification in order to identify open issues and problem areas and to evaluate through simulations the protocol performance, as well as to present protocol features and possible considerations for further assessment and future implementations.

- Section 2 presents an overview of the SpaceWire-D as provided according to [AD, 2].
- Section 4 presents simulations results in order to evaluate real time performance and latency with respect to time-code periodicity, link speed, and RMAP packet length.
- Section 3 presents open issues and initially identified problem areas regarding performance of SpaceWire-D.
- Section 5 presents implementation issues and constraints for the implementation of SpaceWire-D.
- Section 6 presents considerations for further assessment and future implementations that focus on throughput and latency improvement as well as different alternatives for the implementation of Segmentation and Retry mechanisms.

1.2 Applicable and reference documents

1.2.1 Applicable documents (ADs)

The following documents, listed in order of precedence, contain requirements applicable to the activity.

[AD,1] CCN3 to ESTEC Contract No. 22256/09/NL/CBI2 “Protocol Validation System for Onboard Communications (PVS)” – Statement of Work – Evaluation, assessment and prototyping of SpaceWire protocols (SpW-D, RMAP)

[AD,2] “SpaceWire-D – Deterministic Control and Data Delivery over SpaceWire Networks”, Draft B, April 2010 (Ref: SpW-D Draft B)

[AD,3] “SpaceWire Remote Memory Access Protocol” ECSS-E-ST-50-52C

[AD,4] “SpaceWire: Links, nodes, routers and networks” ECSS-E-ST-50-12C

1.2.2 Reference documents (RDs)

The following documents, listed in order of precedence, are used as input for this report. They shall be considered as an integral part of the present deliverable. In the body of the applicable document, they are identified as [REF].

[SC2008] “RMAP over SpaceWire on the exomars rover for direct memory access by instrument to mass memory”, B. Dean et all, Spacewire-Conference 2008

[SOIS] “Spacecraft Onboard Interface Service” Draft Informational Report Jan 2008, CCSDS 850.0-G-R1.1

[SOIS-MA] “Spacecraft Onboard Interface Service Subnetwork Memory Access Service Draft Recommended Practice Jan 2008 CCSDS 852.0-R-1.1 Draft. CCSDS 852.0-R-1.1 Draft

1.3 Acronyms and abbreviations

AOCS	Attitude Orbit & Control Subsystem
GNC	Guidance and Navigation Control
OBSW	On-Board Software
PDU	Protocol Data Unit
QoS	Quality of Service
RMAP	Remote Memory Access Protocol
RMW	Read/Modify/Write
SDU	Service Data Unit
SOIS	Spacecraft Onboard Interface Services
SpW	SpaceWire
SpW-D	SpaceWire-D

2 SpaceWire-D Protocol Overview

This section presents an overview of SpaceWire-D protocol based on the SpaceWire-D Draft B specification [AD, 2].

2.1 Introduction

The aim of SpaceWire-D protocol as defined in [AD,2] is to pass information over a SpaceWire network deterministically i.e. where the time of delivery can be determined a priori within certain bounds.

As stated in [AD,2] the need for deterministic delivery of information arises from AOCS and GNC systems on board a spacecraft where the time that information is read from a sensor or sent to an actuator is important. In order for SpaceWire networks to be used for both payload and command and control applications a means of providing this determinism is essential. Combining payload and command and control onboard communications networks could significantly reduce system mass and complexity.

The solution presented in [AD,2] provides deterministic data delivery over SpaceWire networks which fully conform to the ECSS-E-ST-50-12C standard [AD,3]; no changes are required to SpaceWire interfaces or routers.

Spacewire-D uses the RMAP protocol, ECSS-E-ST-50-52C [AD,3], for transferring information over the SpaceWire network, no changes are required to RMAP protocol except some imposed timing constraints and a constraint in the maximum RMAP packet length that shall be used.

SpaceWire-D uses time division multiplexing for medium access control and defines a schedule which specifies when a particular node is allowed to initiate an RMAP transaction. The Scheduling layer is the main part of SpaceWire-D and is placed between RMAP and SpaceWire in the protocol stack, as presented in the following figure.

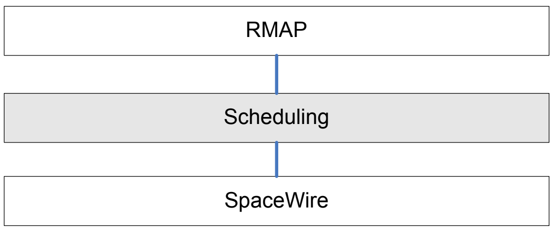


Figure 1: SpaceWire-D protocol stack

2.2 Time-slots and scheduling

Spacewire-D uses a time division multiplexing medium access scheme where the time-slots are delimited by SpaceWire time-codes.

- The receipt of a time-code at a node shall indicate the start of a time-slot.
- The time-slot number shall be the same as the time-value of the time-code that indicates the start of a time-slot.
- The end of a time-slot shall normally be indicated by the arrival of the next time-code.

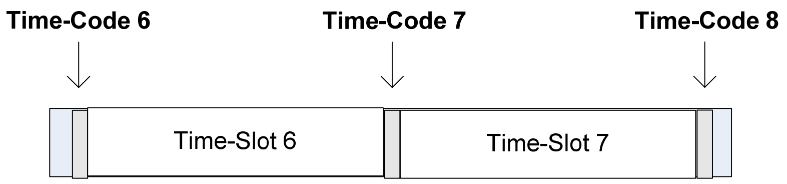


Figure 2: Time-slots

2.2.1 Time-codes processing

A time-code watchdog timer should be kept in each initiator to check for the correct arrival of each time-code.

- The time-code watchdog timer shall check for arrival of a time-code sooner than the minimum expected interval between time-codes (early time-code).
- The time-code watchdog timer shall check for a time-code that does not arrive before the maximum expected interval between time-codes (late time-code).
- In the event of an early or late time-code the initiator shall flag an error to the user application.

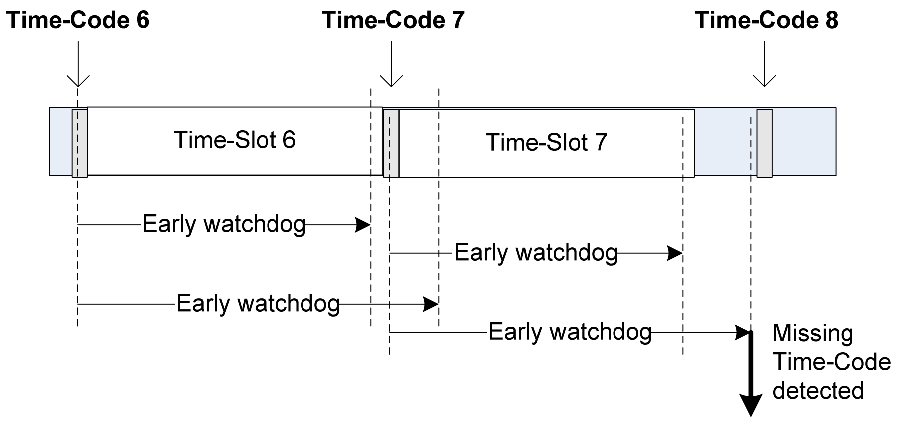


Figure 3: Time-code watchdog

2.2.2 Simple schedule

The simple schedule gives an initiator full control of the network for one or more specified time-slots.

- Any RMAP transaction shall start and finish in the same time-slot.
- Only one RMAP transaction shall take place in a particular time-slot.
- An initiator shall be permitted to initiate a transaction with any target device during a time-slot in which it is scheduled to initiate RMAP transactions.
- A simple schedule shall be defined using a schedule table for each initiator which specifies in which time-slot that initiator is allowed to initiate RMAP commands.

2.2.3 Concurrent schedule

The concurrent schedule makes more efficient use of network bandwidth by allowing more than one initiator to initiate RMAP transactions in a time-slot.

- More than one initiator may initiate RMAP transactions in the same time-slot provided that the paths from each of the initiators to their targets do not use any of the same SpaceWire links in the network.

2.2.4 Multi-slot schedule

The multi-slot schedule builds on the concurrent schedule to improve network efficiency further. Where a large amount of data has to be transferred between two nodes, the RMAP transaction to accomplish this is permitted to occupy more than one adjacent time-slot. This allows more data to be transferred in the one RMAP transaction.

- An RMAP transaction may initiate an RMAP transaction which has a duration of more than one time-slot i.e. the amount of data being written or read exceeds

- The schedule table shall ensure that when an RMAP transaction has a duration of more than one time-slot, it does not use the same network resources (SpaceWire links) as any other transactions occurring during any of those time-slots.

2.3 Initiator node processing

2.3.1 Packet transmission

The following functionality is required by initiators during packet transmission:

- Each node that is capable of being an initiator shall hold a copy of the schedule table.
- On receipt of a time-code an initiator shall check the schedule to determine if it is allowed to initiate an RMAP command during that time-slot.
- If the initiator is not allowed to initiate an RMAP command during the current time-slot, it shall not initiate any RMAP commands.
- If the initiator is allowed to send an RMAP command during the current time-slot, and if it has an RMAP command to send to any target device permitted by the schedule table, it shall send out that RMAP command.
- RMAP write command shall always request an acknowledgement.
- RMAP write commands can use verify before write for critical commands.
- After sending out an RMAP command the initiator shall listen for the reply to the RMAP command.

2.3.2 Packet reception

The following functionality is required by initiators during packet reception:

- On receipt of the reply to the RMAP command the information that it contains including the status information and any data returned in response to a read command, shall be passed to the user application that initiated the command.
- If simple or concurrent scheduling is being used and no reply is received by the time the next time-code is received, the initiator shall flag an error to the user application.
- If multi-slot scheduling is being used, an initiator may send an RMAP transaction that will take longer than one time-slot interval to complete, as specified by the schedule table.
- When mutli-slot scheduling is being used and an RMAP transaction longer that one time-slot is being initiated, the initiator shall check for completion of the RMAP transaction at the end of the last time-slot allocated for that transaction and flag an error to the user application if the RMAP transaction has not completed in time.

2.4 Target node packet processing

On receipt of an RMAP command the target device shall process it in accordance with the SpaceWire RMAP standard [AD,3].

- Target only nodes shall not need to hold a copy of the schedule table.
- Target only nodes may hold a copy of the schedule table for fault detection purposes.

2.5 Implementation constraints

2.5.1 Initiator constraints

The following constraints are defined in SpW-D Draft B for initiators:

- The maximum amount of data that can be read in an RMAP read command or written in an RMAP write command shall be 256 bytes (TBC) when simple or concurrent scheduling is being used.
- The maximum amount of data may be longer than 256 bytes when advanced scheduling is being used.
- The time taken from the receipt of a time-code to the starting to send out an RMAP command from an initiator shall be less than 5 µs (TBC).

2.5.2 Target constraints

The following constraints are defined in SpW-D Draft B for targets:

- The time taken from receipt of the complete RMAP command header in a target node to the authorisation or rejection of that RMAP command shall be less than 5 µs (TBC).
- The latency in transferring data from SpaceWire interface to memory shall be less than 5 µs (TBC).
- The time taken from completion of writing data to memory to starting to send the RMAP command shall be less than 5 µs (TBC).

3 Evaluation of SpW-D through simulation

In this section we analyse and evaluate the performance of SpW-D protocol in a simple network topology in order to estimate best case values for the data rate, time-slot duration and network efficiency.

3.1 Assumptions

For all the simulation studies the following assumptions are made:

- No additional delays are imposed from the SpW flow control, and the SpW throughput is equal to 0.8 * SpW link speed.
- The latency imposed from SpW routers is constant, as a result the total propagation delay of a packet in network is equal to R*D where D is the forwarding delay per router and R is the number of routers in the path from initiator to target.

3.2 Performance calculations

In this simulation studies the following formulas are used for computing the total time required for a write and a read RMAP transaction, which are the same as defined in SpW-D Draft B with an exception regarding the computation of the latency caused in the interface from SpaceWire to memory. In this case a constant arbitration delay (Tf1) and the latency caused by memory bandwidth of the interface (Tf2) are taken into account.

Ta: Interval from receipt of time-code to the RMAP command starting to be sent by the initiator. This interval includes: the time to receive, decode and respond to the time-code; the time to check the schedule table; the time to start to send out the RMAP command (assuming that the command has already been prepared ready for sending). This interval is entirely dependent upon the initiator implementation.

Tb: Interval for the SpaceWire packet containing the RMAP command to propagate across the SpaceWire network from initiator to target. This will mainly depend upon the number of routers between the initiator and the furthest target node. Assuming a time delay per router of 0.6 μs, the total propagation delay will be 0.6R where R is the number of routers in the longest path used between an initiator and a target.

Tc: Interval for sending the RMAP header, including any path address bytes. The size of the RMAP header including the SpW Target Address and Reply Address is H=R+16+P bytes, where R is the number of routers in the path from initiator to target and P is a the closest multiple of 4 which is greater than or equal to R. Thus if there are four router R=4 and P=4, so H=24 bytes. The time to send this header depends upon the SpaceWire data rate, S Mbits/s, and is 10H/S μs. For example with H=24 and S = 200 Mbits/s, Tc = 1.2 μs.

Td: Interval for authorising the RMAP command once the header has been received. This is dependent upon the target implementation.

Te: Interval to send the data and data CRC. This is given by 10(D+1)/S, where D is the number of data bytes. For D = 256 (e.g. the defined maximum amount of data permitted in a SpW-D RMAP write command or read reply), the time to send the data and data CRC is Te = 12.85 μs when S = 200 Mbits/s.

Tf1: Arbitration delay this interval covers an initial constant latency in the transfer of data from the SpaceWire interface to the target memory. It is dependent upon the implementation of the target node. A value that has been selected for simulation studies is equal to Tf1 = 3μs.

Tf2: This interval covers an additional latency, caused by the actual memory bandwidth, in the transfer of data from the SpaceWire interface to the target memory. This is given by (8*D)/MemBW, where D is the number of data bytes and MemBW is memory bandwidth which is dependent upon the implementation of the target node. A value that has been selected for simulation studies is equal to Tf2 = 1Gbps.

Note: The sum of Tf1 and Tf2 for a 256Bytes packet is equal to Tf1 + Tf2 = 5.048 μs which is near to the constant memory delay Tf defined in the SpW-D Draft B specification.

Tg: Interval from the completion of writing data to memory in the target to starting to send the RMAP reply. This interval is dependent upon the implementation of the target node.

Th: Interval for the SpaceWire packet containing the RMAP reply to propagate across the SpaceWire network from target to initiator. This will mainly depend upon the number of routers between the initiator and the furthest target node. Assuming a time delay per router of 0.6 μs, the total propagation delay will be 0.6R where R is the number of routers in the longest path used between an initiator and a target. This interval is the same as (b).

Ti: Interval for sending the RMAP reply, including any path address bytes. The size of the RMAP reply including the Reply Address is E=R+8 bytes, where R is the number of routers in the path from target to initiator. Thus if there are four router R=4, E=12 bytes. The time to send this header depends upon the SpaceWire data rate, S Mbits/s, and is 10E/S μs. For example with E=12 and S = 200 Mbits/s, T_i = 0.6 μs.

The total time for the complete transaction TT is as following:

TT = Ta + Tb + Tc + Td + Te + Tf1 + Tf2 + Tg + Th + Ti

The delays for the read command are very similar to those of the write command. The letters used to denote each of the time delays for the read command are the same as the comparable delays in the write command (hence they are not in alphabetical order).

Ta: Interval from receipt of time-code to RMAP read command starting to be sent by initiator.

Tb: Interval for the SpaceWire packet containing the RMAP read command to propagate across the SpaceWire network from initiator to target.

Tc: Interval for sending the RMAP read command, including any path address bytes.

Td: Interval for authorising the RMAP read command once it has been received.

Tg: Interval from authorisation of the read command to starting to send the RMAP reply.

Th: Interval for SpaceWire packet containing the RMAP reply to propagate across the SpaceWire network from target to initiator.

Ti: Interval for sending the RMAP reply header, including any path address bytes.

Te: Interval to send the data and data CRC.

Tf1: Arbitration delay in the transfer of data from the memory to the SpaceWire interface.

Tf2: Additional latency, caused by the actual memory bandwidth, in the transfer of data from memory to the SpaceWire interface.

The total time for the complete transaction is thus:

TT = Ta + Tb + Tc + Td + Tg + Th + Ti + Te + Tf1 + Tf2

The total time is the same as for the write command. Hence the performance is the same for either read or writes operations, as defined also in SpW-D Draft B.

3.3 SpW-D evaluation in simple network topology

In this section we evaluate protocol performance with respect to time code periodicity and data rate in a simple network topology.

The scope of this simulation study is to identify the maximum data rate, the maximum efficiency and the minimum time-slot interval for different packet lengths, SpW link speeds and memory interface bandwidths.

3.3.1 Simulation scenario

The following parameters are used in this simulation study:

Simulation Parameters	Value
SpW Link Speed	200Mbps or configurable
SpW Throughput	0.8 * SpW Link Speed
SpW router forwarding delay	0.6μs
Number of SpW routers	4
Time-slot interval	configurable
Maximum RMAP data length	256B or configurable
Guard Time-slot margin	2μs
Delay between reception of time-code, start transmission of command (Ta)	5μs
R-MAP authorisation delay (Td)	5μs
Delay between completion of memory transaction, start transmission of reply (Tg)	5μs
Arbitration delay at memory (Tf1)	3μs
Memory bandwidth (Tf2)	1Gbps or configurable

Table 1: Simulation configuration parameters

In this simulation scenario the following assumptions were made:

- A simple schedule was used so only one initiator is performing transactions at any time. The simulation results will be the same also for the case of concurrent schedule when there is no network resource conflict between transactions (i.e. no idle slots).
- The traffic comprised of 32 Write and 32 Read transactions with payloads that have the maximum RMAP data length.
- The maximum number of routers in a path is equal to 4.
- Path addressing is used

In the simulation results the following metrics are presented for a varying number of parameters:

- Minimum duration of the time-slot (which is equal to the maximum total time to complete any transaction with an addition of a safe margin “Guard Time-slot margin” equal to 2μs, the time-slot intervals are rounded to 1μs resolution).
- Maximum data rate (which is equal to RMAP data length / Time-slot interval).
- Efficiency (which is equal to maximum data rate / maximum SpW data rate, the maximum SpW data rate is considered in full duplex and equal to 2 * 0.8 * SpW link speed)

These results are important to identify the “Best Case” performance of the SpW-D protocol (i.e. fully utilized network with no network resource conflicts between the paths).

3.3.2 Simulation results

3.3.2.1 Minimum Time-slot interval and maximum data-rate vs data length for 2Mbps SpW link speed

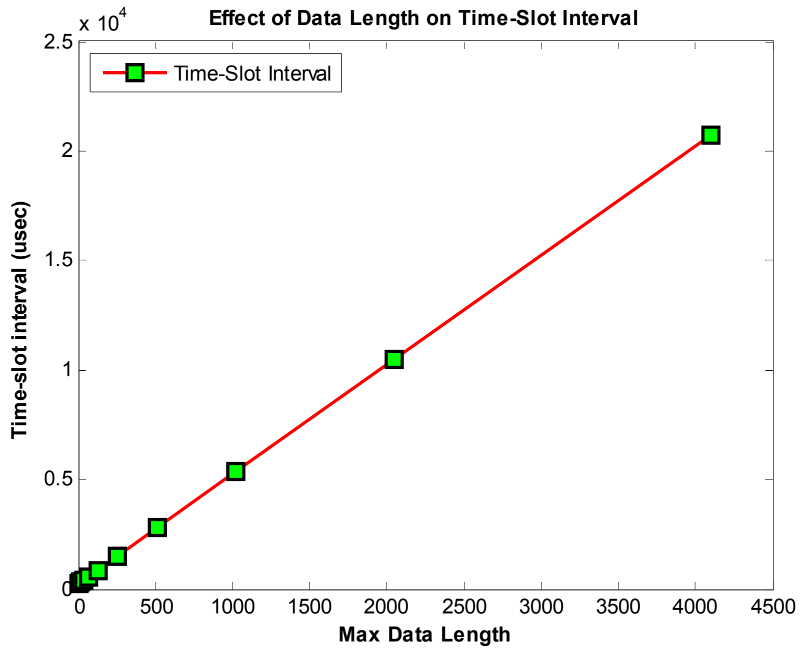


Figure 4: Minimum Time-slot interval vs data length

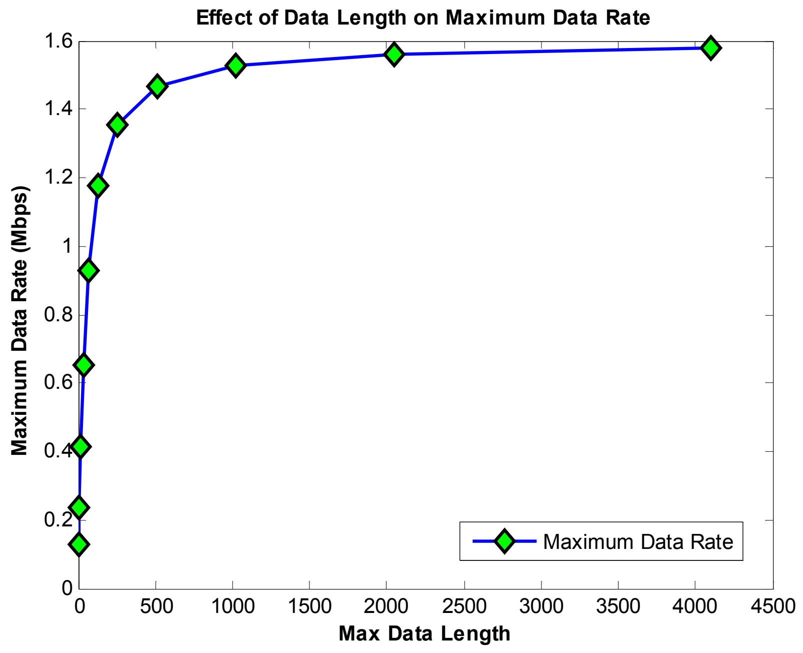


Figure 5: Maximum data-rate vs data length

Data Length (Bytes)	Time-slot (μs)	Data Rate (Mbps)
4	250	0.13
8	270	0.24
16	310	0.41
32	391	0.65
64	551	0.93
128	871	1.18
256	1512	1.35
512	2794	1.47
1024	5358	1.53
2048	10487	1.56
4096	20743	1.58

Table 2: Time-slot interval and data rate for 2Mbps SpW link speed

3.3.2.2 Minimum Time-slot interval and maximum data-rate vs data length for 10Mbps SpW link speed

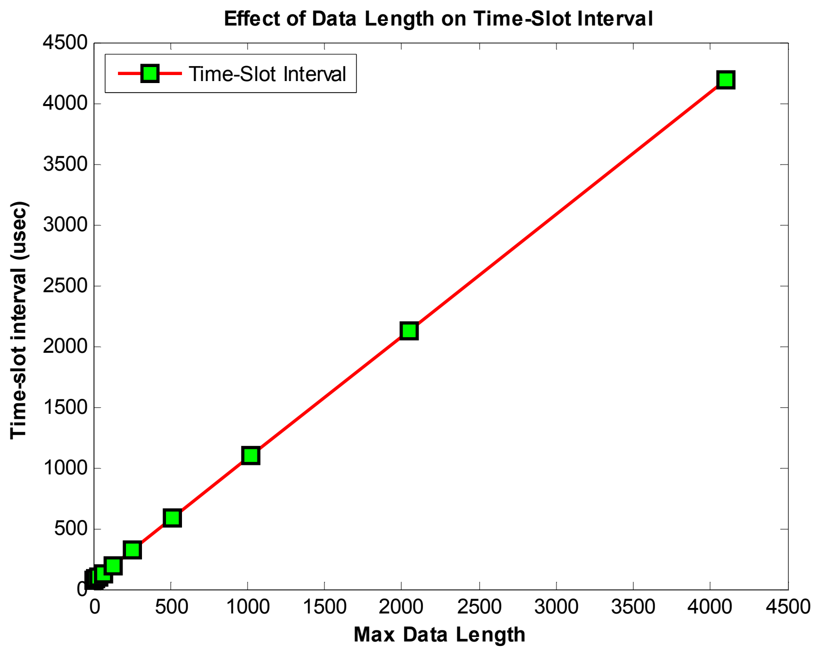


Figure 6: Minimum Time-slot interval vs data length

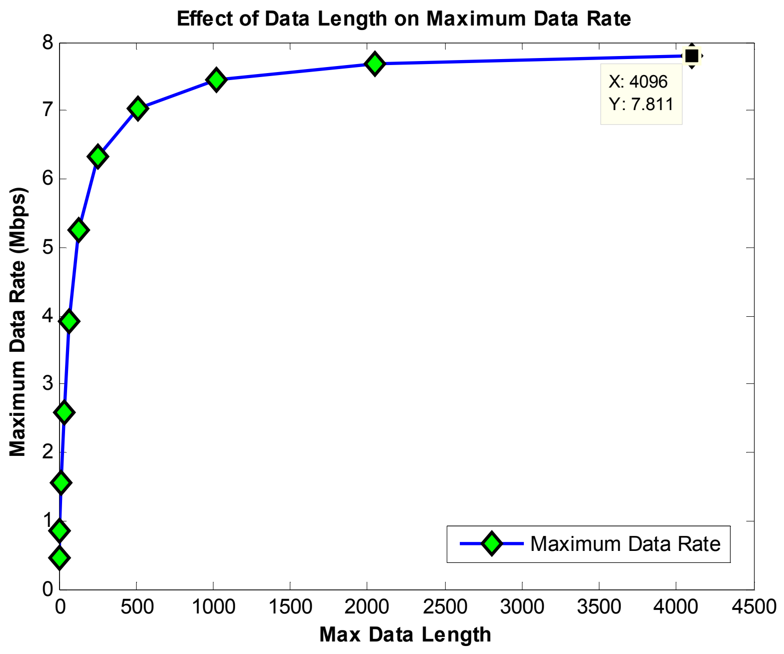


Figure 7: Maximum data-rate vs data length

Data Length (Bytes)	Time-slot (µs)	Data Rate (Mbps)
4	70	0.46
8	74	0.86
16	82	1.56
32	99	2.59
64	131	3.91
128	195	5.25
256	324	6.32
512	582	7.04
1024	1098	7.46
2048	2131	7.69
4096	4195	7.81

Table 3: Time-slot interval and data rate for 10Mbps SpW link speed

3.3.2.3 Minimum Time-slot interval and maximum data-rate vs data length for 50Mbps SpW link speed

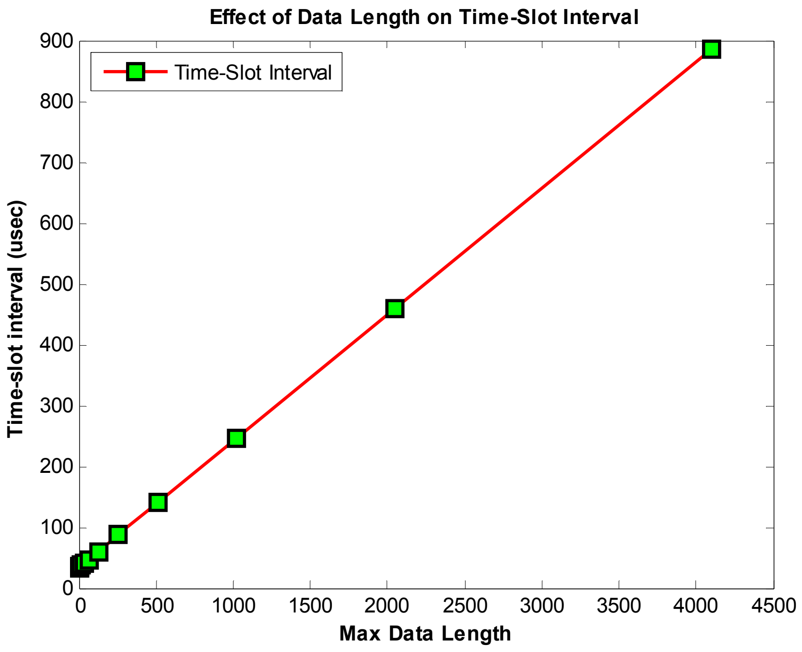


Figure 8: Minimum Time-slot interval vs data length

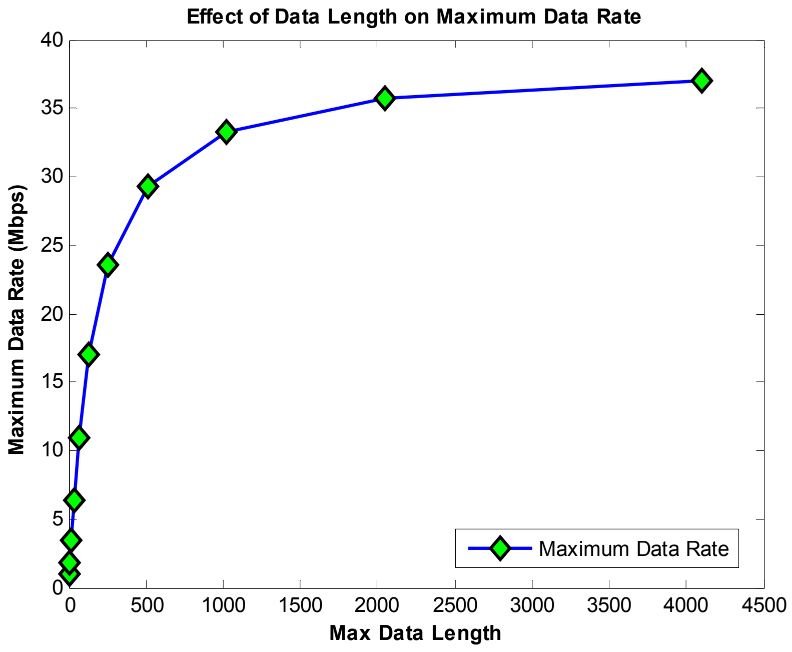


Figure 9: Maximum data-rate vs data length

Data Length (Bytes)	**Time-slot (μs)**	**Data Rate (Mbps)**
4	34	0.94
8	35	1.83
16	37	3.46
32	40	6.40
64	47	10.89
128	60	17.07
256	87	23.54
512	140	29.26
1024	246	33.30
2048	459	35.69
4096	885	37.03

Table 4: Time-slot interval and data rate for 50Mbps SpW link speed

3.3.2.4 Minimum Time-slot interval and maximum data-rate vs data length for 100Mbps SpW link speed

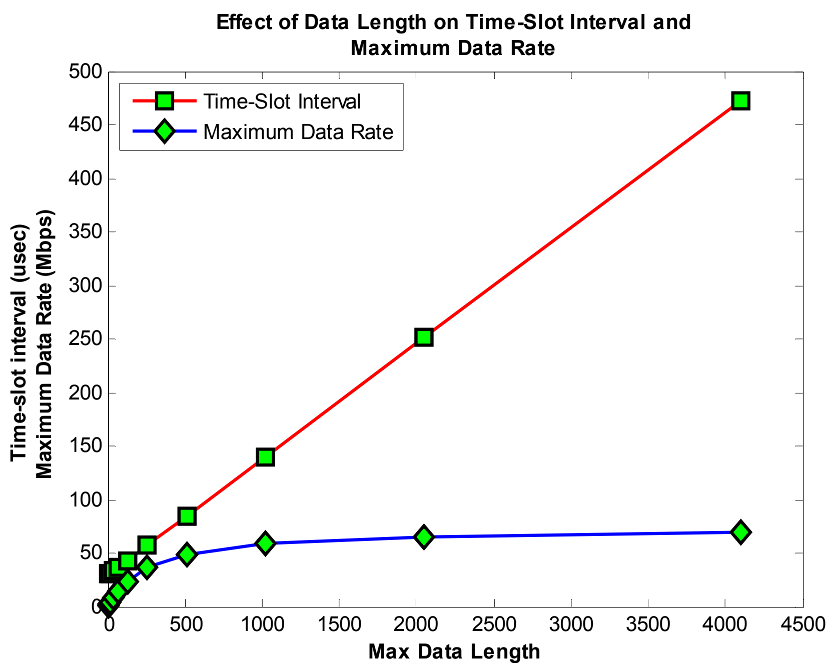


Figure 10: Minimum Time-slot interval and maximum data-rate vs data length

Data Length (Bytes)	Time-slot (μs)	Data Rate (Mbps)
4	30	1.07
8	30	2.13
16	31	4.13
32	33	7.76
64	36	14.22
128	43	23.81
256	57	35.93
512	85	48.19
1024	140	58.51
2048	251	65.27
4096	472	69.42

Table 5: Time-slot interval and data rate for 100Mbps SpW link speed

3.3.2.5 Minimum Time-slot interval and maximum data-rate vs data length for 150Mbps SpW link speed

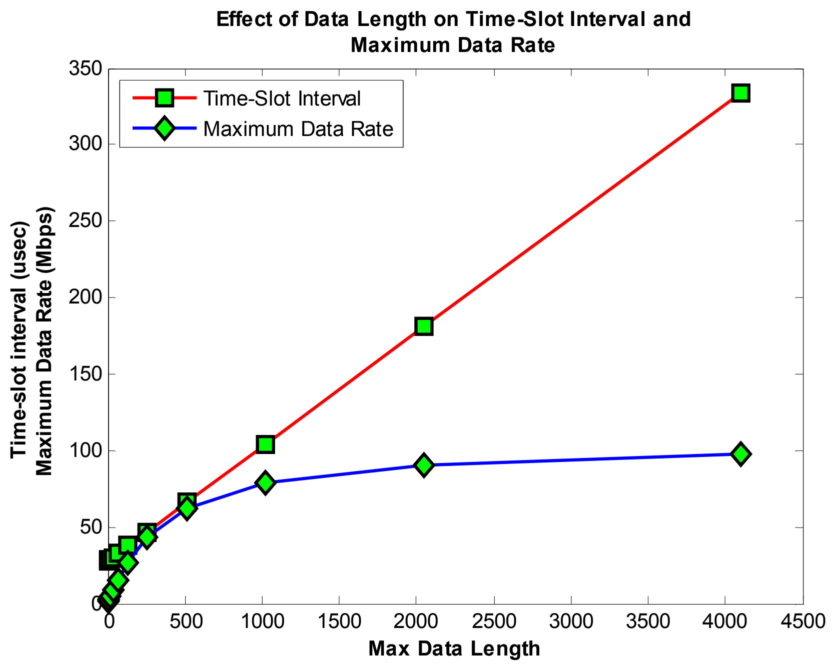


Figure 11: Minimum Time-slot interval and maximum data-rate vs data length

Data Length (Bytes)	Time-slot (µs)	Data Rate (Mbps)
4	28	1.14
8	29	2.21
16	29	4.41
32	30	8.53
64	33	15.52
128	38	26.95
256	47	43.57
512	66	62.06
1024	104	78.77
2048	181	90.52
4096	334	98.11

Table 6: Time-slot interval and data rate for 150Mbps SpW link speed

3.3.2.6 Minimum Time-slot interval and maximum data-rate vs data length for 200Mbps SpW link speed

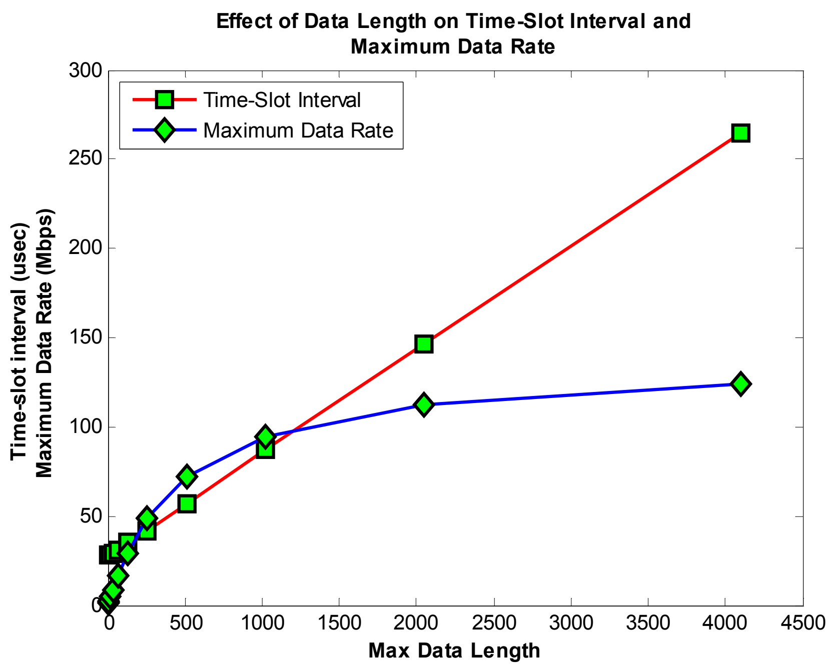


Figure 12: Minimum Time-slot interval and maximum data-rate vs data length

Data Length (Bytes)	Time-slot (µs)	Data Rate (Mbps)
4	28	1.14
8	28	2.29
16	28	4.57
32	29	8.83
64	31	16.52
128	35	29.26
256	42	48.76
512	57	71.86
1024	87	94.16
2048	146	112.22
4096	265	123.65

Table 7: Time-slot interval and data rate for 200Mbps SpW link speed

3.3.2.7 Minimum Time-slot interval and maximum data-rate vs data length for 400Mbps SpW link speed

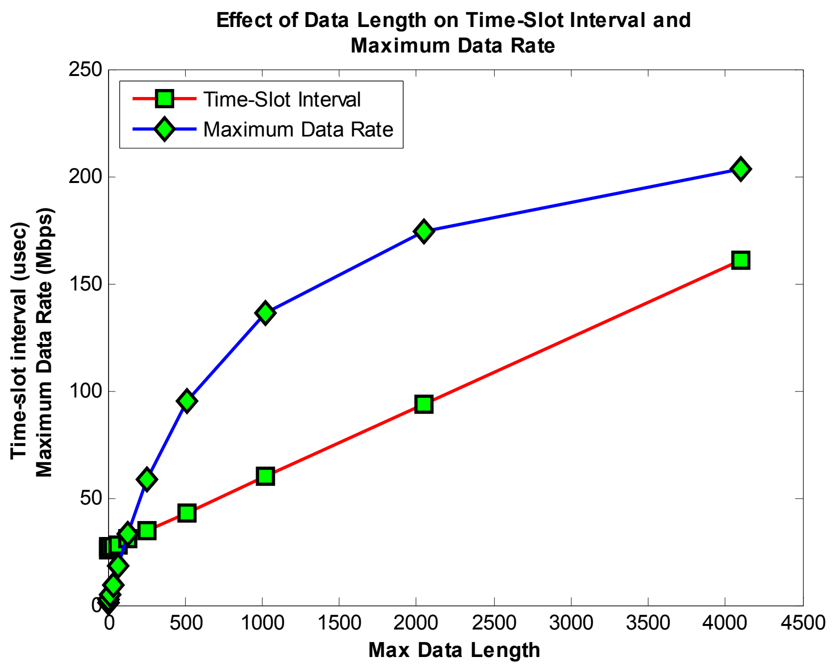


Figure 13: Minimum Time-slot interval and maximum data-rate vs data length

Data Length (Bytes)	Time-slot (µs)	Data Rate (Mbps)
4	26	1.23
8	27	2.37
16	27	4.74
32	27	9.48
64	28	18.29
128	31	33.03
256	35	58.51
512	43	95.26
1024	60	136.53
2048	94	174.30
4096	161	203.53

Table 8: Time-slot interval and data rate for 400Mbps SpW link speed

3.3.2.8 Minimum Time-slot interval and maximum data-rate vs RMAP target Memory Throughput for 200Mbps SpW link speed and 256Bytes maximum data length

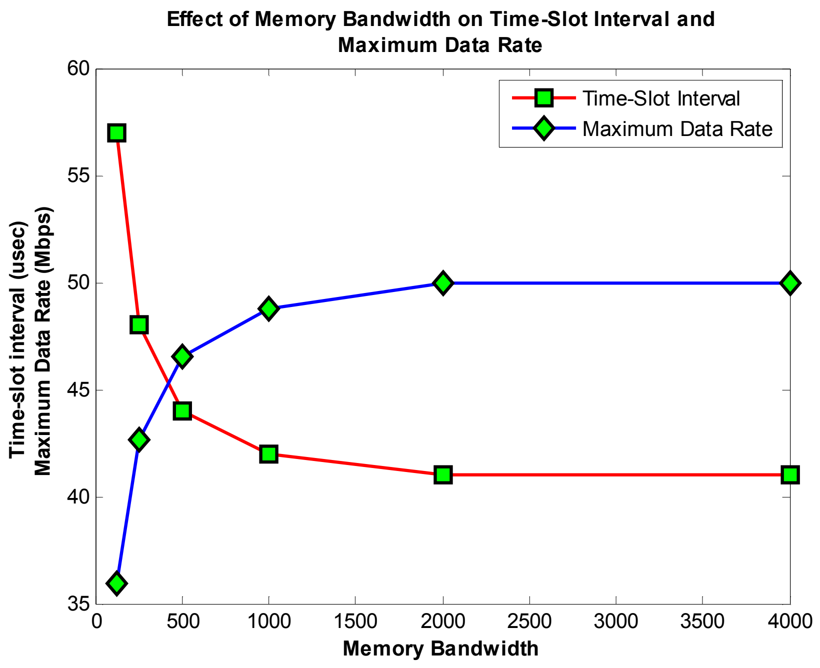


Figure 14: Minimum Time-slot interval and maximum data-rate vs RMAP target memory bandwidth

RMAP Target Memory Throughput (Mbps)	Time-slot (μs)	Data Rate (Mbps)
125	57	35.93
250	48	42.67
500	44	46.55
1000	42	48.76
2000	41	49.95
4000	41	49.95

Table 9: Time-slot interval and data rate for different Target memory throughputs

A noticeable result is that slow memories have a significant impact in the throughput and if the memory throughput exceeds a threshold then the data rate is not increased significantly.

3.3.2.9 Efficiency vs SpW link speed for 256Bytes maximum data length

In this simulation the efficiency is calculated for different SpW link speeds with a maximum data length of 256 Bytes.

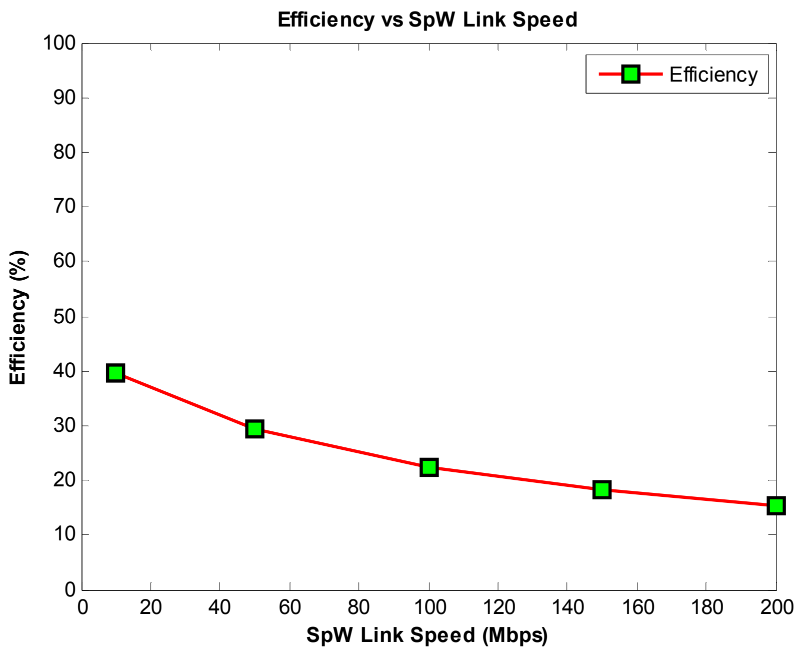


Figure 15: Efficiency vs SpW link speed

SpW link speed (Mbps)	Efficiency (%)
10	39.50
50	29.43
100	22.46
150	18.15
200	15.24

Table 10: Efficiency for different SpW link speeds

A noticeable result is that the efficiency drops as the SpW link speed increases; this is due to the constant delays (HW delays) that jeopardize the effective data throughput as the link speed increases. In low link speeds the transmit time of headers and payload is much greater compared to the constant HW delays and the efficiency increases.

3.3.3 Conclusions

As presented in the section 3.3.2.6 for a network with SpW links running at 200 Mbps and a maximum payload length of 256 Bytes, the maximum network data rate, when simple scheduling is used, or the maximum data rate per initiator, when concurrent scheduling is used, is around 48Mbps.

The efficiency in this case is 15% which is considered low.

The low efficiency and data rate, compared to the SpW link effective data rate, is the main performance issue for the protocol.

The low efficiency and data rate is the result of the following causes:

- SpW-D Scheduling and RMAP transactions do not utilize the full duplex SpW link capability.
  - This can not be mitigated in the SpW-D protocol as proposed in SpW-D Draft B
- Overhead imposed by RMAP header length
  - This can be mitigated by increasing the maximum data length or using multi-slot scheduling. The results of both of these approaches in maximum data rate and efficiency are similar*.
- Constant delays imposed by the HW implementation and the SpW router forwarding delay.
  - The HW implementation delays are much greater compared to the delays imposed by SpW routers (as defined in performance calculations of section 3.2 and in SpW-D Draft B). The impact of these delays is however reduced when the SpW link speed is decreased

*Note:

The performance of multi-slot scheduling is highly variable and depends on the number of time-slots selected to perform transactions with large payloads and the actual payload length that is going to be transferred by each transaction. Theoretically a multi-slot transaction can occupy the whole schedule table and last 64 time-slots. Due to this fact simulation results have not be provided but can be calculated for multi-slot transactions with payloads up to 4096 Bytes using the previous simulation results.

Assuming the simulation results of section 3.3.2.6 for a network with SpW links running at 200 Mbps and a maximum payload length of 254 Bytes the time-slot duration will be 42μs. A multi-slot transaction with 4096 bytes payload requires a total duration of 265μs and 7 time-slots to complete. The maximum number of multi-slot transactions that can be performed in the epoch interval is 9. If we assume that 9 such transactions are performed (will occupy 63 time-slots) and in the last time-slot a single transaction is performed. The total amount of data transferred is 9*4096+256 = 37120Byes. The data rate for the duration of this epoch (2688msec) shall be 110.47Mbps. If simple schedule is used with 4096Bytes maximum payload then the data rate is 123,65Mbps (it is better than the multi-slot schedule because the multi-slot transactions do not occupy entirely all the 7 allocated time-slots in this example).

4 Analysis and open issues in SpW-D Draft B specification

This section presents open issues including initially identified problems, potential performance issues and open points for the SpW-D Draft B specification.

4.1 Scheduling issues

SpaceWire-D inherits two fundamental problems from its simple TDMA access scheme, which assigns fixed time-slots to each initiator.

- Waste of network resources in the presence of asynchronous traffic, caused by time-slots allocated to initiators that have no traffic to transmit.

  This problem has been solved in some TDMA based protocols using dynamic scheduling or time-slot reservation (e.g. in GSM) or in some cases by dividing the time-slots in different periods for isochronous and asynchronous communications (e.g. in FlexRay).
- Fixed size of the time-slot must be carefully selected based on communication requirements. If time-slot duration is relatively small it will cause overhead (due to RMAP header in each SDU segment) to initiators that need to transmit large payloads. If time-slot duration is relatively large it will waste network bandwidth for initiators that need to transmit small payloads.

Another fundamental problem in SpaceWire-D is the following:

- SpaceWire-D does not utilize the full duplex SpW link and path capability and performance is degraded. In any transaction the path is half-duplex since the Initiator sends the Command and then pauses waiting for the Target Reply (this includes the end-to-end path from the Initiator to the Target, including core network links that connect different SpW routers, for the entire duration of a time-slot).

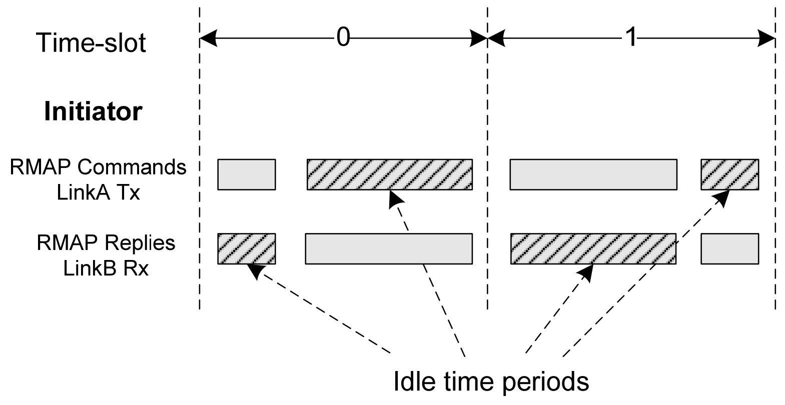


Figure 16: Idle periods in SpaceWire-D RMAP transactions

4.1.1 Scheduling types

Simple Scheduling

Simple schedule gives an initiator full control of the network for one or more specified time-slots. This means that when that initiator is permitted to send an RMAP transaction it may do so to ANY target node on the network and no other initiator node can access the network at the same time-slot.

In the following scheduling example OBC can access targets only in time-slots S2-S63 and MM can access targets only in time-slots S0-S1 in each epoch.

Time-Slot	S0 - S1	S1- S2	S2-S3	S3-S63
OBC	**YES**	**YES**	**NO**	**NO**
MM	**NO**	**NO**	**YES**	**YES**

Figure 17: Simple-schedule table

- In simple scheduling the network capacity is shared between initiators and performance is degraded due to the fact that an initiator can not transmit in time-slots reserved for another initiator.

  SpW-D Draft B has proposed the use of concurrent schedule to overcome this issue.

Concurrent scheduling

The concurrent schedule makes more efficient use of network bandwidth by allowing more than one initiator to initiate RMAP transactions in a time-slot. This gives rise to the possibility that two initiators might attempt to use the same network resources (SpaceWire links) at the same time. The schedule table has to be constructed to prevent this.

More than one initiator may initiate RMAP transactions in the same time-slot provided that the paths from each of the initiators to their targets do not use any of the same SpaceWire links in the network.

We can conclude that in order to avoid conflicts, the targets that an initiator is allowed to initiate transactions with, in each time-slot, shall have no network resource conflict (i.e. no shared link from initiator to target path) with any other initiator's scheduled transactions for the same time-slot.

Example:

Assume the following star network topology.

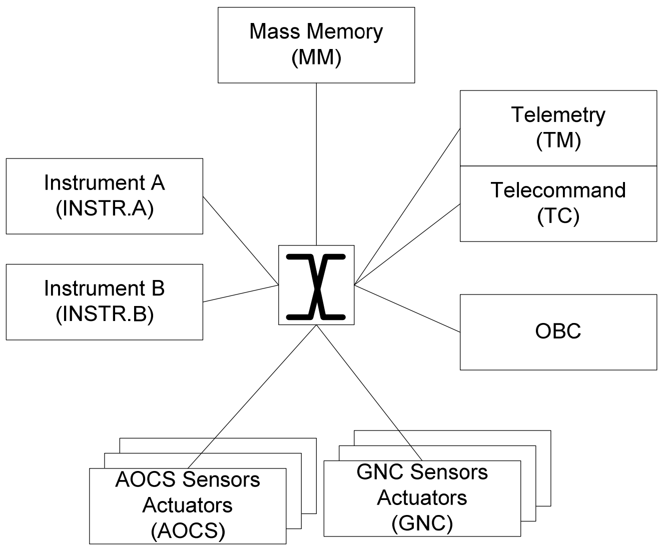


Figure 18: Example network topology

In this example concurrent schedules are used for OBC and Mass Memory (MM) initiators, which initiate transactions with different targets with the restriction that paths from initiators to targets do not conflict in the same time slot. Using the following schedule tables a set of time-slots from S0 to S2 are allocated to OBC in order to perform data handling with INSTR. A, INSTR. B, TM as well as data transfers from/to Mass Memory (MM).

Time-Slot	S0 - S1	S1- S2	S2-S3	S3-S63
OBC	**INSTR. A, TM, MM**	**INSTR. B, TM, MM**	**AOCS**	**GNC**
MM	**-**	**-**	**INSTR. A, INSTR. B, TM**	

Figure 19: Concurrent schedule table

In this scheduling configuration the sets of time-slots from S0 to S1 can not be used by MM to access any target in order to avoid conflicts with potential OBC transactions accessing MM at any of these time-slots. This will degrade the average data rate of the MM initiator since it must be idle for a potential large number of time-slots.

If specific time-slots (less than S2) are defined for the RMAP transactions from OBC to MM in order to overcome this problem then the previous schedule table can be configured as in the following figure.

Time-Slot	S0 - S1	S1- S2	S2-S3	S3-S63
OBC	**INSTR.A, INSTR.B, TM, MM**	**INSTR. B, TM**	**AOCS**	**GNC**
MM	**-**	**INSTR. A**	**INSTR. A, INSTR. B, TM**	

Figure 20: Concurrent schedule table

The OBC can perform transactions to Mass Memory only in time-slots from S0 to S1. In this configuration more time-slots are allocated to Mass Memory from S1 to S2, where it can access INTST.A, As a result the average throughput of the Mass Memory initiator can be increased but in epochs where large data transfers are required to/from Mass Memory the effective throughput of these transfers (OBC accessing MM) will be reduced and transfer latency will increase due to the reduced number of allocated time-slots for OBC-MM transactions in these epochs.

Based on this example we conclude that:

- In concurrent schedule the performance is degraded in some scenarios. The use of different schedule tables in different epochs (synchronised for all initiators) will be beneficiary and will improve network efficiency further.

A possible solution to overcome this problem is presented in the following section ("Concurrent scheduling with different scheduling tables in different epochs").

Concurrent scheduling with different scheduling tables in different epochs (new concept)

A possible solution to further improve efficiency can be the use of concurrent scheduling with different scheduling tables used in different epochs. Such scheduling scheme is not defined in SpW-D Draft B.

Assuming that at predefined time periods (T1 – T3) specific scheduling tables are used from each initiator then the waste of network resources can be further minimized. Applying such scheme to the previous case will result in the following scheduling tables

Time-Slot	S0 - S1	S1- S2	S2-S3	S3-S63	
OBC	INSTR. A	INSTR. B	AOCS	GNC	T1
	TM		AOCS	GNC	T2
	MM		AOCS	GNC	T3
MM	INSTR. B	INSTR. A	INSTR.B	INSTR. A	T1
	INSTR. A		INSTR. B		T2
	-		TM		T3

Figure 21: Schedule-tables of initiators

In this case the MM is only idle in the set of time-slots S0 – S2 only in period T3 when it is being accessed by the OBC and acts as target. Additionally, a sequential number of time-slots are again allocated for OBC in time-slots S0-S2 and it can perform multi-slot scheduling to increase the data rate and efficiency of OBC to MM transactions for the transfer of large payloads.

This scheduling access scheme will require a signalling mechanism for synchronising the initiators at the different epochs in order to use different scheduling tables or schedule different transactions in each time-slot and avoid network conflicts. Such synchronisation problem can occur after a reset of MM, where it uses the first scheduling table (period T1) and acts as initiator and OBC at the same time uses another scheduling table (e.g. of period T3) where it initiates transactions to MM and MM shall act as target.

Such access schemes are common in many TDMA protocols where time-slots are divided in a frame hierarchy with TDMA frames composed by a number of time-slots and multi-frames or super-frames composed by a number of TDMA frames that are repeated in specific intervals.

A new concept to apply such scheme is presented in section 6.1.3.

Multi-slot scheduling

The multi-slot schedule builds on the concurrent schedule to improve network efficiency further. Where a large amount of data has to be transferred between two nodes, the RMAP transaction to accomplish this is permitted to occupy more than one adjacent time-slot. This allows more data to be transferred in one RMAP transaction. The schedule has to ensure that no conflict of network resources occurs over the duration of this extended RMAP transaction.

In case that large data transfers are known a priori at network design time, multi-slot scheduling will improve network performance by reducing RMAP protocol overhead in such transfers.

As an example, assume that OBC initiates long RMAP transactions with targets INSTR.A, INSTR.B, TM and MM in the set of time-slots S0 and S1. Using multi-slot scheduling will greatly reduce RMAP overhead and improve network performance for these transactions as presented in the following figure.

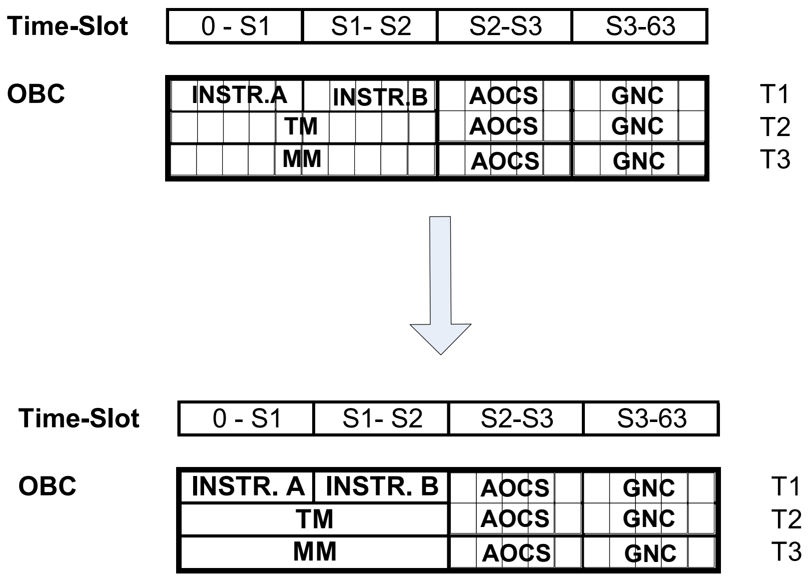


Figure 22: Modification of a schedule-table using multi-slot scheduling

- In Multi-slot scheduling, as defined in SpW-D Draft B, there are no open issues or considerations currently identified. This section is presented for consistency reasons.

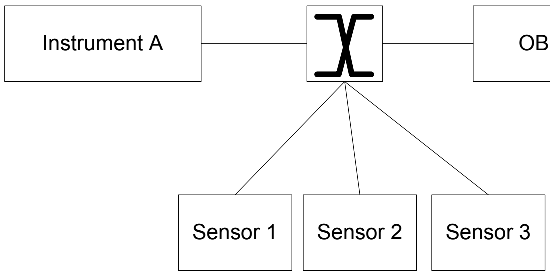
Multi-transaction scheduling (new concept)

The current SpW-D specification restricts the number of allowed RMAP transactions to one per initiator in each time-slot. This simplifies implementation of the SpW-D protocol logic, but several issues may arise. Assume the following example:

1. The OBC performs RMAP reads to a number of sensors (e.g. for AOCS) and to an instrument
2. RMAP reads from all sensors return a reply with a payload of a few bytes (e.g. 4 Bytes)
3. RMAP reads from Instrument A returns a reply with a payload of 512 Bytes

According to the current SpW-D specification, the following problems arise

- Since the time-slot duration is set according to the maximum allowed payload, the majority of time is wasted when the OBC is performing RMAP reads from the sensors (time-slots S0, S1 in Figure 23).
- As a side effect, one time-slot (a scarce resource) is consumed for each sensor in the network.



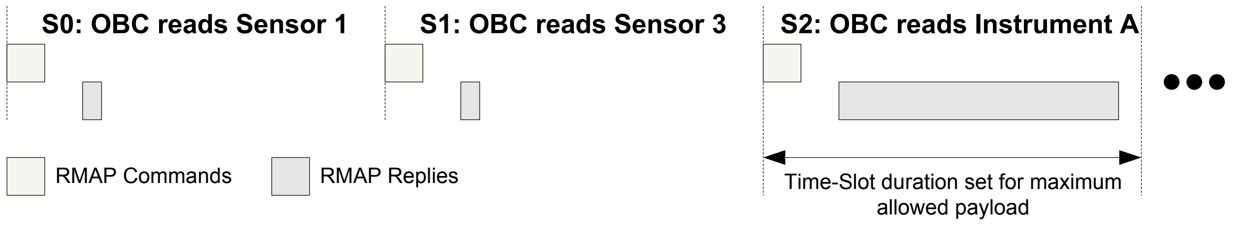

S0: OBC reads Sensor 1
S1: OBC reads Sensor 3
S2: OBC reads Instrument A
RMAP Commands
RMAP Replies
Time-Slot duration set for maximum allowed payload

Figure 23: Example scenario for RMAP transactions with small and large payloads

A possible solution to overcome the previously mentioned problems is to allow more than one transactions in time-slots (or in some of them), improving bandwidth utilization and saving time-slots in networks in which the length of the RMAP payload varies significantly.

Such approach (namely Multi-transaction scheduling) in not defined in SpW-D Draft B but is possible to be implemented to improve network efficiency further, in cases where small amounts of data have to be transferred between an initiator and a number of different target nodes, by permitting an initiator to initiate more than one RMAP transaction in the same time-slot. The schedule has to ensure that no conflict of network resources occurs over the duration of this time-slot and that all RMAP transactions can be finished in the duration of the time-slot. Multi-transaction scheduling is presented in the following figure.

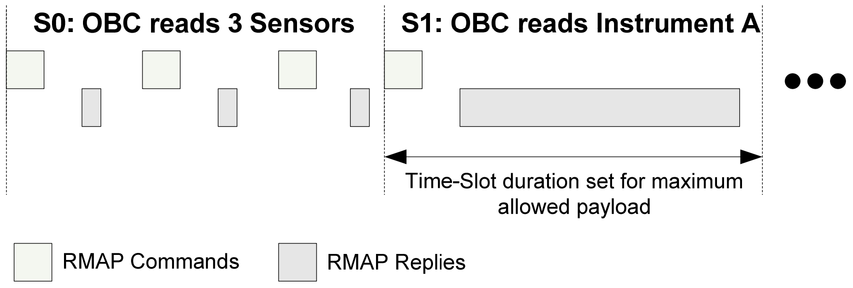

S0: OBC reads 3 Sensors
S1: OBC reads Instrument A
Time-Slot duration set for maximum allowed payload
RMAP Commands
RMAP Replies

Figure 24: Example scenario for a time-slot with multiple RMAP transactions

An initial analysis of Multi-transaction scheduling is presented in section 6.1.1.

4.1.2 Scheduling large data transfers

The main issues for scheduling large data transfers in data handling communications are the following:

- Large data transfers require a large number of time-slots to complete.
- Pre-allocating a large number of time-slots may waste network resources when transfers are not periodic.
- Pre-allocating a small number of time-slots increases the time required for the completion of the large data transfer.

Assume the example network topology in Figure 18, and a payload of 10MB raw data acquired from Instrument A (e.g. camera) that must be transferred to Mass-Memory (initiator) with a time-slot of 42μs, max data length of 256 bytes and SpW link speed at 200Mbps.

This SpW-D transfer will require 10MB / 256B = 40960 slots, 40960/64 = 640 epochs and a total transfer time of 40960*42µs = 1.72sec and this for the case that the Mass Memory is scheduled to initiate transaction in all the 64 time-slots.

An issue in this case is that 640 full epochs are required for such transfer without taking into account that the mass-memory and Instrument A must also be accessed from OBC in specific time-slots.

With the current version of the SpW-D specification, the following alternatives are possible for the configuration of initiator's schedule tables (Mass-Memory and OBC for this example) with different constraints:

- Pre-allocate a large number of slots per epoch for large data transfers, If large data transfer needs to be performed at large intervals (e.g. every 10sec) the slots pre-allocated for large data transfer will waste network resources since there will be many idle slots in many epochs.
- Pre-allocate a small number of slots, this will linearly increase the large data transfer time.

As a result this is a network design issue which can be critical if the nature of large data transfer is asynchronous (i.e. not periodic) because the more slots are allocated, transfer time will decrease but waste of resources will increase and vice versa.

A potential solution for this problem is to divide the slots in different periods for isochronous and asynchronous communications (e.g. as in FlexRay), such concept is presented in Section 6.1.2.

4.1.3 Scheduling command and control communications

The main issues regarding scheduling of command and control communications in SpW-D are the following:

- **Issue 1**: Control loops in command and control communications (e.g. AOCS) need to be performed at large periods (e.g. >20 ms) compared to epoch interval (e.g. in case of 200Mbps SpW links and a maximum RMAP payload of 256 Bytes the time-slot duration is 42µs and the epoch interval is equal to 64*42µs = 2.688msec)
- **Issue 2:** Control loops may need to perform reads and writes from/to different sensors actuators with small latency between transactions. This requires allocating consecutive time-slots for these transactions and consumes a large number of time-slots in the Schedule table.
- **Issue 3:** The potential large number of sensors and actuators used in modern S/C architectures will require a large number of transactions and given the fact that only one transaction is allowed per time-slot, results in a large number of time-slots for the completion of the transactions required for control loop. In the case of complex topologies and a common network for command and control and data handling, If a large number of time-slots is allocated for command and control communication in a epoch then a small number of time-slots will remain for data handling and vice versa.
- **Issue 4:** If the payload of control loop RMAP transactions is relatively small (e.g. 4Bytes) compared to the maximum RMAP data length (e.g. 256Bytes) there will be waste of network bandwidth, since the initiator will remain idle for a relatively large percentage of the time-slots used for command and control communications.

If the payload of command and control communications is relatively small then a potential solution for these issues is to use multi-transaction scheduling initiating more than one transactions per time-slot in order to reduce the total number of required transactions (issue 3) and increase network efficiency (issue 4).

Additionally a technique for concurrent scheduling with different scheduling tables used in different epochs may be used, as presented in the previous section. This will allow the allocation of transactions for command and control to specific epochs increasing the period of control cycles (issue 1) and allowing all transactions to be in consecutive time-slots (issue 2) allowing unallocated time-slots in other epochs for the data handling communications.

Such solutions are presented with more detail in Section 6.1.

4.2 RMAP transactions overhead

In some cases the RMAP transaction may introduce overhead for data transfers.

- If the initiator needs to transfer data from one target to another (e.g. OBC commands a transfer from instrument to mass memory) the same payload must be transferred from target to initiator and then from initiator to target. A possible solution for a direct target to target transfer using RMAP will improve performance while not requiring complex initiator-target node implementations

A solution based on SpW-D and standard RMAP would require that one of the targets should also have a role of initiator (e.g. OBC write a command to mass memory to trigger the transfer, mass memory acts as initiator and starts to perform read transactions from instrument).

A possible solution for a direct target to target transfer using RMAP has been presented in [SC2008].

An alternative solution is provided in section 6.2.

4.3 Possible issues from time-code distribution

According to SpW specification [AD,4] page 81, the accuracy with which system time can be distributed is dependant upon the number of links over which it is distributed and the corresponding operating rate of each of those links. with a time - skew across a network of at least STskew = 14N/R, where N is the number of links traversed and R is the average link operating rate. Across a network, this gives rise to a total jitter of STjitter = 10N/R.

This delay must be considered in time-slot duration because in worst case scenarios (many routers - low SpW link speed) this may cause network resource conflicts due to delayed time-code reception from nodes that are many links away from time master and early time-code reception from nodes that are close to the time master, as presented in the following figure.

In the following example if we assume that at time-slot 6 initiator B initiates a transaction with initiator A (which acts as target) then in the next time-slot initiator B will receive a time-code while being in transmit mode sending the reply to initiator B, which may cause an error at the initiator.

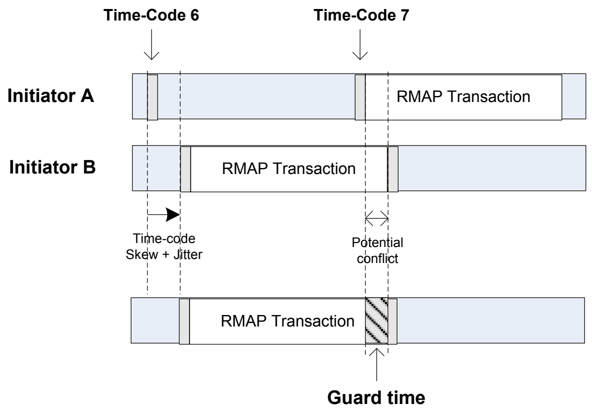


Figure 25: Potential conflicts due to time-code distribution latency

Based on this issue we can conclude that:

- Time-code distribution delay must be considered in time-slot duration because in worst case scenarios this may cause network resource conflicts due to delayed time-code reception from nodes that are many links away from time master and time-code reception from nodes that are close to the time master.

  In order to avoid such network resource conflicts due to time-code distribution skew and jitter a possible solution would be to add a safe margin (namely Guard time in the previous figure) at the time-slot duration equal to STskew_max + STjitter_max.

4.4 Time-slot violation issues

In SpW-D the time-slot duration is set to allow a maximum RMAP payload for Read and Write commands whereas the RMAP transaction must finish within the time-slot interval.

There are some issues regarding a potential violation of time-slot interval that shall be handled by SpW-D protocol:

- If at the initiator there is a delay in transmission of a command due to SpW flow control (e.g. RMAP target does not digest SpW N-Chars within a defined interval) or any other cause for delayed transmission of command. RMAP transaction may not finish until the end of time-slot and it may cause a conflict in the next time-slot with a RMAP transaction from another initiator.

A potential solution to avoid this problem is that the Scheduler shall maintain a timer for the maximum allowed transmit time (which is different for the Read and Write commands) and shall inject EEP at the RMAP command if the timer expires (this concept is the same as KILL function in SpW-T protocol specification).

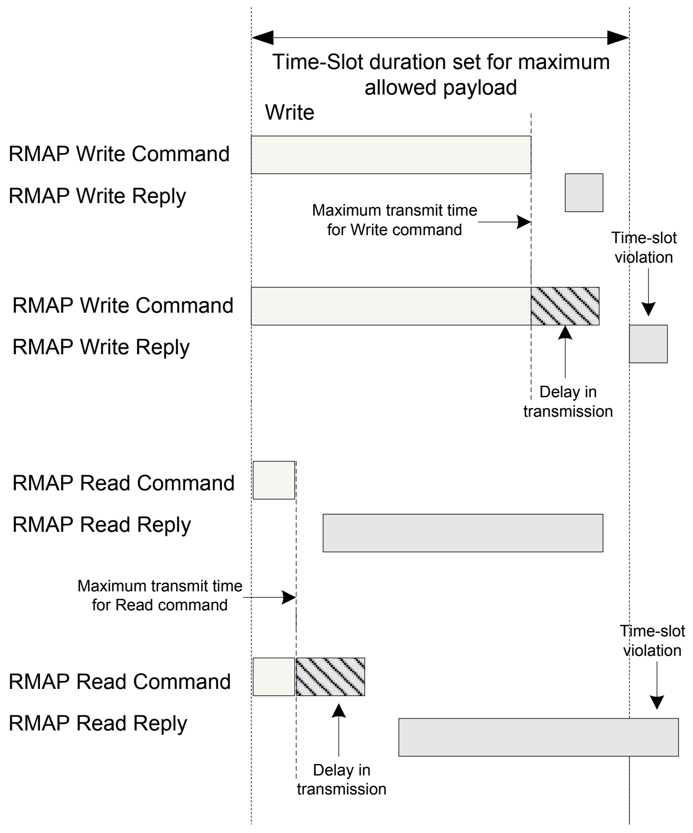


Figure 26: Potential conflicts due to delayed transmission of command

4.5 Segmentation and Retry

Segmentation and Retry have not been defined in SpW-D Draft B.

In this section an initial analysis is performed and implementation issues are presented. In Sections 6.3 and 6.4 different alternatives and considerations for future implementations are provided for the Segmentation and Retry functions.

4.5.1 Segmentation mechanism

In SpW-D the maximum length of the RMAP data field is limited (either to a maximum payload length e.g. 256 bytes or N x maximum payload length when multi-slot schedule is used, where N is less or equal to 64 time-slots) and as a result segmentation functionality may be required.

For a write transfer request of a large payload from initiator to target, the initiator will receive a SDU from an upper layer protocol or interface and will segment the SDU and transmit it within multiple RMAP PDUs whose payload lengths depend on the maximum payload supported by the underlying SpW-D network.

The target will perform reassembly when all PDUs have been received and may provide a write transfer indication to any upper protocol layers or interfaces. (Note that in this case the target may be a SpW-D Initiator/Target node which acts as target)

For a read transfer request of a large payload (i.e. SDU) from target to initiator, the initiator will receive a request from an upper layer protocol or interface and shall initiate a number of RMAP read transactions to receive the PDUs.

The target may perform segmentation of the SDU into PDUs, store the PDUs in its target memory space and send them in the RMAP replies.

The initiator will perform reassembly when all PDUs have been received in order to provide the reassembled SDU to the upper protocol layers or interfaces.

Some issues regarding the segmentation are the following:

- How can the target distinguish between RMAP payload containing a SDU segment and a standard RMAP payload?
- How can the target know the exact size of the SDU in order to perform reassembly?
- How can the target identify that the payload of the RMAP write transaction is the start, middle or end segment of a SDU?
- How shall the target react is case of out of order delivery of a PDU? (assuming that such case can be caused by the Retry mechanism)?

A possible implementation of the Segmentation mechanism is presented in detail in section 6.3.

4.5.2 Retry mechanism

If it is important to provide reliability mechanisms for the SpW-D then it is necessary to extend the SpW-D in order to add a Retry mechanism. The Retry function shall provide a mechanism for re-initiating RMAP transactions that have failed.

In SpW-D the retry of a failed RMAP transaction can be initiated only in another time-slot. An exception to this is the new concept of Multi-transaction schedule where more than one RMAP transactions can be initiated in the same time-slot, (presented in section 4.1.1. and in more detail in section 6.1.1).

Important issues that must be considered for the retry mechanism are the following:

- What is the maximum latency in the retransmission of a failed transaction?
- Is there a possibility for out of order reception of PDUs if segmentation is used?
- Is network efficiency affected?
- Is the retry mechanism compatible with all the scheduling techniques?
- Does the implementation of the retry mechanism introduces significant problems?
- **A major issue for the implementation of the Retry mechanism is to define the time-slot(s) in which a failed RMAP transaction is allowed to be re-transmitted.**

The selection of the time-slot(s) in which a failed RMAP transaction is allowed to be re-initiated also defines the main mechanism for the implementation of the Retry. There are four initially identified alternatives:

1) Use a dedicated time-slot. For example, specific time-slot(s) may be allocated for retries at the end of the epoch. In this case the Retry is performed with bounded latency in a known time-slot and does not interfere with the transmission of scheduled transactions in other time-slots.
2) Use the next **free** time-slot that is allocated for the particular target (or channel) communication. The retry in this case may be sent in any time-slot scheduled to handle the same initiator/target pair as the failed RMAP transaction, provided that this time-slot is not needed for the initiation of an RMAP transaction. In this case the Retry is performed with lower priority than scheduled transactions.
3) Use **any** next time-slot that is allocated for the particular target (or channel) communication. The retry in this case may be sent in any time-slot scheduled to handle the same initiator/target pair as

the failed RMAP transaction even if this time-slot is needed for the initiation of an RMAP transaction. In this case the Retry is performed with higher priority than scheduled transactions.

4) Use a number of dedicated time-slots for asynchronous traffic – namely asynchronous segment (this new concept is presented in section 6.1.2) and perform retransmission in the beginning of the asynchronous segment with higher priority than the asynchronous traffic. In this case the Retry is performed with higher priority than asynchronous traffic and does not interfere with transmission of scheduled transactions.

These four alternatives of the Retry mechanism are presented and analysed in detail in section 6.4.

5 Protocol issues towards implementation

SpW-D Implementation in hardware presents several challenges that have to deal with design extensibility, the strict timings that shall be met in order to maximize throughput, the placement in the protocol stack, the use of resources etc.

5.1 System Architecture issues

At system design level the SpW-D timings impose requirements regarding the bus traffic, the way the bus arbiter handles simultaneous requests from multiple masters and the capabilities supported by the system bus masters and slaves.

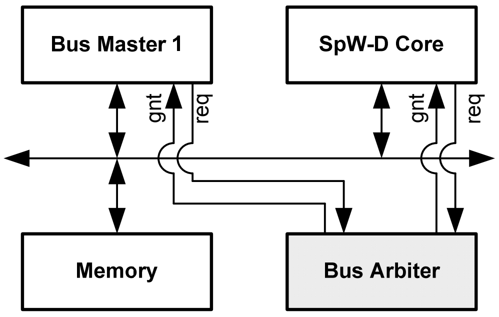


Figure 27: A typical system with the SpW-D Core integrated

Figure 27 shows a representative system into which the SpW-D Core will be integrated. It consists of two bus masters (Bus Master 1 and SpW-Core) a memory accessed by both masters and the bus arbiter. The masters issue requests for bus ownership to the arbiter and the arbiter asserts the ”gnt” signal to one of them assigning bus ownership.

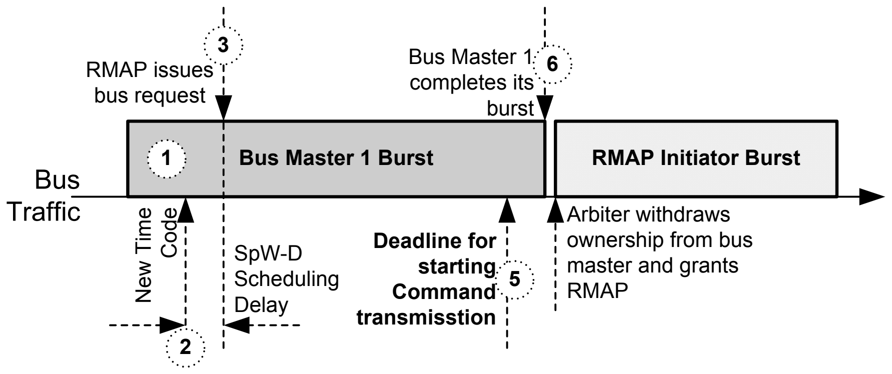

3
RMAP issues bus request
Bus Master 1 completes its burst
6
1
Bus Master 1 Burst
RMAP Initiator Burst
Bus Traffic
New Time Code
SpW-D Scheduling Delay
2
Deadline for starting Command transmisstion
5
Arbiter withdraws ownership from bus master and grants RMAP

Figure 28: Bus traffic of the example system

Depending on the capabilities of the bus arbiter, the bus masters and slaves and the transactions allowed by the bus specification, then the following scenario is possible:

- Step 1: Bus Master 1 performs a burst to the memory.
- Step 2: Time-code is received while the burst is in progress.
- Step 3: After the scheduler has determined which packet shall be transmitted, the RMAP Initiator requests bus ownership to fetch the command from the memory.
- Step 4: The bus arbiter does not have the capability to interrupt an on-going burst.
- Step 5: The deadline for transmission expires.
- Step 6: Bus is granted to the RMAP Initiator after the deadline for the start of transmission.
- Step 7: Since transmission starts late, time-slot boundaries will most probably be violated.

RMAP Targets can handle read packets with payload length of up to 16 Mbytes. This means a read reply cannot be fetched in a single burst, stored in their Tx FIFO and then transmitted. Instead, segments of the payload are fetched in the Tx FIFO and as this is emptied, new bursts are initiated to fetch the rest of the payload.

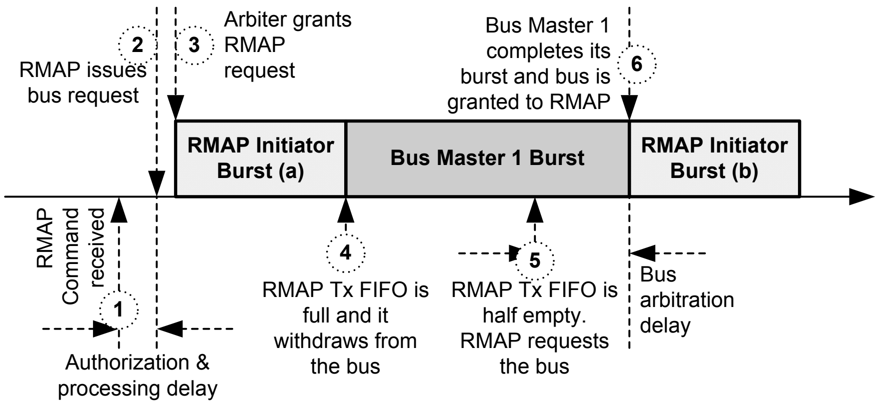

2
RMAP issues bus request
3
Arbiter grants RMAP request
Bus Master 1 completes its burst and bus is granted to RMAP
6
RMAP Initiator Burst (a)
Bus Master 1 Burst
RMAP Initiator Burst (b)
RMAP Command received
1
Authorization & processing delay
4
RMAP Tx FIFO is full and it withdraws from the bus
5
RMAP Tx FIFO is half empty. RMAP requests the bus
Bus arbitration delay

Figure 29: A SpW-D Target transmission with possible time-slot violation

In case such a core is used in a SpW-D network then the problem shown in Figure 29 may occur. As soon as the RMAP command is received, authorized and decoded by the target a bus request is issued. After the request is granted, the target fetches the payload until its Tx FIFO becomes full. At this point the target backs-off and waits until the FIFO becomes half-empty in order to initiate the next burst.

However, at this time another master may own the bus and the target shall wait until the end of the burst. In case the burst is very long, it may be the case that the Tx FIFO becomes empty until the initiation of the next burst. This may means that the transmission will be with gaps and it is very possible that the read reply will violate the time-slot boundaries.

As shown with the examples above the SpW-D timing requirements impose the following requirements at system design level:

- The bus arbiter shall support interleaved bursts
- The bus arbiter shall either
  - o support static priorities and the SpW-D Core shall be connected to the highest-priority port, or
  - o support different priority levels for transactions, so that a Master can assert the "high priority" signal in case it is about to miss a deadline, or
  - o bound the time of continuous bus ownership for each master and define a back-off period for each one
- The bus masters shall support interleaved bursts, i.e. they shall have the capability to be interrupted by the arbiter and resume their burst after they are re-assigned the bus.
- The bus slaves shall be capable of identifying bursts that have been interrupted.
- The bus transactions shall include "Unspecified Length Bursts" in order to resume a burst that was stopped at a non-aligned address.
- The system designer shall ensure that atomic transactions (cannot be interrupted even by higher priority requests) are time-bounded.

5.2 SpW-D Core Architecture issues

In order to build SpW-D on top of existing RMAP Cores, one has to take into account the RMAP Core's functionality. What is critical for SpW-D Core design is the way the RMAP Initiator works, since, according he the Draft specification, only the Initiator is required to have knowledge of the SpW-D timing and scheduling information.

Figure 30 shows they way a typical RMAP Initiator operates. A number of Commands are downloaded into its Transaction Table and are executed in a FIFO order. The Commands typically contain pointers to a memory that contains the actual RMAP payload to be transmitted along with header and control information (e.g. timeout period, address into which the Reply shall be stored etc.).

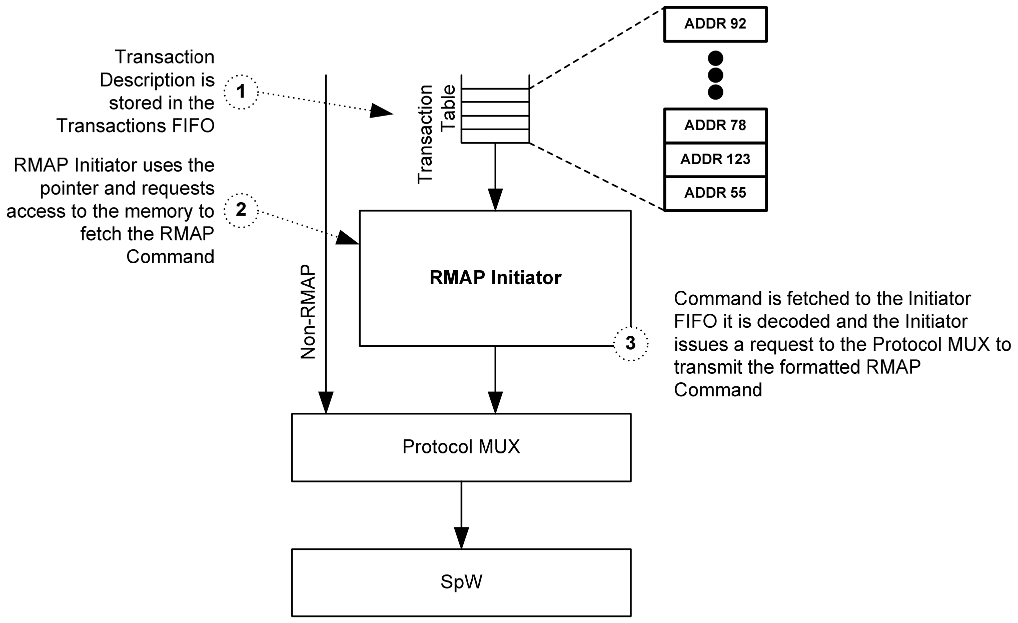

Transaction Description is stored in the Transactions FIFO
1
RMAP Initiator uses the pointer and requests access to the memory to fetch the RMAP Command
2
Transaction Table
ADDR 92
ADDR 78
ADDR 123
ADDR 55
Non-RMAP
RMAP Initiator
3
Command is fetched to the Initiator FIFO it is decoded and the Initiator issues a request to the Protocol MUX to transmit the formatted RMAP Command
Protocol MUX
SpW

Figure 30: RMAP Initiator functionality

As soon as a pointer exists in the Transaction table, the Initiator fetches it, and issues a request to the memory in order to fetch the actual command that shall be transmitted into its transmission FIFO. The Command is fetched, decoded and as soon as the RMAP packet is formatted the Initiator issues a request to the Protocol MUX block in order to gain access to the SpW layer and transmit the RMAP command.

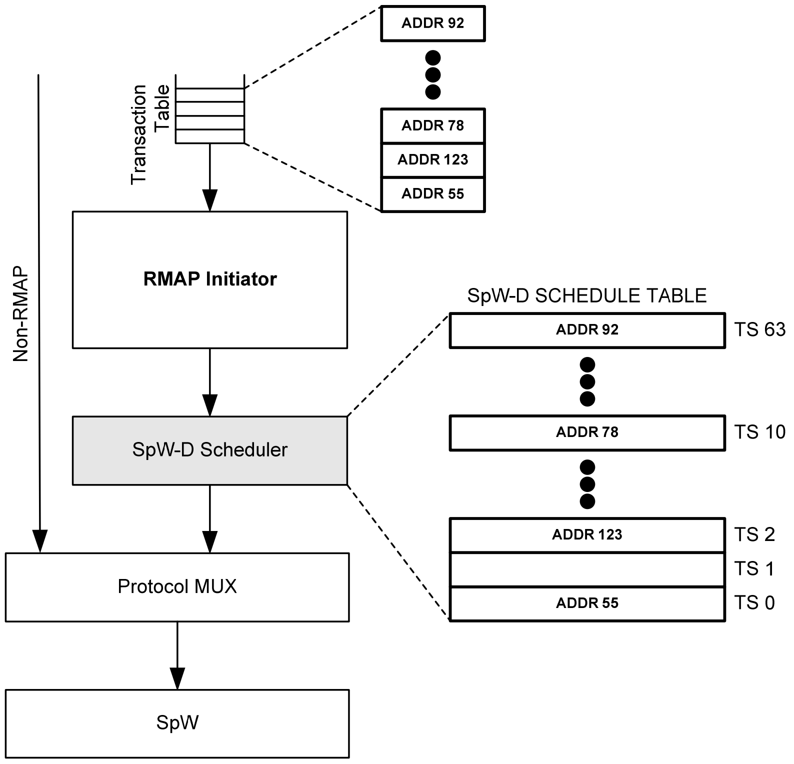


Figure 31: The synchronization problem with the SpW-D Scheduler

SpW-D adds scheduling to the RMAP Commands that shall be transmitted and also imposes a set of timings that shall be respected during transmission. A problem arises regarding the positioning of the extra scheduling layer in the Core stack.

Placing the SpW-D Scheduler between a RMAP Core and the SpW layer is demonstrated in Figure 31. This architecture is according to the SOIS stack, regarding the Resource Reservation function (**TBC**), and allows Scheduling at microsecond level.

However, when combined with an existing RMAP Core it presents a major problem, which is depicted in Figure 31. Assume that the SpW-D block implements simple scheduling and the schedule table is programmed as shown in the figure. Also the host has downloaded a set of pointers that correspond to RMAP Commands issued for Targets with Addresses 55, 123, 78 and 92 in the same order as they appear in the Schedule Table.

In case all pointers in the transaction table are downloaded during time-slot 1, then the following will happen:

- When time-slot 2 arrives the SpW-D Scheduler will inform the RMAP Initiator that a command for Target address 123 can be transmitted.
- The RMAP Initiator will check the transmission FIFO and will see that it only has a Command for Target address 55 and therefore no command will be transmitted during time-slot 2, although a corresponding entry exists in the transaction FIFO.
- The same will happen during time-slots 10 and 63.

In this scenario, all commands will be transmitted with one epoch delay. However, if the commands are not downloaded in the same order as they appear in the schedule table greater delays will occur.

In order to solve this problem there are two alternatives:

- Modification of the RMAP Core: The core can be modified in order to search the entire table for a matching entry upon the request from the SpW-D Scheduler. However, searches are costly both in terms of time and gate count they are usually slow and this approach requires the modification of an existing RMAP Core, a practice which should normally be avoided.
- Placement of the SpW-D scheduler elsewhere in the stack (shown in Figure 32). This approach is analyzed below.

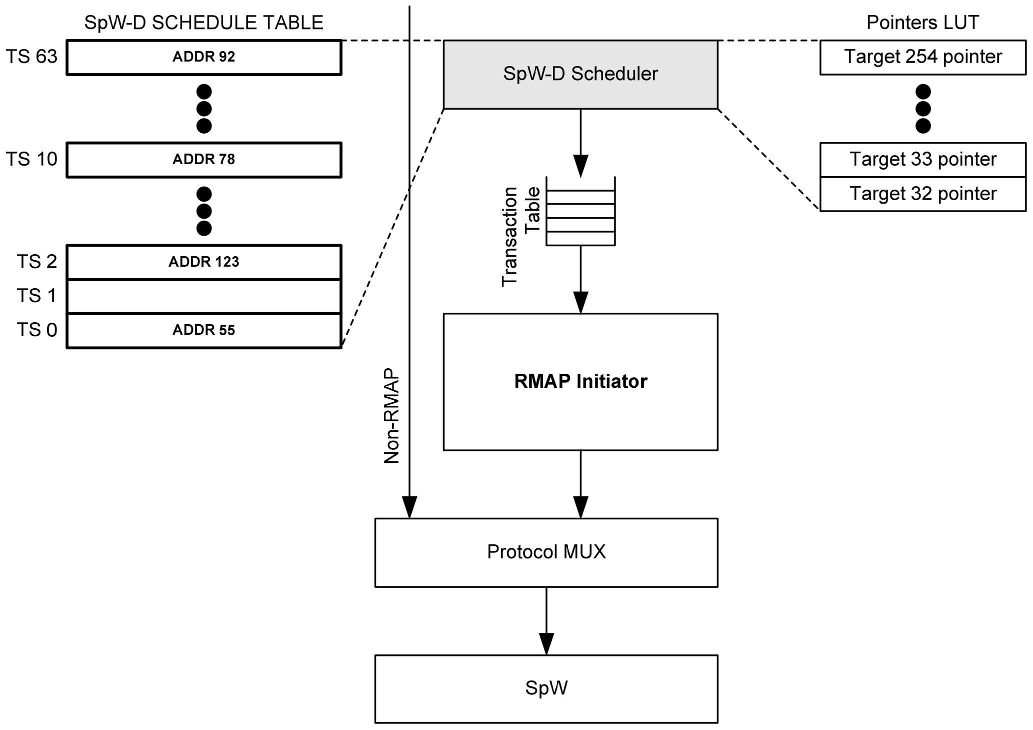


Figure 32: Architecture without synchronization problems

The SpW-D Scheduler is placed above the RMAP Core. The SpW-D Scheduler maintains two tables

- The Schedule table, which contains, as previously, the target addresses to which transactions are permitted in each timeslot.
- The Pointers LUT, which contains the pointers in the memory, where the command structures are stored. When the host attempts to write a pointer to the RMAP core, the SpW layer intercepts the

host transaction and writes the pointer to the respective address in the pointers LUT. When the appropriate time-slot arrives, the SpW-D Scheduler performs a Look-up in the LUTR and passes the pointer to the RMAP core. With this approach, RMAP transactions in the RMAP core transaction table are synchronized with the current time-slot.

5.3 SpW-D Scheduler issues

The issues with SpW-D Scheduling deal with the required memory resources for the SpW-D Scheduler, with possible extensions to support more than one transaction in a time-slot and on whether scheduling is done with or without prefetching.

Scheduler Memory Resources:

Although the architecture proposed in Figure 32 solves the problem of synchronization, it has a problem regarding the size of memory resources required. In case transactions to more than one target are desired for a particular time-slot then another byte for each Schedule Table entry is required and therefore the table size may increase prohibitively for a large number of supported targets per time-slot. Furthermore, in case an Initiator sends commands to Targets 32 and 254, the LUT shall have 223 entries whereas only 2 entries are required.

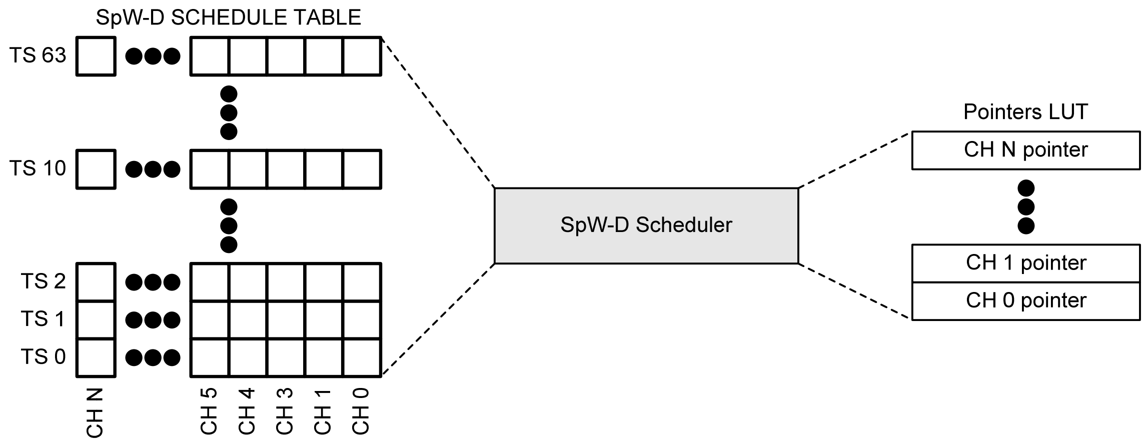


Figure 33: The "channel based" SpW-D Scheduler

In order to overcome this problem, the notion of channels is introduced. The Schedule Table consists of one bit per channel for each entry. Setting this bit means that the channel has the right to transmit in the particular time-slot. If the scheduler finds this bit set, it fetches the pointer from the LUT and passes it to the RMAP core.

This approach has the following advantages:

- There are no unused entries in the tables. Using the same example as above, in case an Initiator transmits to 2 targets only, then the schedule table is a 2x64 table and the pointers LUT has only 2 entries.
- It is possible for different epochs, to transmit commands to different targets without modifying the Schedule Table. For example, in epoch M it is possible to download a pointer for a certain target for a time-slot, whereas for the next epoch a pointer for a different target address may be downloaded.

Support for multiple transactions per time-slot:

The SpW-D scheduler architecture shown in Figure 33 is extensible regarding the maximum number of transactions supported per time-slot. Normally, the scheduler' s state machine, searches for the first match in the Schedule table, passes the pointer to the RMAP core and then returns to the IDLE state until the next time-code arrives.

In order to support more than one transactions per time-slot, the state machine can be modified and return to the IDLE state after it has found N matches and passed N pointers to the RMAP core.

Scheduling with or without prefetching:

Since SpW-D timings are strict and in order to have a more relaxed system design (regarding timings) the SpW-D scheduler may be designed so that it fetches Commands that are to be transmitted in the next Time Slot.

Whereas this approach does not guarantee that all commands will be sent on time, it offers a higher performance as can be seen with the help of Figure 35.

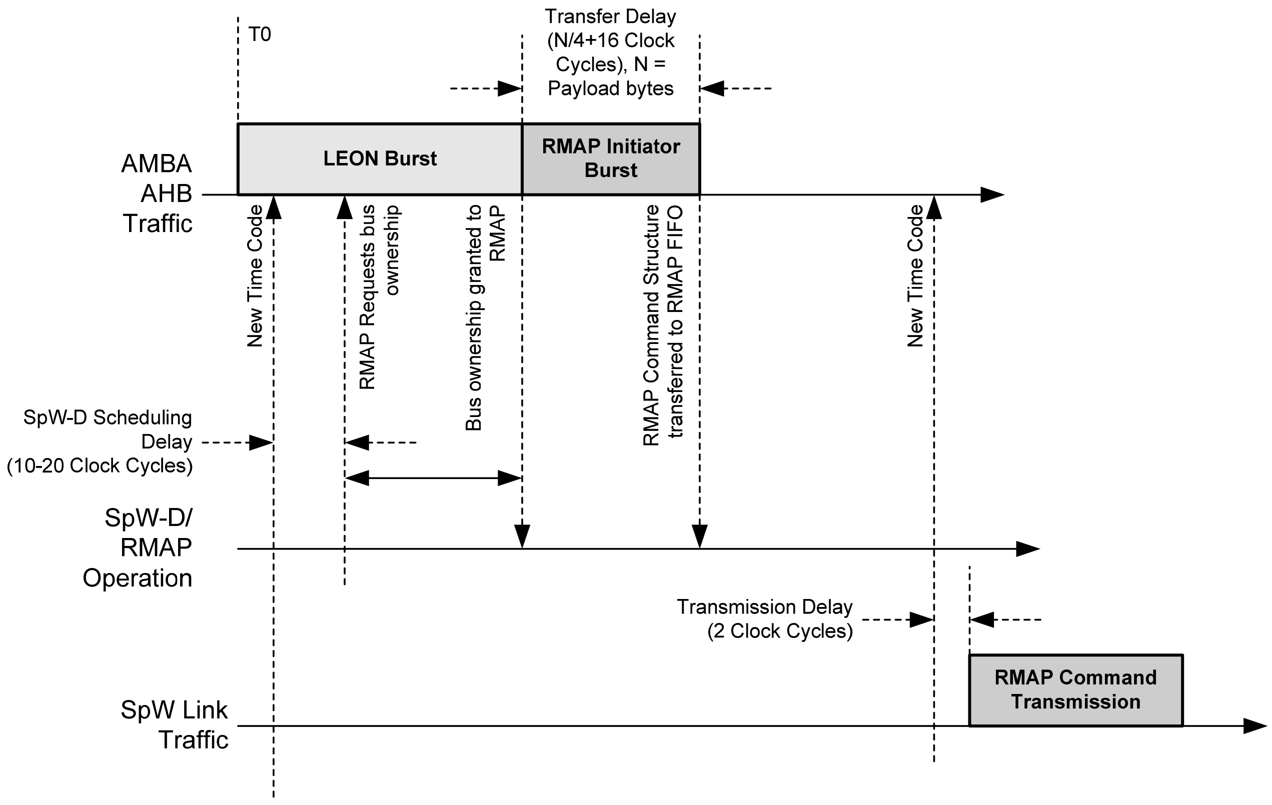

T0
Transfer Delay (N/4+16 Clock Cycles), N = Payload bytes
AMBA AHB Traffic
LEON Burst
RMAP Initiator Burst
New Time Code
RMAP Requests bus ownership
Bus ownership granted to RMAP
RMAP Command Structure transferred to RMAP FIFO
New Time Code
SpW-D Scheduling Delay (10-20 Clock Cycles)
SpW-D/ RMAP Operation
Transmission Delay (2 Clock Cycles)
RMAP Command Transmission
SpW Link Traffic

Figure 34: Scheduling with prefetching

Since commands are prefetched from memory Scheduling has been performed, RMAP encapsulation may have also be performed and therefore the packet is ready for transmission. This means that the interval from the reception of a time-code to the actual start of transmission may be as sort as 2 clock cycles, which is in the order of nanoseconds, as opposed to a couple of microseconds achieved when prefetching is not used.

This approach can shorten the time-slot and achieve a much higher data throughput at SpW-D network level. However, this approach has also some drawbacks explained below:

- Commands are prefetched and the RMAP Header and Payload are stored within a SpW-D Block memory. An open point is how an existing RMAP Core will be stimulated in order to execute the Command since now it has to be fetched from the SpW-D memory and not from the common memory. **Consequently this implies RMAP Core modification.**

- Retries are much more complex and cancel the advantage of prefetching or require more memory resources. For example, assume that a SDU is segmented and transmitted in SpW-D PDUs. If segments N and N+1 are to be transmitted in two consecutive Time Slots then the following scenario may happen. In the first Time Slot segment N is transmitted and N+1 is prefetched to the transmission buffer. In case, no Reply is received for segment N the Core should normally fetch it again from the memory and re-transmit it. However in the Buffer it has now stored segment N + 1. Therefore, in order to efficiently support retries, one has to transmit segments of the same channel every two Time Slots or store inside the SpW-D block two commands at each instance. **Consequently, Retry support increases the memory requirements**.

- Normally prefetching will start right after the reception of the time-code. However at this time, the packet to be transmitted at the current time-slot exists in the transmission buffer. **Consequently this approach increases memory requirements even more**.
- Supporting multiple commands per time-slot as future-extension, means that the SpW-D core shall store multiple packets internally. For example, support N commands per time-slot means that the block shall retain at least N+1 buffers (up to 2N depending on the implementation) . **This means that support for multiple transactions multiplies the memory requirements**.

5.4 RMAP Initiator controller issues

The SpW-D timings impose strict requirements to the way the RMAP Initiator controller fetches and executes commands from the memory and they way it handles timeouts.

Fetching and executing commands from memory:

A RMAP core is not required to operate with strict timing constraints. Upon downloading a pointer to the transaction table it may fetch an entire command (or an entire segment) from the memory, process the header information and then start transmitting, or perform the fetch and transmission in an overlapped way. Both cases are shown in Figure 35 along with the potential result for SpW-D operation.

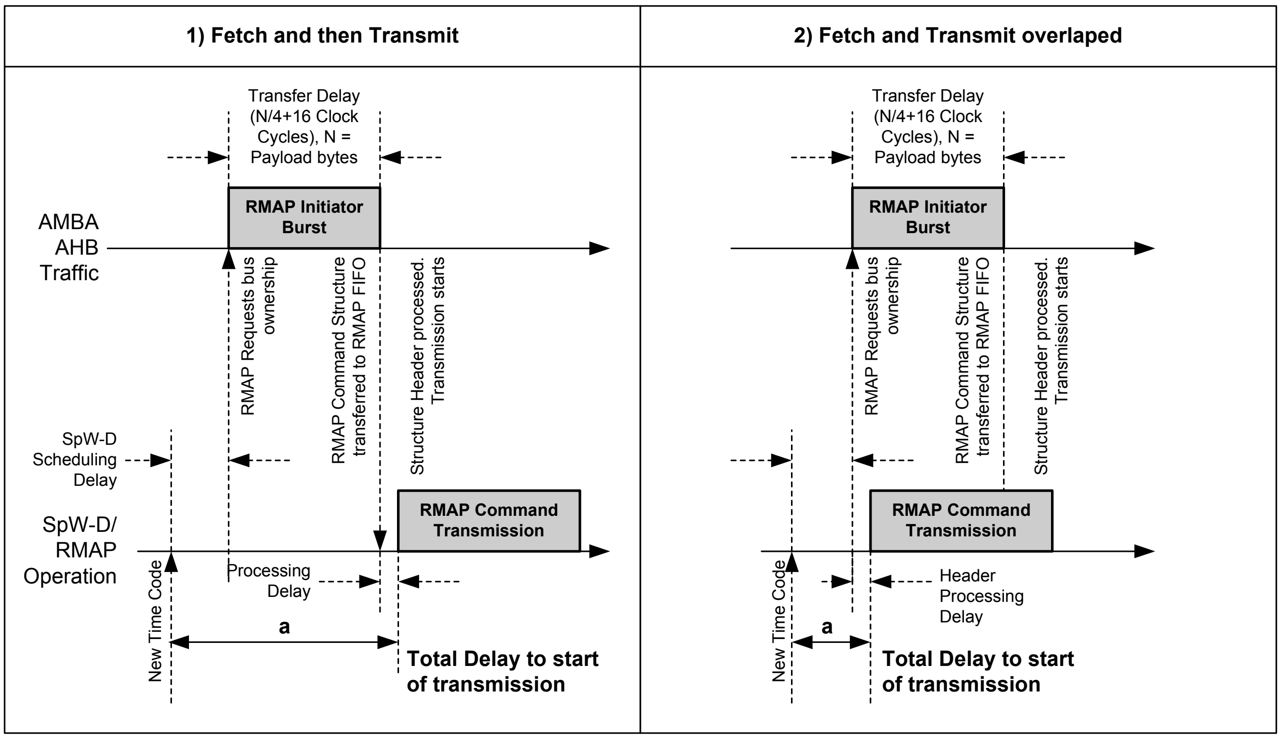

1) Fetch and then Transmit
2) Fetch and Transmit overlaped
Transfer Delay (N/4+16 Clock Cycles), N = Payload bytes
RMAP Initiator Burst
AMBA AHB Traffic
RMAP Requests bus ownership
RMAP Command Structure transferred to RMAP FIFO
Structure Header processed. Transmission starts
SpW-D Scheduling Delay
SpW-D/ RMAP Operation
RMAP Command Transmission
New Time Code
Processing Delay
Header Processing Delay
a
Total Delay to start of transmission

Figure 35: RMAP Initiator functionality and its impact during SpW-D operation

5.4.1.1 Fetch and then transmit

In the first case the Initiator controller consists of a single state machine which processes the packet stored in its FIFO after the burst from the memory has completed. Whereas such an approach may be acceptable for a RMAP core, it may introduce certain problems in SpW-D operation.

- Large packets, or low system clock frequencies may cause violation of the maximum time from the reception of the time-code to the start of transmission.

- Such an approach would require extension the timeslot duration in order to compensate for the delayed response of the core and therefore, would worsen overall bandwidth utilization in a SpW-D network.
- Command with different sizes will be transmitted with Jitter.

Assuming a maximum payload size of N bytes, with this approach transmission will start

"N/4+scheduling delay+header processing delay" clock cycles after the reception of a time-code.

Putting it into numbers, for a system operating at 50 MHz with 512 maximum packet size transmission will start after more than 3 us (assuming 15 clock cycles for scheduling and 15 clock cycles for header processing). It has to be noted that this value is for the case in which system bus is always free, which is very unlikely to happen in a real-world scenario.

5.4.1.2 Overlapped Fetch and Transmit

In the second case, the Initiator controller overlaps fetch from memory and transmission to the SpW network. In this case, the latency until the start of transmission consists of the Scheduling latency and the header processing latency. Therefore:

- Transmission starts N/4 clock cycles faster (2.56 us in the above example) resulting in better bandwidth utilization and in a more relaxed safety margin for fetching the command in case another master owns the bus.
- Transmission starts at a constant time after the reception of the time-code and Jitter is mitigated ,

Timeouts handling:

Timeouts handling is an issue that shall be taken into account early in a SpW-D system's design cycle. Most existing RMAP Cores define the time-out period in clock cycles, but this is not suitable for SpW-D operation. The reasons are:

- Since SpW-D transactions are expected to have expired by the time the next time-code is received, in order to set the timeout period the host has to know the duration of the time-slot and the exact time at which the command will be sent.
- Time-code reception however has jitter and timeouts cannot be defined with clock cycle accuracy.
- In the case of late time-codes the Command will expire.
- The exact time at which a command is transmitted from the RMAP Core cannot be defined since it relates with the time to fetch it from the memory (variable), the time to decode and prepare the RMAP packet (constant or predictable) and the time the protocol MUX assigns SpW ownership (variable).

For the aforementioned reasons the following solutions are foreseen:

- When designing a RMAP Core, it is important to include two modes of operation. A RMAP mode which will define the time period in terms of time (e.g. clock cycles, prescaler cycles etc) and a SpW-D mode which defines the timeout period in number of time-slots.
- When adapting an existing RMAP Core for SpW-D operation it is important to define the exact time (with respect to the received time-code) at which the command will be transmitted.
  - In case timeout handling cannot be disabled, this imposes traffic shaping functionality to the Protocol MUX block which allows transmission after a certain time from the beginning of the time-slot has passed.
  - In case timeout handling can be disabled (e.g. UoD Core) timeouts handling can be implemented outside the Core.

5.5 Provision for SOIS service interfaces

In this section we analyse the requirements imposed in SpW-D from the SOIS sub-network interface services [SOIS].

The three SOIS services supported by SpaceWire are the following:

- Packet Service
- Memory Access Service
- Device Discovery

In future the SOIS Test and Time Distribution services may be included.

The Device Discovery is not considered currently as a requirement because it shall use the SpaceWire-PnP which is currently being defined. The Packet Service shall use the SpaceWire-PTP interface and protocols and is also not considered in this study.

An implementation of SpW-D shall at least provide the Memory Access Service defined in [SOIS-MA].

There are five primitives used by this service:

- READ.request: will initiate a Write transfer to the Target.
- READ.indication: will indicate the end of the read transfer and will provide memory contents of the read transfer.
- WRITE.request: will initiate a Write transfer to the Target.
- READ/MODIFY/WRITE.request: will invoke an atomic Read/Modify/Write cycle at the memory of a target.
- MEMORY_ACCESS_FAILURE.indication: will inform the user of the failure during a memory access operation.

The Memory Access service shall use the underlying RMAP protocol.

If the transfer of payloads larger than the maximum RMAP payload is required, then the Memory Access Service shall use an underlying Segmentation mechanism (presented in Section 6.3).

The channel parameter of the Memory Access Service primitives may be used to specify the time-slot(s) in which the RMAP transactions are going to be initiated. The use of channels and their mapping to time-slots and Target addresses can be implementation specific and is not considered in this study.

Additionally a configuration interface (Configure.request) must be provided by the implementation of SpW-D in order to configure the schedule tables at the initiator.

6 Recommendations for updates of SpW-D draft specification

This section provides considerations for future assessment and possible implementations based on the open issues of SpW-D Draft B specification and performance issues identified in section 4.

The considerations presented in this section try to address the following issues:

Scheduling:

- If the payload of RMAP transactions for a communication (e.g. command and control) of an initiator is relatively small (e.g. 4Bytes) compared with the maximum RMAP data length (e.g. 256Bytes), there will be waste of network bandwidth since the initiator will remain idle for a relatively large percentage of these time-slots.
  - ➢ A new concept, for future assessment, to improve performance for these cases is presented in section 6.1.1 “Multi-Transaction scheduling”.
- Control loops for command and control communications (e.g. AOCS) may need to be performed at large periods (e.g. >20ms) compared to epoch interval (e.g. for a time-slot duration of 42μs, the epoch interval is equal to 64*42μs = 2.688msec).
- Control cycles may need to perform reads and writes from/to different sensors actuators with small latency between transactions. This requires allocating consecutive time-slots for the control cycle and consumes a large number of time-slots in the Schedule table.
- A potential large number of sensors and actuators may require a large number of transactions and as a result the control communication will require a large number of time-slots to complete. In case of complex topologies and a common network for command and control and data handling, if a large number of time-slots is allocated for command and control communication then a small number of time-slots will remain for data handling and vice versa.
  - ➢ A new concept, for future assessment, that improves performance in these cases is presented in section 6.1.3 “Concurrent scheduling with different scheduling tables in different epochs”.
- Waste of network resources in the presence of asynchronous traffic, caused by the reserved slots that are idle when initiators have no asynchronous traffic to transmit.
  - ➢ A new concept, for future assessment, to improve performance for these cases is presented in section 6.1.2 "Use of dedicated time-slots for asynchronous traffic”.

RMAP transactions:

- If initiator needs to transfer data from one target to another (e.g. OBC commands a transfer from instrument to mass memory) the same payload must be transferred from target to initiator and then from initiator to target.
  - ➢ A possible solution for direct target to target transfer using RMAP that will improve performance while not requiring complex initiator-target node implementations, is presented in section 6.2 “RMAP transactions”.

Segmentation and Retry:

- Segmentation and Retry have not been defined in SpW-D Draft B specification [AD, 2].
  - ➢ Sections 6.3 and 6.4 present possible alternatives for the Segmentation and Retry mechanism. These alternatives are presented for future study and assessment.

6.1 Scheduling

Based on the open issues regarding SpW-D scheduling, addressed in section 4.1, the following concepts for potential future updates are presented and analysed in more detail in the following sections.

6.1.1 Multi-Transaction scheduling

Multi-transaction scheduling in not defined in SpW-D Draft B but is possible to be implemented to improve network efficiency further in cases where the length of the payload of different transactions diverges significantly and small amounts of data have to be transferred between an initiator and a number of different target nodes, by permitting an initiator to initiate more than one transaction in the same time-slot.

In order to support multiple RMAP transactions from the same initiator at the same time-slot a modification is required in the scheduling scheme.

In any multi-transaction time-slot the following conditions shall apply:

a) The initiator must be able to initiate a configurable number of transactions in the same time-slot.

b) Schedule table, timing configurations and RMAP transactions payload must guarantee that all RMAP transactions shall be finished within the time-slot.

c) Schedule table must ensure that no conflict of network resources occurs over the duration of the time-slots (i.e. paths from other initiators to their targets do not use the same SpW link with the paths used for the transactions in this time-slot).

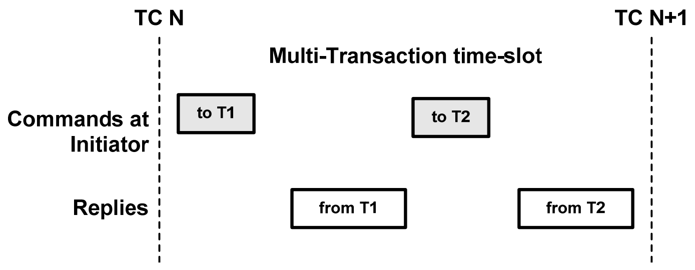


Figure 36: Time-slot with multiple RMAP transactions

In this case the schedule table may be configured to have a number of time-slots with multiple transactions and a number of time-slots with only one allowed transaction (as in concurrent or simple schedule).

As presented in the previous figure, the time-slot is used for two transactions, the Initiator initiates a transaction to target T1 and when finished initiates a second transaction to target T2. Both transactions must finish within the time-slot duration and the schedule table shall ensure that there is no conflict in transactions with targets T1, T2 with the transactions of other initiators as in concurrent schedule. The transactions to target T1 and T2 can use paths with common SpW links since they are performed in different time intervals.

This solution is easy to implement, allows RMAP transactions to be performed with the same target and no conflicts occur in the network, but does not increase the throughput significantly.

Another possible implementation is to permit the Initiator to issue a next RMAP command before receiving the previous RMAP reply, as presented in the following figure.

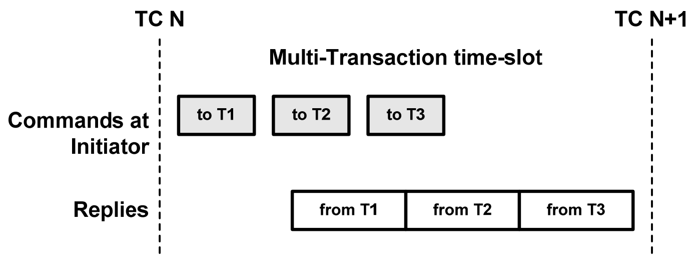


Figure 37: Time-slot with multiple RMAP transactions

In this case more RMAP transactions can be performed in the same time-slot. The overall throughput is increased, since conflicts are allowed during the time-slot and the total time for RMAP transaction is reduced (the network can operate in full-duplex), but this alternative is more difficult to be implemented and the same target can not be used for consecutive RMAP transactions.

Performance evaluation for a selected scenario

Assume that mutli-transaction is performed using the first solution (where the initiator must wait for the RMAP reply before issuing the next RMAP command). According to the simulation results of section 3.3.2.6 ("Minimum Time-slot interval and maximum data-rate vs data length for 200Mbps SpW link speed") for a 4Bytes maximum payload the time-slot duration is equal to 28μs. This time is equal to the total time required to complete the RMAP transaction with the addition of a safe margin of 2μs and rounded to a 1μs resolution. Consequently the total time to complete a RMAP transaction with 4Bytes payload is less or equal to 26μs. If we assume that the initiator is capable to initiate the next transaction with a delay equal to Ta = 5μs (i.e. Ta is the delay defined between the reception of a time-slot and the start of transmission of the command) after the reception of the previous reply then the time-slot duration required for a number of N transactions is equal to Nx26+2μs. For N=2 the time-slot duration must be 54μs.

Based on the same simulation results the total time for a transaction of 256Bytes is 40μs and the corresponding time-slot with a safe margin is 42μs. According to these numerical results multi-transaction scheduling with 4Bytes payload can not be performed if 256 Bytes is the maximum RMAP payload size and time-slots of 42μs unless the HW latencies and constraints are reduced. In case that the maximum payload is set to 512 bytes, then the time-slot will be 57μs. In this case, it is possible to utilize multi-transaction scheduling with 2 RMAP transactions per time-slot.

Assuming that in an epoch interval there are 32 RMAP transactions for data handling with the maximum payload of 512 bytes and 32 RMAP transactions for command and control with a small payload of 4Bytes, the overall data rate will be 8*(32*512Bytes + 32*4Bytes) / 64*52μs = **39.69Mbps**

In the same case if we assume that we apply multi-transaction scheduling the number of RMAP transactions with small payload will be 16 (two transactions in every time-slot). As a result, more transactions can be used for data handling (48 transactions). In an epoch interval there will be 48 RMAP transactions for data handling with the maximum payload of 512 bytes and 16 RMAP transactions with a small payload of 4Bytes. The overall data rate will be 8*(48*512Bytes + 16*4Bytes) / 64*52μs = **59.38Mbps and the latency for the completion of the control communication will be halved** (16 time-slots are required instead of 32).

By applying Multi-transaction scheduling in the previous example scenario the improvement of data rate will be (59-39) Mbps/59Mbps = 33%.

Additionally the time to complete the transactions with the small RMAP payload will be halved.

Conclusions

This method of scheduling has the following advantages and disadvantages:

Advantages:

- Reduced latency in transactions with small RMAP payload.
- Improves data rate when many transaction with small payload are scheduled in the epoch interval.

Disadvantages:

- Implementation of the SpW-D scheduler is more complex and requires increased memory resources depending on how it will be implemented.
- Payload length checking shall be performed in these RMAP transactions (either from Host SW or Scheduler Core HW) in order to not violate the time-slot boundary.

6.1.2 Use of dedicated time-slots for asynchronous traffic

As already analyzed in 4.1, a problem in networks that use TDMA based access scheme is the waste of network resources in the presence of asynchronous traffic, caused by idle time-slots when initiators have no traffic to transmit.

This problem has been mitigated in some TDMA based protocols by:

- using dynamic scheduling or time-slot slot reservation (e.g. in GSM)
- dividing the time-slots in different periods for isochronous and asynchronous communications (e.g. in FlexRay).

The first method requires the following:

- A master node is required in the network to dynamically select the optimum schedule table for every initiator and define in which time-slots each initiator can initiate transactions with specific targets.
- Initiators must periodically (in each epoch or for a number of epochs) publish to the master node their communication requirements (i.e. number of time-slots and targets to initiate transactions with) and shall retrieve from the master node the schedule tables to use.

This method is very complex and difficult to be applied in the SpW-D and is not considered, due to the following reasons:

- The master node must perform a Schedulability analysis for every initiator in different time-intervals which is a very complex task in case of SpW-D due to the complex network topology, the requirement that no shared SpW link shall exist in the paths used for transactions in every time-slot, the different scheduling schemes that may be utilised by the initiators (e.g. use of multi-slot schedule), the dynamic changes in topology by redundancy management due to failures, etc
- Since different tasks with different communication requirements are executing in an initiator, it will be difficult for a network task in the initiator to identify the exact time-slots and targets that transactions shall be performed in different time periods and publish this information to the master node. This would require the interaction between different tasks and would increase the onboard SW complexity.
- In most networks the data exchange with the master node (or a core network entity) for dynamic reservation is performed in broadcast channels. The lack of broadcast in SpW-D will require that each initiator shall perform a number of write and read transactions with the master node publishing communication requirements and retrieving the schedule table. Taking into account that this information may be large in size, a significant number of time-slots will be wasted for this communication reducing network efficiency.

The second method can be applied in the SpW-D case by using specific time-slot(s) for asynchronous traffic (as in FlexRay, please note that although FlexRay is a bus network some of its concepts can also be applied to switched network topologies).

With this method the following rules apply:

- An Epoch is divided into different segments:
    - **Synchronous**: time-slots used for scheduled transactions. SpW-D scheduling is used.
    - **Asynchronous**: allocated for asynchronous transactions from a number of initiators. Initiators operate according to the RMAP standard; Time-Codes are transmitted in the network. The initiators use the schedule table to determine to which targets they are allowed to send commands, conflicts are allowed and the time-slot boundaries may be violated. During this segment the network operates as a SpW network which propagates RMAP packets of limited length.
    - **Guard**: one time-slot with configurable duration for receiving any pending replies from the asynchronous transactions (in worst case shall have a duration of N x Total time for processing and transmission of an RMAP reply, where N is the number of initiators). Initiators must not transmit in this period (analysed below).
- The Asynchronous segment consists of a number of time-slots.

- Any initiator may initiate a transaction towards any target during asynchronous segment.

The different segments shall be configurable in the network level and shall be selected based on the communications requirements of the network. Extreme values of this configuration is to have the synchronous segment occupy the whole epoch, where the network operates in SpW-D mode, or the asynchronous segment occupy the whole epoch where the network operates in SpW RMAP mode.

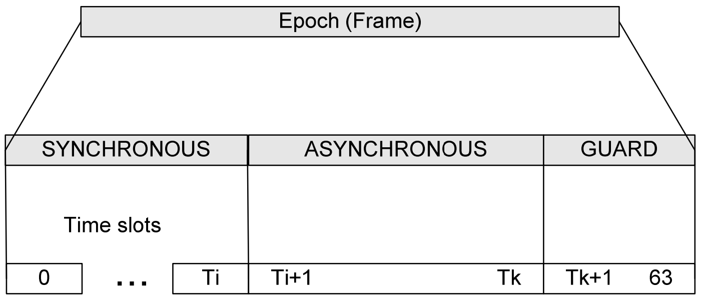


Figure 38: Dedicated Time-slots for asynchronous traffic

This technique shall be employed carefully since, during the Asynchronous segment, conflicts are possible. For this reason the duration of the asynchronous time-slot(s) shall be carefully selected based on network topology and asynchronous traffic requirements in order to guarantee that N RMAP transactions can be serviced within the asynchronous segment duration.

In worst case if all initiators try to perform a RMAP transaction at the beginning of the segment to the same Target there will be a conflict and blocking of initiators. In this case:

- Transactions will finish after N x Total time for RMAP transaction.
- The minimum duration for the Asynchronous segment shall be at least equal to N x Total time for RMAP transaction, where N is the number of initiators in the network that may use the Asynchronous segment.
- If the Asynchronous segment duration is not carefully chosen, Blocked Asynchronous traffic may violate the Asynchronous segment and interfere with Synchronous traffic of the next epoch.

In order to solve the last issue the Guard segment has been added. In this segment the transmission of RMAP commands is not allowed. This guard segment shall guarantee that all replies from all unfinished and pending RMAP transactions are transmitted within its duration. This will guarantee that there will be no network resource conflict with the scheduled RMAP transactions in the next time-slot used for synchronous traffic. The minimum duration of the guard segment shall be at least equal to N x Total time for processing and transmission of an RMAP reply.

In the asynchronous segment the initiator must operate as a standard RMAP initiator, timers shall be configured based on the segment durations (e.g. watchdog may be set to the end of asynchronous segment). At the guard segment, transmission of commands shall be disabled and the time-code watchdog timer shall be set to the end of the guard segment.

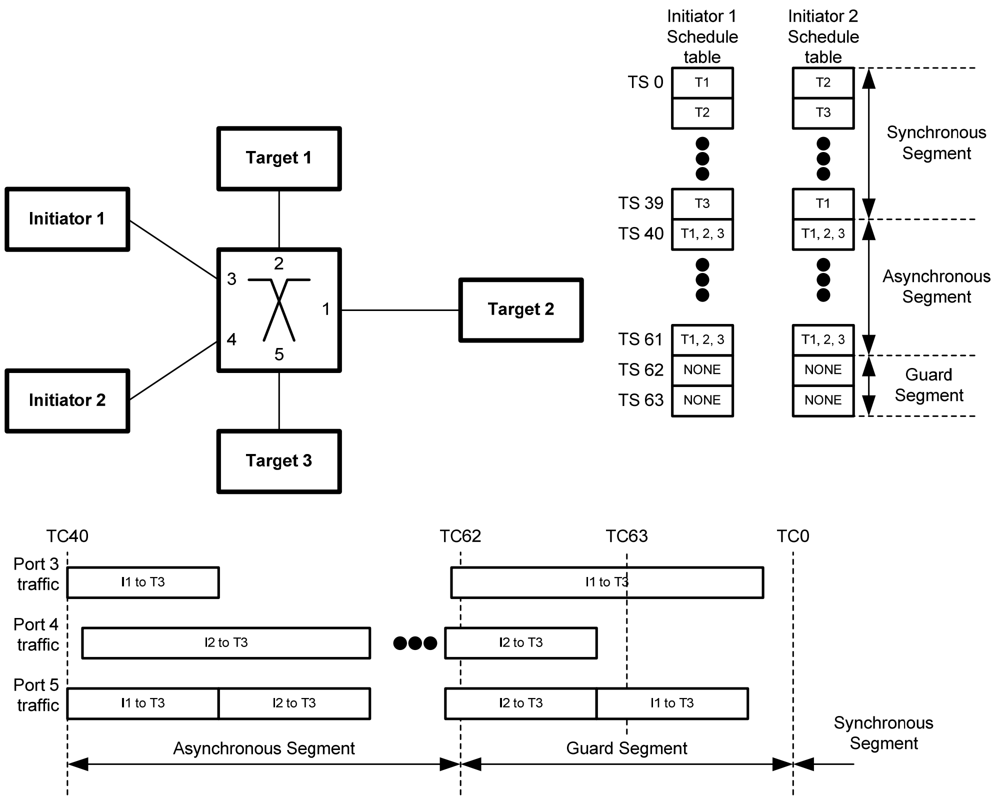


Figure 39: Example of Asynchronous Segment usage

An example implementing this approach is shown in Figure 39. The first 40 time-slots of each epoch are reserved for the Synchronous Segment. Asynchronous Segment consists of time-slots 40 to 61 and Guard segment occupies the last two time-slots.

As shown in the example:

- Initiators 1 & 2 schedule tables do not have any conflicting paths for the Synchronous Segment.
- They may have conflicting paths for the Asynchronous Segment. However, for the asynchronous segment, commands may be sent to any target by both initiators. A conflict in accessing a target (or a common path) will be handled by the SpW router, which will block one of the incoming packets.
- At the start of the Asynchronous Segment, Initiator 1 and 2 try to access Target 3. Initiator's 1 command will pass through the router, whereas initiator's 2 will be blocked until the first command has reached target 3. Afterwards, initiator's 2 command will be routed to target 3.

Figure 39 also presents the worst case for the calculation of the number of time-slots required for the Guard Segment. It is possible that more than one initiator transmits commands that traverse a common path or access the same target at the end of the Asynchronous Segment. Avoiding such cases is crucial, thus the number of time-slots the Guard Segment occupies must equal the number of initiators using a common resource. This means that for N initiators accessing the same target (or traversing a common path) during the last Asynchronous Segment time-slot, the Guard Segment shall be set to N time-slots.

Conclusions

Concluding, this solution will provide the following benefits:

- ➢ Waste of bandwidth is reduced in the presence of high asynchronous traffic.
- ➢ The network behaves as a hybrid of SpW-D and standard SpW RMAP network (in different time-periods) combining benefits from both (i.e. determinism from SpW-D and improved performance and efficiency from standard SpW - RMAP).

This solution has the following drawbacks:

- ➢ Guard segment may reduce network efficiency since some time-slot(s) are wasted.
- ➢ If the guard and asynchronous segments use a large number of time-slots then the number of available time-slots for synchronous traffic is reduced.

6.1.3 Concurrent scheduling with different scheduling tables in different epochs

Scheduling large data transfers and command and control communications that require many time-slots to complete and have relatively high period compared to the epoch interval can be a difficult task in a common network.

Additionally the relatively small number of time-slots (64) is a scarce resource and may become a problem in scheduling complex network topologies with many initiators and many transactions between initiator and target nodes.

A potential solution to overcome these problems is to use a signalling mechanism for synchronising the initiators at the different epochs in order to use different scheduling tables or schedule different transactions in each time-slot and avoid network conflicts. This is a common practise in many networks that use TDMA based access schemes, where they use a hierarchy of TDMA frames composed by a number of time-slots and multi-frames or super-frames composed by a number of TDMA frames.

Such solution is possible to be implemented in SpW-D by utilizing the control bits of the time-codes (2bits). In such a case 4 Frames can be defined by the control bits 7 – 6, and each Frame can contain 64 time-slots which are defined by bits 5-0 (as in SpW-D Draft B). As a result, the super-frame will be composed by 4 Frames and will be repeated every 4 epochs, as presented in the following table and figure. In this case each frame can be associated with a different schedule table, although this is not a requirement and specific initators can use the same schedule table in all Frames.

time-code bits[7-6]	time-code bits[5-0]	Frame	Time-slot
00	00	0	0
,,,	,,,	,,,	,,,
00	3F	0	63
01	0	1	0
,,,	,,,	,,,	,,,
01	3F	1	63
10	00	2	0
,,,	,,,	,,,	,,,
10	3F	2	63
11	00	3	0
,,,	,,,	,,,	,,,
11	3F	3	63

Table 11: Time-code values and their relation with Frames and Time-slots

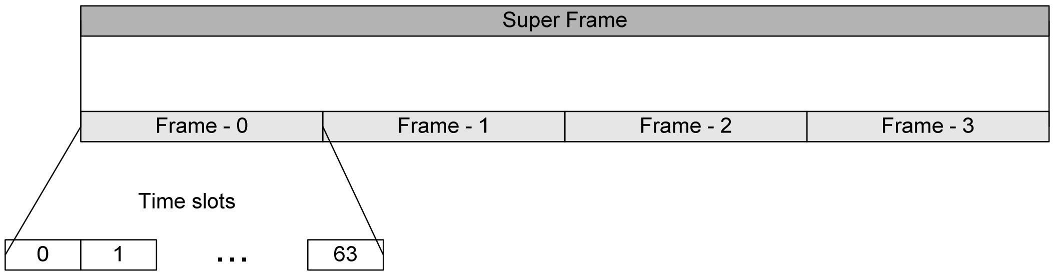


Figure 40: Super Frames, Frames and Time-slots

If the Frames are repeated periodically as in the previous figure then the effect on the network will be the same as extending the number of time-slots from 64 to 256.

Alternatively the control bits of time-codes can be used in order to repeat a specific Frame in predefined intervals. If we assume that 1 Control bit is used by the Time-Code then there can be two different Frames in the network, Frame-0 and Frame-1. In this case an initiator can use two schedule tables in different time-periods (based on epoch interval) that are switched when the Time-code Control bit changes. Another initiator may use only one schedule table that can be the same in all Frames.

As an example Frames and Time-slots can be as in the following table and figure.

time-code bit [6]	time-code bits[5-0]	Frame	Time-slot
0	00	0	0
,,,	,,,	,,,	,,,
0	3F	0	63
1	0	1	0
,,,	,,,	,,,	,,,
1	3F	1	63
1	0	1	0
,,,	,,,	,,,	,,,
1	3F	1	63
...	...	...	...
,,,	...	...	...
...	...	...	...
0	00	0	0
,,,	,,,	,,,	,,,
0	3F	0	63

Table 12: Time-code values and their relation with Frames and Time-slots

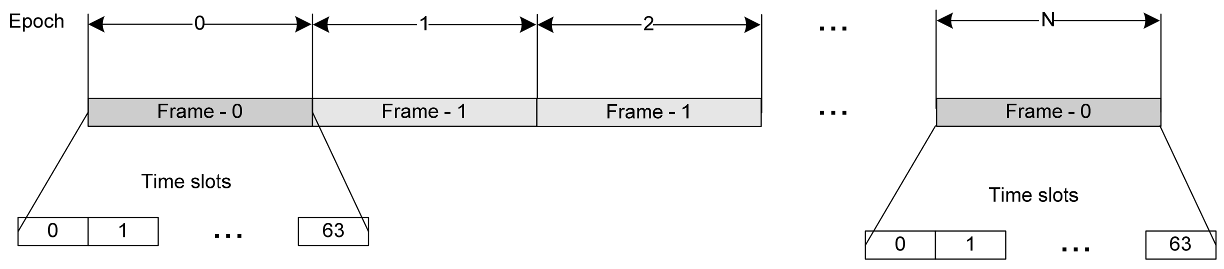


Figure 41: Frames repeated in defined intervals

In this case more flexible scheduling can be performed in both data handling and command and control communications which require large repetition period compared to epoch interval and low latency.

A more general case to apply this scheduling scheme is the following:

- Time-code bits [5-0] shall always increase and roll over from 63 to 0.
- Time-code bits [7-6] shall change based on a predefined pattern.
- The Scheduler shall select time-slots and shall schedule RMAP transactions from different Frames (or schedule tables) based on the 8-bit Time-code value.

As a result, a specific number of time-slots (and not the whole Frame) can be repeated and the corresponding RMAP transactions can be scheduled for selected intervals, as presented in the following figure.

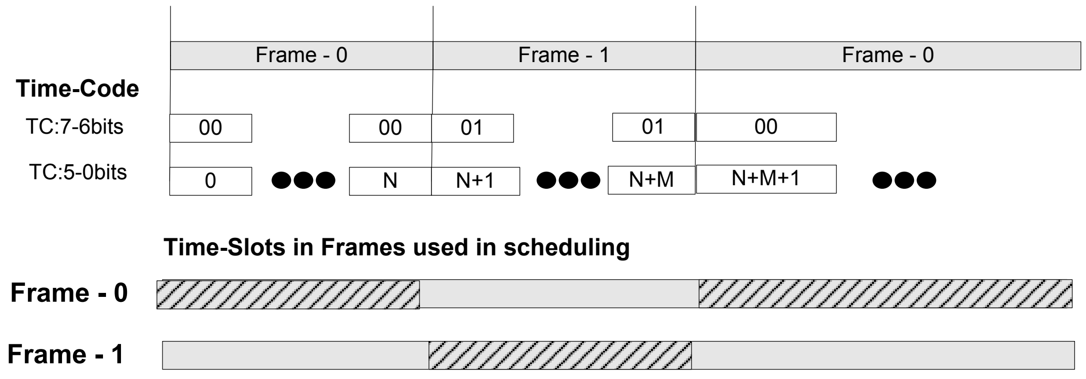


Figure 42: Scheduling based on time-code value

Performance evaluation for a selected scenario

Note: This is an example topology for illustration purposes and does not provide an optimum or realistic network topology.

Assuming the following example topology and scenario where concurrent scheduling is used and CDMU initiator needs to perform command and control (C&C) communication with a period of 10Hz (100msec), for this communication 32 RMAP transactions are required for reading/writing data from/to PDHU and the Instrument. Additionally, if the latency between these transactions must be small, then 32 consecutive time-slots must be allocated for these transactions.

PDHU Initiator needs to perform data handling (DH) with the Instrument and there are common SpW links in the paths from these initiators to their targets.

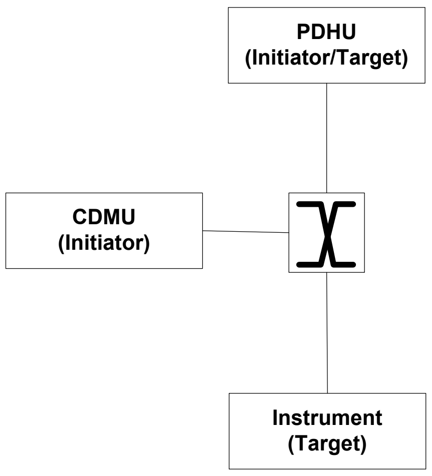


Figure 43: Example network topology

As a result the scheduling tables for the CDMU and PDHU initiators, as well as the communication activity of the Instrument can be as following, where we assume that N+K = 32 time-slots.

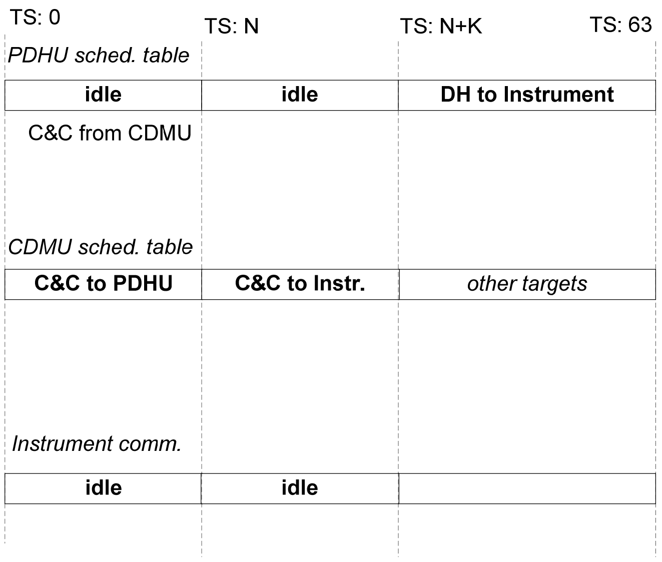


Figure 44: Example Concurrent schedule tables

In this case PDHU can not access the Instrument in time-slots 0 to 32 (until time-slot N+K) in order to avoid conflict with the command and control communications performed by CDMU every 100msec. This is a great waste of time-slots and the overall data rate of data handling performed by PDHU will be reduced by 50%.

If less time-slots are used for CDMU, as an example 8, then the overall data rate of DH communications from PDHU will be reduced by 12.5%. But in this case CDMU will require 4 epochs to perform the command and control communication. If the time-slot interval is around 39μs the epoch interval will be 2.5msec and CDMU will require 10msec, while in the first case it would require only 39μs * 32 = 1.24msec. In this case the time between the first and last RMAP transaction for C&C is increased by around 4 epochs, or 10msec.

If the proposed concept is applied in this case two different concurrent schedule tables can be used. The first for command and control communications, scheduled in Frame 0, and the second for data handling communications, scheduled in Frame 1.

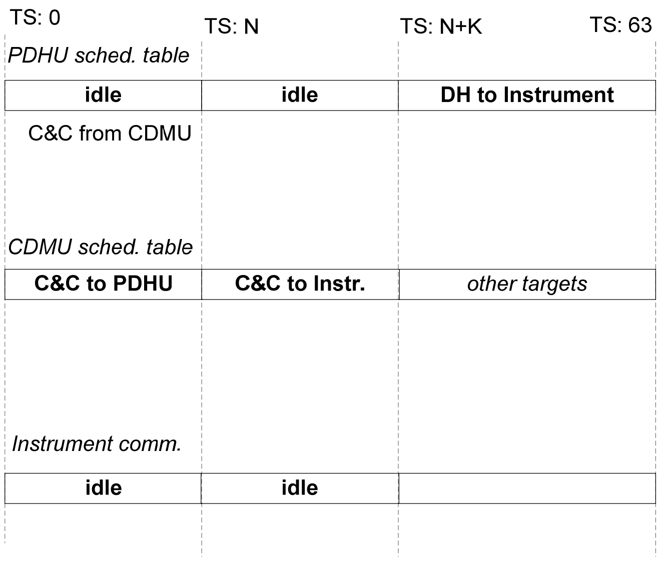


Figure 45: Concurrent schedule tables for Frame 0

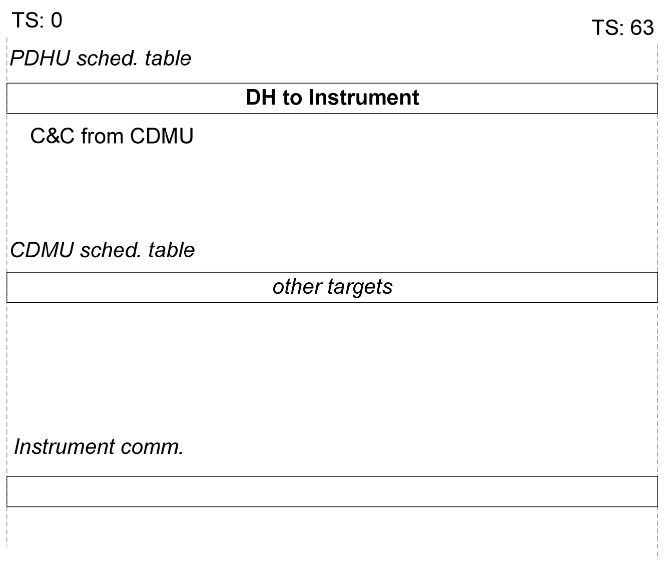


Figure 46: Example Concurrent schedule tables for Frame 1

If Frame 0 is scheduled every 100msec (40 epochs) which is the period required by the control loop and Frame 1 is scheduled in all other epochs, the Frames will be repeated as in the following figure.

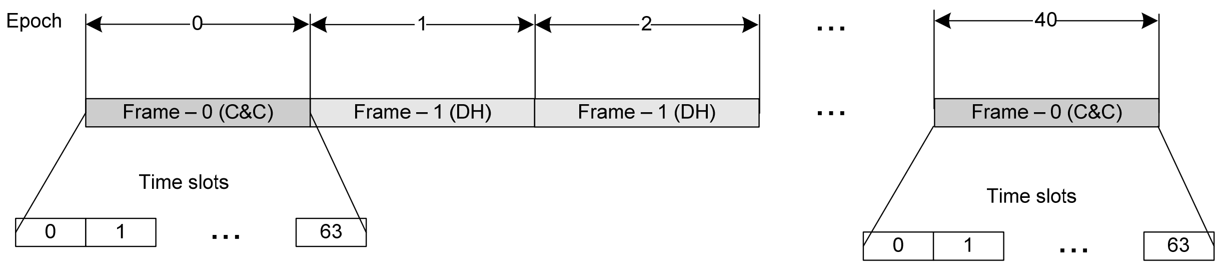


Figure 47: Frames scheduled for Command and Control (C&C) and Data Handling communications (DH)

In this case all command and control RMAP transactions are performed within the same epoch resulting in the minimum latency between first and last transaction, around 1.24msec and the data rate of data handling communications from PDHU is the optimum since only 32 time-slots are wasted every 40 epochs. As a result, **in this case the reduction in the overall data rate for PDHU will be around 1% while the latency imposed in C&C RMAP transactions for CDMU will be the minimum.**

A potential issue in this case is that the repetition of Frame 0 may produce periodic Jitter in the communications performed in Frame 1.

Conclusions

This solution will provide the following benefits:

- Better scheduling capability improving network efficiency.
- Increased number of available time-slots providing flexibility in scheduling.
- Possibility to schedule isochronous RMAP transactions with a higher periodicity than the epoch interval.

This solution has the following drawbacks:

- If the schedule tables (up to 4) or the 256 time-slots per schedule table are maintained in the Core the memory requirements in the initiator will be multiplied by up to N.
- Time-master modification is required in order to increment the control bits of time-code in every epoch or with a network configurable predefined pattern.

- There is an issue regarding the compatibility of such solution with existing routers, and time-codes may not propagate correctly to the network by some routes (according to SpW specification [AD,4] the two most significant bits of time-codes shall contain control flags that are distributed isochronously with the time-code).
- This technique reserves available broadcast SpW characters and does not allow their use for other purposes (e.g. distributed interrupts).
- Jitter may be imposed in data handling applications due to the repetition of the schedule table for command and control communications.

6.1.4 Comparison of scheduling alternatives

Applicability of the scheduling schemes

Schedulability analysis in the SpW-D network is a difficult task especially in the case of complex network topologies and mixed traffic profiles (e.g. periodic, asynchronous, control traffic etc).

Based on network requirements different trade-offs can be taken into account and different scheduling methods or functionalities can be applied in the SpW-D network.

Table 13 tries to correlate the different requirements regarding network topologies, traffic characteristics and potential scheduling functionalities of SpW-D that may be applied in order to increase network performance.

This is only an example to illustrate the capabilities and the potential flexibility of SpW-D network, the result of the selected methods and features in a real network will be the result of a schedulability and trade-off analysis and depends in many factors including mission requirements, exact topology, real-time requirements, exact traffic profiles, etc.

	Simple Schedule	Concurent Schedule	Multi-Slot Schedule	Multi-Transaction Schedule	Concurrent scheduling with different scheduling tables in different epochs	Use of dedicated time-slots for asynchronous traffic
Simple topology or small number of initiators						
CC or DH with small payloads	x					
DH with large payloads	x		x			
DH with high asychronous traffic	x		x			
Common network for CC and DH, DH with small payload	x					
Common network for CC and DH, DH with large payload	x		x	x		
Common network for CC and DH, DH with high asychronous traffic	x		x	x		
More complex topologies or large number of initiators						
CC or DH with small payloads		x				
DH with large payloads		x	x			
DH with high asychronous traffic		x	x			x
Common network for CC and DH, DH with small payload		x				
Common network for CC and DH, DH with large payload		x	x	x		
Common network for CC and DH, DH with high asychronous traffic		x	x	x		x
Large number of nodes with many transactions between nodes						
CC or DH with small payloads					x	
DH with large payloads			x		x	
DH with high asychronous traffic			x		x	x
Common network for CC and DH, DH with small payload					x	
Common network for CC and DH, DH with large payload			x	x	x	
Common network for CC and DH, DH with high asychronous traffic			x	x	x	x

*CC: Command and Control communications

*DH: Data Handling communications

Table 13: Existing and proposed scheduling techniques and their applicability to various network topologies and communication profiles

The above table provides three general categories of network topologies and combination of traffic profiles and scheduling techniques of SpW-D (either defined in SpW-D Draft B or proposed in this document) that can be applied in order to improve performance and efficiency. Depending on each case a combination of scheduling techniques can be applied.

The first category is “Simple topology or small number of initiators”. This category assumes network topologies with one initiator or more than one with low bandwidth requirements for the communication.

The second category is "More complex topologies or large number of initiators". This category assumes network topologies with more than one initiator and relatively high bandwidth requirements that can not be fulfilled by a simple schedule.

The third category is "large number of nodes with many transactions between nodes". This category assumes very complex topologies with many initiator and target nodes and many different transactions between initiators and targets. In this case the use of different schedule tables in different epochs will increase the available time-slots and schedulability analysis will provide schedule tables that make more efficient use of network resources.

The comparison of the examined scheduling alternatives is summarized in Table 14.

	Throughput	**Latency**	**Time-code availability**	**Implementation issues**	**Other issues**
SpW-D draft	Poor, network operates in half-duplex mode			Easiest to implement	
Multi-transaction	Increases in the presence of small-payload transactions	Decreased for small-payload transactions	Saves time-codes used for small payload transactions	Slight modifications to the SpW-D Scheduler. Memory resources increase if different number of transactions shall be supported per multi-transaction time-slot	
Concurrent scheduling with different scheduling tables in different epochs	Same as SpW-D Draft spec. Can be increased in complex network topologies due to more flexible scheduling.		Best	Increases the SpW-D Scheduler required memory resources. Slight modifications to the time codes-master	Capability to schedule transactions with high periodicity but may insert periodic Jitter to other transactions
Asynchronous Segment	Increases in the presence of asynchronous traffic	Increased for asynchronous traffic	Decreases proportionally with the duration of the Asynchronous Segment and the number of conflicting initiators (the Guard Segment increases)	Slight modifications on the SpW-D Scheduler. Has as a prerequisite the multi-transaction scheduling	

Table 14: Proposed SpW-D scheduling alternatives

6.2 RMAP transactions

The current SpW-D draft defines only a single QoS service category which in many cases doubles the required bandwidth for a single transfer.

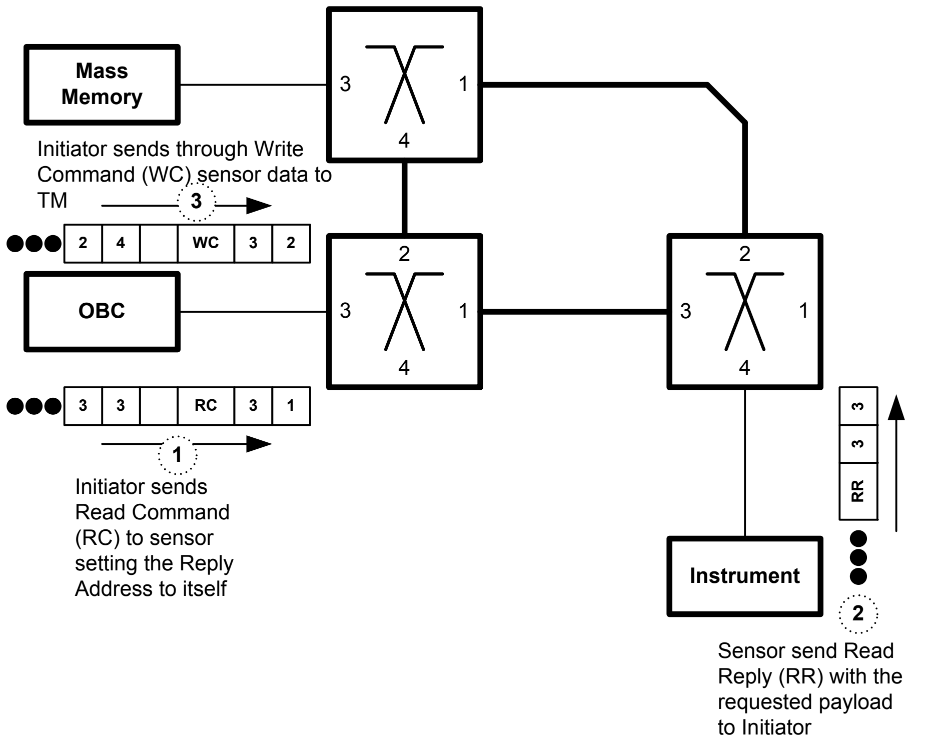

Mass Memory
Initiator sends through Write Command (WC) sensor data to TM
3
2 4 WC 3 2
OBC
3 3 RC 3 1
1
Initiator sends Read Command (RC) to sensor setting the Reply Address to itself
3 1 4
2 3 1 4
2 3 1 4
RR 3 3
Instrument
2
Sensor send Read Reply (RR) with the requested payload to Initiator

Figure 48: Data delivery to a node according to the current SpW-D specification

Assume that the OBC shall read data from the instrument and deliver them to the Mass Memory unit. According to the SpW-D specification, since all transactions shall be acknowledged, thus the OBC shall issue a Read Command to the instrument, wait for the reply and then copy the instrument payload in another RMAP Write command packet for final delivery to the Mass Memory unit. As shown in the figure, the OBC initiates a transaction and sets the Reply Address Field of the RMAP header to 3, 3, which is the path to itself. The instrument replies with a RMAP reply packet, with its SpW address field to 3, 3 and the reply packet containing the sampled data is delivered to the OBC.

Although this approach ensures that the sample is delivered to the Mass Memory unit, it doubles the bandwidth required for a single transfer. If a different QoS category is desired then it is possible to exploit the capabilities offered by the functionality of the RMAP protocol in order to use the bandwidth more efficiently in case, where acknowledged transactions are not mandatory. Such a case is presented in Figure 49.

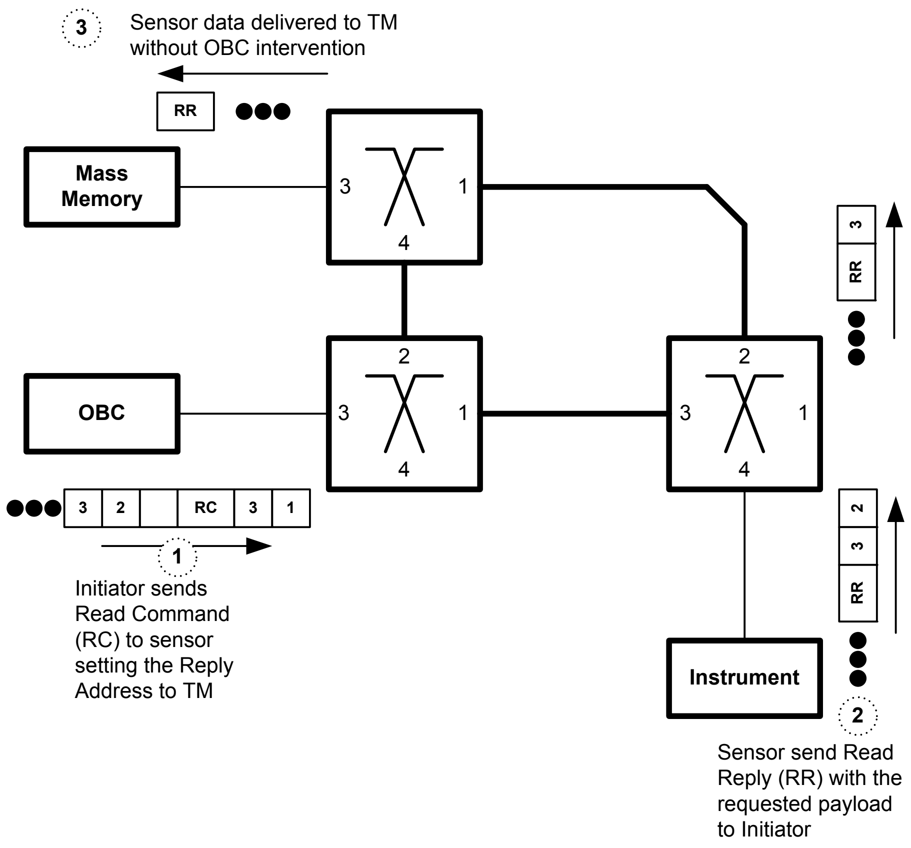

3
Sensor data delivered to TM without OBC intervention
RR
Mass Memory
3
1
4
2
OBC
3
1
4
2
3
1
4
3
2
RC
3
1
1
Initiator sends Read Command (RC) to sensor setting the Reply Address to TM
RR
3
RR
3
2
Instrument
2
Sensor send Read Reply (RR) with the requested payload to Initiator

Figure 49: Data delivery to a node with the proposed modification on the SpW-D specification

As shown in Figure 49, it is possible for the OBC to initiate the first transaction, by setting the path to the Mass Memory unit in the Reply Address Field. In this case only a single transaction and a single reply is required for a simple transfer. This however can be used for non-critical information since the initiator (OBC in this case) cannot know whether the Command was executed and the data is delivered to the destination. Concluding, for such cases it would be useful to add service category level to the SpW-D specification.

From the hardware implementation point of view, there is no impact in adopting this service as non-acknowledged RMAP transfers are already supported for RMAP functionality.

6.3 Segmentation

Segmentation has not been defined in SpW-D Draft B specification [AD, 2].

In this section Teletel's analysis on this function is presented.

6.3.1 Requirements

Segmentation & reassembly mechanisms have the following requirements from the protocol that will be designed to support it.

1. RMAP Write and RMAP Read transactions shall be supported.
2. For the transfer of a large SDU, the initiator shall split a request to multiple RMAP transactions in order to not violate the time-slots configured in the underlying SpW-D network.
3. The target shall inform its application that the SDU has been received only upon the reception of the last segment.
4. The initiator shall have a way to ensure that all segments have been delivered to/from the target[1].
5. Supporting more than one pending RMAP transactions is desirable since it increases the bandwidth utilization.
6. The target shall have a way to inform its application on the total length of the received SDU and the start address in which it is stored.
7. The target **should** be able to discriminate whether the received RMAP Command is an ordinary RMAP Command or a RMAP command corresponding to a SDU segment transfer.

6.3.2 Alternative system architectures

For the Segmentation and reassembly three possible solutions initially exist.

- Segmentation and Reassembly are performed at the Initiator:
  - The initiator receives a request for a transfer of a large SDU and chops it to multiple RMAP transactions according to the maximum payload size configured in the SpW-D network.
  - The target receives the commands and executes them without any processing.
  - For RMAP writes of large SDUs, the Initiator receives a SDU, chops it to multiple segments and transmits them through multiple RMAP Write transactions. The target responds with RMAP Write replies.
  - For RMAP reads the Initiator receives the request to read a large SDU and transmits multiple RMAP Read Commands. The Target responds with multiple RMAP Read Replies and the Initiator reassembles the consecutive replies.
  - In case a reply is missing (caused either by a command not reached the target or a lost reply) the target does not take any action. It is the responsibility of the Initiator to take any further actions.
- Target performs reassembly only:
  - The initiator performs as in the first case for both reads and writes
  - The target receives segmentation/reassembly information and is capable of identifying packets that are received out of order.
  - In case an out of order segment is received the target does not transmit a reply.

[1] Non-reliable delivery is not addressed in the SpW-D Draft B. However, if desired it can be supported without any modification to the RMAP Cores and is also compatible with the segmentation mechanisms presented herein.

- Target performs segmentation
  - The initiator performs as in the first two cases regarding RMAP Writes. However for RMAP reads it transmits a single RMAP command requesting to read the entire SDU
  - The target receives a RMAP Read command for the transfer of a large SDU and responds with multiple reply packets, each one containing a segment of the requested SDU.

The third solution is not compatible with the RMAP standard, since for one command, multiple replies are returned and therefore is not examined herein.

6.3.3 Implementation of Segmentation & Reassembly at the initiator

This solution requires minimal or no changes to the target

- Write transactions:
  - The Initiator chops the request for the transfer of a SDU to multiple RMAP write commands.
  - The first SDU segment is marked as “start segment” (or “Unsegmented” in case a SDU fits a single RMAP packet payload).
  - Subsequent segments, except for the last one are sent through successive RMAP write commands and are marked as “middle segments”.
  - All commands and replies have a field (Sequence Number) which is increased by ‘1’ for each successive segment.
  - The last segment of a SDU is sent through a RMAP write command marked as “last segment”.
  - When an entire SDU has been received at the target:
    - Either the target identifies that the last segment was received and notifies the application (requires minimal RMAP target modifications), or
    - The target does not do anything and the Initiator sends a RMAP write to a control address space to inform the target application, passing the total length of the SDU.
  - If an out-of-order Command is received at the target
    - Either the target does not send a reply, or
    - The target sends a reply and it is the responsibility of the initiator to take any further actions, if required.

- Read transactions:
  - The Initiator chops the request for the transfer of a SDU to multiple RMAP read commands.
  - The first SDU segment is marked as “start segment” (or “Unsegmented” in case a SDU fits a single RMAP packet payload).
  - Subsequent segments, except for the last one are sent through successive RMAP read commands and are marked as “middle segments”.
  - All commands and replies have a field (Sequence Number) which is increased by ‘1’ for each successive segment.
  - The last segment of a SDU is sent through a RMAP read command marked as “last segment”.
  - The target does not need to be aware that a SDU has been read so no modification is needed for RMAP targets (**TBC**)
  - If an out-of-order Command is received at the target
    - Either the target does not send a reply, or
    - The target sends a reply and it is the responsibility of the initiator to take any further actions, if required.

Since with this solution the target may not perform Sequence Number checking, there may be **a problem with RMAP transactions with non-incrementing addresses**. If the target implements a FIFO and a write command is correctly received, the payload is put in the FIFO. In case, however, the reply is lost the initiator may retransmit the segment and in the target's FIFO the SDU will not be correctly reassembled. The same for a read command if the reply is lost the initiator can not perform reassembly.

This problem may be handled either by the solution proposed in the next paragraph or with the use of a CRC at SDU level.

6.3.4 Implementation of Reassembly at the initiator

This solution requires target modifications

- Write & Read transactions:
  - The Initiator chops the request for the transfer of a SDU to multiple RMAP commands.
  - The first SDU segment is marked as "start segment" (or "Unsegmented" in case a SDU fits a single RMAP packet payload).
  - Subsequent segments, except for the last one are sent through successive RMAP commands and are marked as "middle segments".
  - All commands and replies have a field (Sequence Number) which is increased by '1' for each successive segment.
  - The last segment of a SDU is sent through a RMAP command marked as "last segment".
  - Upon the reception of a RMAP command the target checks the "type" field and:
    - In case it indicates "first segment" or Unsegmented" then:
      - The target ensures that no command has been received, or that the previous command was marked as "end segment" or "unsegmented".
      - If the above condition is not true the command is rejected and no reply is sent to the initiator, otherwise.
      - The PDU is accepted and its contents are the RMAP command is executed and the sequence number is stored to the target.
    - In case it indicates "middle segment" or "end segment" then the target shall ensure that:
      - Its Sequence Number is one more than the previously received Sequence number.
      - A command marked as "first segment" has been received.
      - In case one of the above conditions is not true, the RMAP command is discarded and no reply is sent to the initiator.

      Note: In case of restart (e.g. reset) of the initiator, the transfer can not be reinitiated since the target shall drop the "first segment" request. Other means shall be used to trigger the target to change its state for the unfinished transaction (e.g. FDIR).
  - When an entire SDU has been received at the target:
    - The target notifies the application that a SDU was transferred
    - It passes the total SDU length to the application (for write transfers)

6.3.5 Evaluation of the proposed alternatives

The comparison of the two proposed solutions is the following:

	Handling of out-of order packets	SDU Length	Target modifications required	Other issues
Reassembly at initiator side only	Supported for incrementing address transactions ONLY	Shall be included in the SDU, or sent through separate transaction by the initiator	None or minimal	SDU corruption due to retry for non-incrementing transactions
Reassembly at the target	Can be supported for both incrementing and non-incrementing address transactions	Can be calculated by the target	Extensive	

Table 15 Summary of the pros and cons of the proposed SpW-D segmentation alternatives

6.3.6 Segmentation & Reassembly information mapping

As discussed above, segmentation and reassembly mechanism requires the transmission of control information in the RMAP packets. Two alternatives exist for this purpose

- Control information inserted in the payload:
    - Sequence number, and segment type information may be inserted in the RMAP payload.
        - In this case this information will not be propagated in RMAP Write replies to the Initiator and RMAP Read Command to the target
        - The bandwidth utilization will degrade
        - Existing RMAP Cores are not compatible with this solution
    - Sequence number and segment type are mapped in the RMAP Header
        - Does not insert any extra overhead
        - Control information is present in all RMAP packets
        - Compatible with existing RMAP Cores

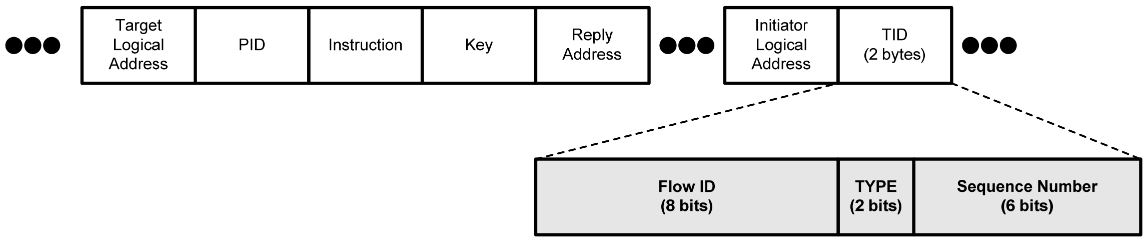


Figure 50: Mapping of Segmentation/Reassembly control information in the RMAP header fields

Implementation of the Segmentation and Reassembly with the second alternative is shown in Figure 50. With this proposal, both the Sequence number and the packet type are mapped into the TID header field. The type field identifies whether packet contains:

- The start of a SDU
- A middle segment of a SDU
- The end segment of a SDU
- An Unsegmented SDU

The sequence number (SQ) provides a 6-bit field used for the identification of out-of-order segments. The proposed mechanism works as follows:

- When the initiator transmits the first segment of large SDU, or an Unsegmented SDU the SQ can be set to any value
- Subsequent packets of the same SDU contain successive SQ numbers

In this way an out of order Command (at the target) or reply (at the initiator) can be identified in order to trigger the application layer or the Retry function.

Since the TID field is two bytes wide, more than 6 bits are available for the SQ field. However

- an initiator may issue commands to more than one targets and therefore there should be a field to identify replies from different targets
- The maximum address space in SpW is 223 addresses

Therefore, an 8-bits field is required in order to identify the issuer of the RMAP reply.

Compatibility of this solution with existing RMAP Cores however is not easily visible. This can be implemented in through the authorization logic. In order to support custom authorization logic, RMAP cores provide an authorization logic interface, through which the user is provided the facility to implement its own authorization logic. If the logic is extended in order to keep tracking of the second byte of the TID field this solution can be easily adopted in existing designs.

It has to be noted however that even in this case the following restrictions apply:

- **Either a PID different than RMAP's PID shall be used for SpW-D, or**
- **If the same PID is used a target will not be able to discriminate between normal RMAP packets and SpW-D segments and therefore it cannot support both functionalities**

6.4 Retry

Retry has not been defined in SpW-D Draft B specification [AD, 2].

This section presents possible alternatives for the Segmentation and Retry mechanism and their impact in SpW-D functionality. These alternatives are presented for future study and assessment activities.

As presented in section 4.5.2 "Retry mechanism" the major design issue for the implementation of the Retry mechanism is to define the time-slot(s) in which a failed RMAP transaction is allowed to be re-transmitted.

There are four initially identified alternatives that are analysed in this section:

1) Use a dedicated time-slot. Specific time-slot(s) may be allocated for retries at the end of the epoch (schedule table).
2) Use the next **free** time-slot that is allocated for the particular target (or channel) communication. The retry in this case may be sent in any time-slot scheduled to handle the same initiator/target pair as the failed RMAP transaction, provided that this time-slot is not needed for the initiation of an RMAP transaction.
3) Use **any** next time-slot that is allocated for the particular target (or channel) communication. The retry in this case may be sent in any time-slot scheduled to handle the same initiator/target pair as the failed RMAP transaction even if this time-slot is needed for the initiation of an RMAP transaction.
4) Use a number of dedicated time-slots for asynchronous traffic – namely asynchronous segment (this scheduling scheme is presented in section 6.1.2) and perform retransmission in the beginning of the asynchronous segment with higher priority than the asynchronous traffic.

6.4.1 Retransmission in dedicated time-slots(s)

For the first case we consider the following example, where an initiator performs transactions with Target 1 and 2 and at time-slot 1 the RMAP transaction fails. In this case the same transaction is re-initiated in a dedicated time-slot (at the end of the epoch) that is guaranteed to be free in the schedule table.

Note:

In the following figures the transactions the first number identifies the target and the number in parentheses uniquely identifies the transaction id to this target.

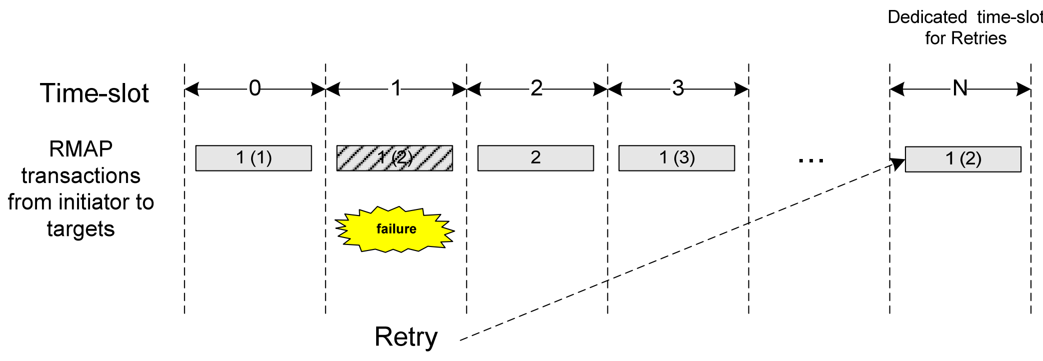

Dedicated time-slot for Retries
Time-slot
0
1
2
3
N
RMAP transactions from initiator to targets
1 (1)
1 (2)
2
1 (3)
...
1 (2)
failure
Retry

Figure 51: Retry performed in dedicated time-slot(s)

The first approach has the following issues:

- There will be waste of network resources (unused time-slots) when RMAP transactions do not fail. Taken into account the BER of the SpW links this waste of time-slots will be permanent in “almost” all epochs.
- In case that more transactions fail than the pre-allocated times-slots for retries (e.g. the transaction in time-slot 3 also fails) then the second transaction can not be sent in the dedicated time-slot in this epoch. As a result, the second failed transaction can only be re-initiated at the next epoch increasing latency.
- If more than one transaction fails from different initiators then the dedicated time-slots must guarantee that there will be no network resource conflict. As a result, as the number of initiators increases more dedicated time-slots for retransmission are required, thus increasing the waste of network resources.
- There may be out of order delivery of PDUs in case the Segmentation function is used. In this example the failed transaction in time-slot 1 is re-initiated after the transaction of time-slot 3, if these transactions contain PDU as a result of the segmentation function, then the retransmitted PDU will be delivered out of order and may be dropped by the Segmentation function.
- In case multi-slot scheduling is used then a number of dedicated consecutive time-slots must be allocated to accommodate the retransmission of RMAP transactions that have duration more than one time-slot. If we assume that a multi-slot transaction can occupy up to M time-slots, then M dedicated time-slots must be allocated for the retry of this initiator. Additionally, if it is required to support conflict free retries from N initiators in the same epoch then the number of dedicated time-slots for the re-transmissions will be increased prohibitively.
- In case multi-transaction scheduling is used and a failure occurs in time-slot where more than one transaction are initiated, then the failed transaction shall be handled as a normal transaction and retransmitted in the dedicated times-slots.
- **The main disadvantage of this solution is the waste of network resources due to the allocation of the dedicated time-slots.**

6.4.2 Retransmission in the next free time-slot for this target

For the second case we consider the following example, where an initiator performs transactions with Target 1 and 2 and at time-slot 1 the RMAP transaction fails. In this case the failed transaction is re-initiated in the next free time-slot that is allocated for the same target.

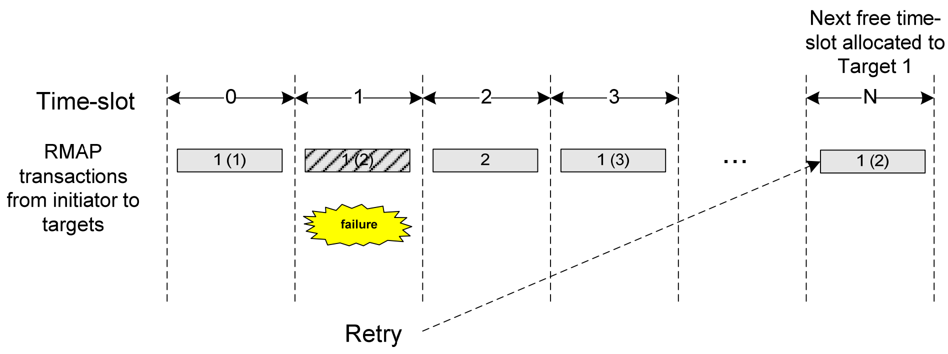

Next free time-slot allocated to Target 1
Time-slot
0
1
2
3
N
RMAP transactions from initiator to targets
1 (1)
1 (2)
2
1 (3)
...
1 (2)
failure
Retry

Figure 52: Retry performed in next free time-slot for the same target

The second approach has the following issues:

- There may be no free time-slot for the re-initiation of the failed transaction in the remaining time-slots of this epoch. So in this case the retry may be performed only in the next epoch. If in the next epoch there are already scheduled transactions and there is again no free time-slot for this target then the problem remains, the latency of the retry may be increased by more than one epoch interval and the Retry queue may overflow.

  In this case the Retry must be performed in a next time-slot for this Target with lower priority than any pending transactions. If we assume that the scheduler maintains a FIFO buffer for commands downloaded by the Host in order to be scheduled for a specific target in time-slots defined by the Schedule table, then the retry function in this approach must add the command of the failed transaction at the end of the FIFO buffer.
- The actual transmission of the command depends on the schedule table configuration and the number of commands that are pending transmission in the different time-slots. So the latency for re-initiating the failed transaction is highly variable.
- Again in this approach there may be out of order delivery of PDUs in case Segmentation function is used. In this example the failed transaction in time-slot 1 is re-initiated after the transaction of time-slot 3, if these transactions contain PDU as a result of the segmentation function, then the retransmitted PDU will be delivered out of order and may be dropped by the Segmentation function.
- In case multi-slot scheduling is used and the failure occurs in a transaction which has duration of more than one time-slot then the retransmission can only be performed in the next free time-slots that can accommodate the initiation of this transaction. In this case the problem is the same as with transactions that have duration of one time-slot but it is more restrictive regarding the selection of the free time-slots to initiate the failed transaction, as presented in the following example.
- In case multi-transaction scheduling is used and a failure occurs in time-slot where more than one transaction are initiated, then the failed transaction shall be handled as a normal transaction and retransmitted in a next “free” times-slot.

In the following example a multi-slot transaction fails in time-slot 0. The failed transaction can only be re-initiated in the next free time-slots that can accommodate the multi-slot transaction which are time-slots 1 and 2 in the next epoch.

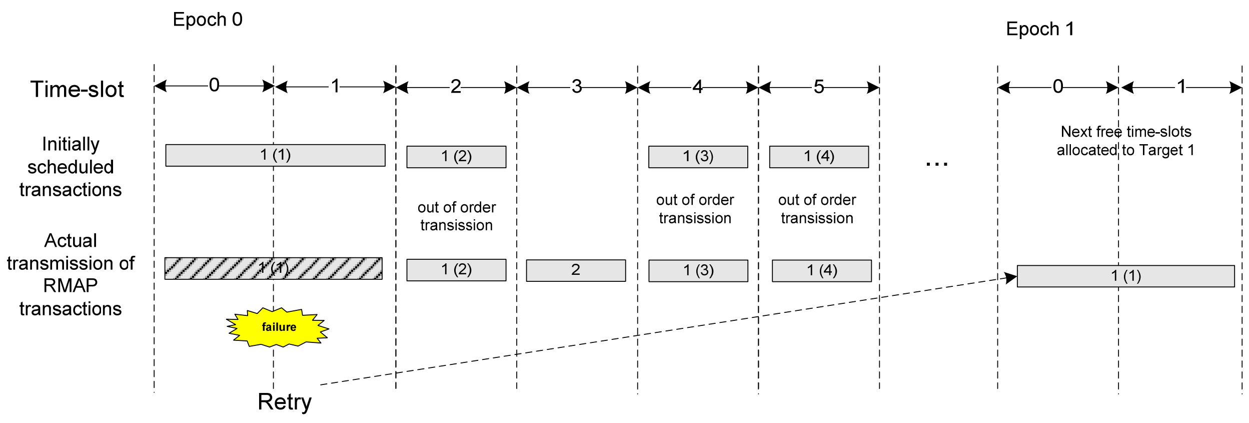


Figure 53: Retry performed in the next free time-slot for the same target in Multi-slot schedule

- **The main disadvantage of this approach is the increased latency, which is also highly variable.**

6.4.3 Retransmission in the next time-slot for this target

For the third case we consider the following example, where an initiator performs transactions with Target 1 and 2 and at time-slot 0 the RMAP transaction fails. In this case the same transaction is re-initiated in the next time-slot that is allocated for the same target.

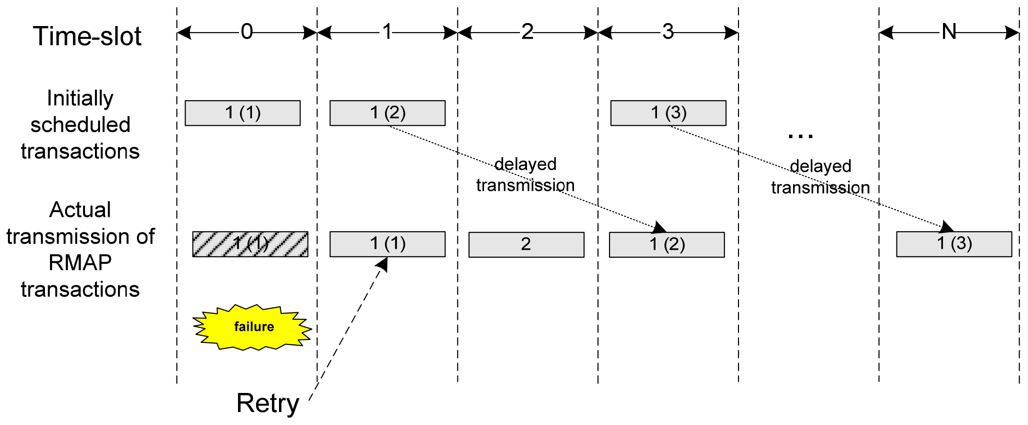


Figure 54: Retry performed in the next time-slot for the same target

The third approach has the following issues:

- The retry is performed in the next time-slot that is allocated for the particular target with a higher priority than any existing scheduled RMAP transactions for this time-slot. In this case there will be latency imposed in all pending RMAP transactions since they will be transmitted in the next time-slot for this target. In the provided example the transmission of the second and third transaction for target 1 are delayed by a number of time-slots.

  If we assume that the scheduler maintains a FIFO buffer for commands downloaded by the Host in order to be scheduled for a specific target in time-slots defined by the Schedule table, then the retry function in this approach must add the command of the failed transaction at the start of the FIFO buffer. The imposed latency in transmission of the commands in the FIFO buffer depends on the schedule table and in worst case is bounded to the epoch interval.

- In case multi-slot scheduling is used and the failure occurs in a transaction which has duration of more than one time-slot then the retransmission can only be performed in the next consecutive time-slots that can accommodate the initiation of this transaction. In this case the selection of the time-slots to initiate the failed transaction is more restrictive and there may be an out of order delivery if segmentation function is used, as presented in the following example.
- In case multi-transaction scheduling is used and a failure occurs in time-slot where more than one transaction are initiated, then the failed transaction shall be handled as a normal transaction and retransmitted in the next time-slot for this Target.

In the following example a multi-slot transaction fails in time-slot 0. The failed transaction can only be re-initiated at time-slot 4.

In this case the transaction in time-slot 2 is initiated before transaction 1 and is out of order.

The transactions that have initially scheduled for time-slots 4 and 5 are sent in the next available time-slots which are in the next epoch at time-slots 0 and 1.

As a result the multi-slot transaction can be transmitted but only in time-slot 4 and again in time-slot 2 of epoch 0 there will be an out of order transmission.

In this case the initiation of multi-slot transactions is shifted in time-slots and there is always an out of order transmission in time-slot 2.

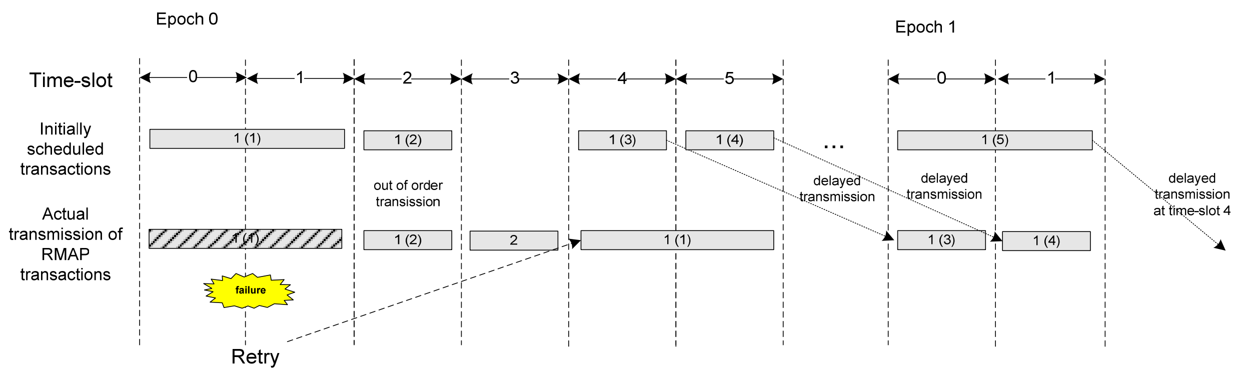


Figure 55: Retry performed in the next time-slot for the same target in Multi-slot schedule

- **The main advantage of this approach is that the latency imposed in transactions is bounded and predictable by the schedule table.**
- **The main disadvantage of this approach is that the transmission of commands may be disrupted by a retransmission, there will be latency and the transactions will not follow the initial ordered transmission scheduled by the Host.**

6.4.4 Retransmission in dedicated time-slots(s) for asynchronous traffic

The use of dedicated time-slots for asynchronous traffic is presented in section 6.1.2 as a potential improvement for scheduling in order to improve network efficiency in the presence of high asynchronous traffic (this is also used in FlexRay). If such technique is utilised in the target network then the retries can be performed in the asynchronous segment.

For this case we consider the following example, where an initiator performs transactions with Target 1 and 2 and at time-slot 1 the RMAP transaction fails. In this case the same transaction is re-initiated in the asynchronous segment (at the end of the epoch) and is transmitted using standard RMAP with higher priority than any pending asynchronous transactions from this initiator.

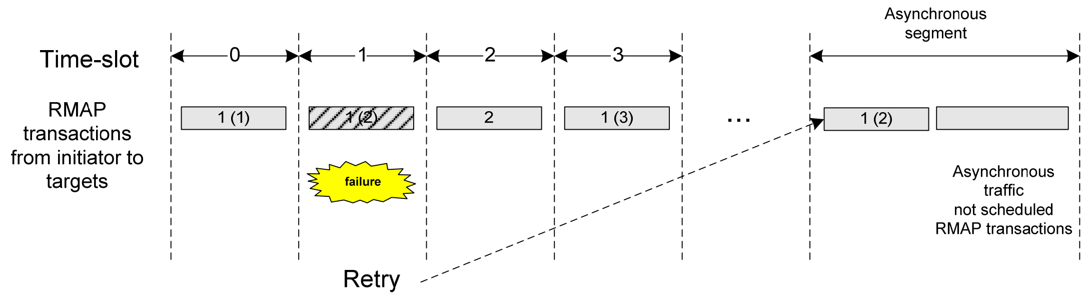

Asynchronous segment
Time-slot
0
1
2
3
RMAP transactions from initiator to targets
1 (1)
1 (2)
2
1 (3)
...
1 (2)
failure
Asynchronous traffic not scheduled RMAP transactions
Retry

Figure 56: Retry performed in dedicated time-slot(s)

This approach has the following issues:

- There will be contention in the transmission of the retry with the asynchronous transactions from other initiators. The duration of the asynchronous segment must be configured accordingly to accommodate a number of RMAP retries from all initiators in the network. If the duration of the asynchronous segment is large enough and since in this segment the maximum RMAP payload is also bounded then it can be guaranteed that a number of retries can be transmitted.
- At the start of the asynchronous segment the initiators shall wait for a small duration before transmitting RMAP commands for asynchronous traffic in order to allow other initiators to perform retries which have higher priority than the asynchronous traffic.
- In case a retry is not transmitted in the asynchronous segment due to limited segment duration and high contention, then the transaction can only be re-initiated at the next epoch increasing latency.
- There may be out of order delivery of PDUs in case Segmentation function is used. In this example the failed transaction in time-slot 1 is re-initiated after the transaction of time-slot 3, if these transactions contain PDU as a result of the segmentation function, then the retransmitted PDU will be delivered out of order and may be dropped by the Segmentation function.
- In case multi-slot scheduling is used then the duration of the asynchronous segment must be increased accordingly to accommodate the retransmission of RMAP transactions that have duration more than one time-slot.
- In case multi-transaction scheduling is used and a failure occurs in time-slot where more than one transaction are initiated, then the failed transaction shall be handled as a normal transaction and retransmitted in the asynchronous segment.

In case multi-transaction scheduling is used and a failure occurs in time-slot where more than one transaction are initiated then the failed transaction shall be handled as a normal transaction and retransmitted in the asynchronous segment.

- **The main disadvantage of this solution is that it requires the presence of the asynchronous segment in the network.**

6.4.5 *Evaluation of the proposed alternatives*

The comparison of the four different alternatives solutions is the following:

	Out of order delivery	**Latency in Retransmission**	**Waste of Resources**	**Other**
Retransmission in dedicated time-slots(s)	Yes	From 1 to 63 time-slots, depends on current time-slot	Yes, dedicated time-slot(s) for Retry	
Retransmission in the next free time-slot for this target	Yes	Highly Variable Depends on schedule table configuration and the pending RMAP transactions	No	
Retransmission in the next time-slot for this target	No	Depends on schedule table configuration	No	Imposes latency in pending transactions
Retransmission in dedicated time-slots(s) for asynchronous traffic	Yes	From 1 to 63 time-slots, depends on current time-slot	No (if asynchronous segment is utilised)	Requires an asynchronous segment in the network

Table 16: Summary of the pros and cons of the proposed SpW-D retry alternatives