Modeling Deterministic Spacecraft Networks with Constraint Programming

David Gibson[1], Steve Parkes[1], and Karen Petrie[2]

[1] Space Technology Centre, School of Computing, University of Dundee, Scotland
{davidgibson,sparkes}@computing.dundee.ac.uk
[2] School of Computing, University of Dundee, Scotland
karenpetrie@computing.dundee.ac.uk

Abstract. Traditionally, spacecraft deploy separate networks for handling payload and platform communications. This is due to the high-data rate required by payload instruments and the determinism and quality of service required by platform avionics command and response style transactions. Being able to utilise a single network for all onboard communication will decrease the cost and complexity of fulfilling the data-handling requirements of a mission.
This paper describes the background technology and SpaceWire-D standard, which is a protocol that provides a deterministic data-handling network, and presents an initial constraint programming model for the design and scheduling of a SpaceWire-D network.

1 Background

SpaceWire is a data-handling network for use onboard spacecraft to facilitate communication between instruments, mass-memory, onboard computers, telemetry, and other subsystems [1]. To enable data to flow between multiple subsystems, SpaceWire enabled routers can be used to direct traffic. These routers use wormhole routing which means that as soon as a packet has started to be received, it is forwarded to the required output port, assuming the port is not already in use. This simplifies SpaceWire routers as there is reduced packet buffering requirements and it allows arbitrary length packets as SpaceWire packets are terminated with a special token following the packet cargo. However, wormhole routing may cause blocking and network congestion if two packets are in contention for the same output port, so additional quality of service mechanisms need to be in place for networks that require determinism.

To provide a deterministic network on top of existing SpaceWire devices, a new protocol called SpaceWire-D [3] can be used. SpaceWire-D divides network time into 64 discrete time-slots in which one or more devices may initiate transactions. These transactions take place between an initiator node and a target node and their coordination is controlled by a schedule, which must ensure that no two transactions take place in a single slot if there is a chance they may collide and cause blocking. Two transactions may collide if the paths they use have any SpaceWire links in common. An exception to this is if the transactions

have the same initiating node, as SpaceWire-D allows a node to initiate transactions to multiple targets within a single slot but the initiator must guarantee all transactions are completed before the time-slot ends.

1.1 Real World Example

An example of a spacecraft with a network utilising determinism over SpaceWire is the ASTRO-H X-ray observation satellite [5]. The ASTRO-H network is divided into two subnets. The first is the data-handling section, which consists of the payload instruments, data storage, and telecommunication devices. The second subnet is the attitude and orbit control network which holds the devices used to adjust the rotation and speed of the spacecraft. The two subnets are connected by the attitude and orbit control processor. Determinism is provided by controlling all transaction initiation through a device called the Satellite Management Unit (SMU). As the SMU is the only initiator, the network schedule is simple, with every fourth slot allocated to node housekeeping, and every group of three slots to payload data-handling with the exception of the first group which is used for time and auxiliary data, and the last group which is allocated to router housekeeping.

Constraint programming has been applied to networking before, such as the SONET problem [4] and this paper looks at how constraint programming can be applied to the design and scheduling of SpaceWire-D networks.

2 Problem Description

The SpaceWire-D scheduling problem requires the generation of an optimal network topology and a deterministic schedule which meets the mission specific data-handling requirements. In this case, an optimal network means one that requires the minimum amount of routers. In addition to the general requirements of a SpaceWire-D network, there are user-defined requirements that must be met with regards to minimum bandwidth and timing of transactions, as payload instruments generate a certain amount of data to be stored in mass-memory; housekeeping information must be read from nodes at regular intervals; and commands must be able to be sent from the onboard computer to nodes at regular intervals.

In this model we will use the example mission of an earth observation satellite in low earth orbit (LEO). Due to the curvature of the Earth, the satellite will only be able to communicate with its ground station for part of its orbit, so there are two operation modes for the mission. The first is called normal mode, and for the duration of this mode, the cameras will generate data to be stored in mass-memory. The second mode is called downlink mode and adds the additional responsibility of moving data from mass-memory to downlink telemetry for transmission to Earth.

There are three types of data that are dealt with in this mission. Payload data is generated by the instruments and is of high data-rate but not critical to

the operation of the satellite. Housekeeping data is of low data-rate but critical for keeping the spacecraft in operation and is read from equipment at regular intervals so that onboard equipment can be maintained and problems diagnosed. Command data is the control commands sent to subsystems from the onboard computer. Payload data does not require determinism, but housekeeping and control data have time constraints, and so if we are to combine payload and platform data-handling on a single network, it must be deterministic.

The onboard equipment and subsystems for this mission include the following SpaceWire enabled devices with their identifiers in parentheses: an 8-port router (0), payload camera (1), payload camera (2), mass-memory (3), onboard computer (4), telemetry (5), thermal subsystem (6), power subsystem (7), and the attitude and orbit control subsystem (8).

3 Input Parameters

As input to the model, we take a list of pairs of nodes *node_pairs* which describe the required initiator/target pairs for the network. For example, housekeeping node-pairs allow data to be read from each device by the onboard computer, and in this case there are D devices on the network, so there are $D-1$ housekeeping node-pairs. Another example is payload data node-pairs, when instruments generate information to be read by or written to another node. The node-pairs required for the normal operation mode for this mission are shown in Figure 1 using the device identifiers from the previous section. Downlink mode adds an additional node-pair between the telemetry and mass-memory.

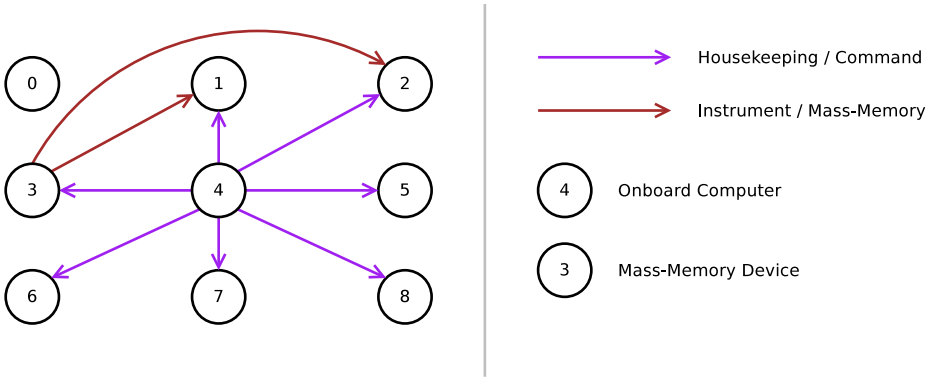


Fig. 1. Node-Pairs for the Normal Operation Mode

Another input parameter is the capacity of each device which is a list *capacities* of length D. For this mission $capacities = \{8, 1, 1, 1, 1, 1, 1, 1\}$, as we have an 8-port router and the other devices are single port nodes.

4 Decision Variables

Network topology is represented as a bidirectional graph and implemented as both an adjacency matrix *network_am* which is a two-dimensional square Boolean variable array of length D, and as an adjacency list *network_al* which is a length D set variable array.

The schedule is represented as a $|node\_pairs|$ length set variable array with the power set of 64 slots as their domain. Bandwidth allocated, in kB/s, to each node-pair *bw_alloc* is a $|node\_pairs|$ length array of integer variables. Two numbers are used to calculate the bandwidth. The first is the maximum number of transactions that a node can execute in a single slot, in this case 23. The second is the size of a transaction, which in this case is 1kB. These values are representative of the deterministic network, running at 64 slots/s, used in ASTRO-H [5]. The allocated bandwidth is then:

$$bw\_alloc[i] = 23 \cdot 1 \cdot |schedule[i]| \; \forall \, 0 \leq i < |node\_pairs| \tag{1}$$

5 Topology Constraints

For this model, we assume a simple graph and as such, there can be no multiple edges or loops. The former is implicit in the selection of a Boolean adjacency matrix, and the latter is restricted by the constraint on the network adjacency list:

$$i \notin network\_al[i] \; \forall \, 0 \leq i < D \tag{2}$$

There is a channeling constraint between *network_al* and *network_am* and as SpaceWire links are bidirectional, the graph is undirected and so *network_am* is symmetrical along its main diagonal:

$$channeling(network\_am, network\_al) \tag{3}$$

$$network\_am[i,j] = network\_am[j,i] \; \forall \, 0 \leq i \leq j < D \tag{4}$$

The maximum number of links that a device can be connected to is specified by the *capacities* input parameter:

$$|network\_al[i]| \leq capacities[i] \; \forall \, 0 \leq i < D \tag{5}$$

To make sure that the network is connected and that each device is connected to at least one router, in this case the only router, we can add a constraint to make sure the set of devices connected to the router contains the set of all non-router devices:

$$network\_al[0] \supseteq \{1, \ldots, D-1\} \tag{6}$$

6 Schedule Constraints

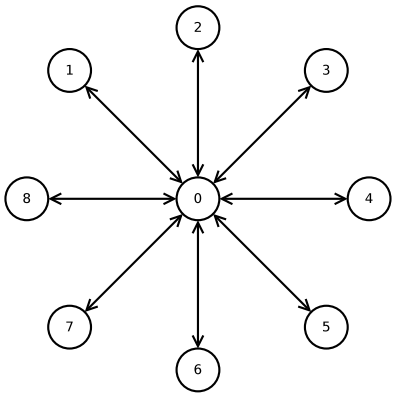
To make sure that a valid SpaceWire-D schedule is generated, no node-pair can be scheduled in a slot if it causes blocking with another node-pair in the same slot. In a single router network, this simply means that no two node-pairs can be scheduled in the same slot if they have the same target node.

7 Mission Constraints

For this mission, we consider two types of user-defined constraints. Interval constraints allow one or more node-pairs to be allocated slots at regular intervals based on a resolution value. If we assume that we are running SpaceWire-D slots at 64 slots/s, then a node-pair with a resolution value of 64 is allocated in each slot, and a node-pair with a resolution value of 32 is allocated 32 slots at equal distances starting from the first slot. Running at 64 slots/s, our mission requires that the housekeeping node-pairs meet an interval constraint resolution of 4, and the command node-pairs meet a resolution of 16. The second type of constraints are bandwidth constraints, and the mission requires instrument and telemetry node-pairs to be allocated a minimum data-rate of 640kBps

8 Results

Implementing the model using the constraint solver Gecode [2] for both operation modes results in a solution as shown in Figures 2 and 3, with the same network topology for both, but different schedules. The schedules are similar for the housekeeping/command node-pairs, but differ in the slots allocated to instrument node-pairs.



	0	1	2	3	4	5	6	7	8
0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0

Fig. 2. Network Topology Result

Normal Mode

NodePair	Source	Dest.	Slots	Bandwidth
0	4	1	0,16,32,48	92
1	4	2	0,16,32,48	92
2	4	3	0,16,32,48	92
3	4	5	0,16,32,48	92
4	4	6	0,16,32,48	92
5	4	7	0,16,32,48	92
6	4	8	0,4,8,12,16,20,24, 28,32,36,40,44,48, 52,56,59..63	460
7	3	1	1..15,17..31,33..47, 49..63	1380
8	3	2	1..15,17..31,33..47, 49..63	1380

Downlink Mode

NodePair	Source	Dest.	Slots	Bandwidth
0	4	1	0,16,32,48	92
1	4	2	0,16,32,48	92
2	4	3	0,16,32,48	92
3	4	5	0,16,32,48	92
4	4	6	0,16,32,48	92
5	4	7	0,16,32,48	92
6	4	8	0,4,8,12,16,20,24, 28,32,36,40,44,48, 52,56,59..63	460
7	3	1	30,31,33..47,49..63	736
8	3	2	30,31,33..47,49..63	736
9	5	3	1..15,17..29	644

Fig. 3. Network Schedule Result

9 Future Work

The example mission in this paper was a simple SpaceWire-D network. Multiple router networks will be considered next, which will require dealing with much larger topology domains and paths between initiator/target node-pairs, ensuring they are scheduled without collisions. SpaceWire networks allow multiple connections between routers and so multi-graph variables should be utilised in future work. Alternative models will need to be explored to find how they scale to meet the added complexities of a multiple-router network.

In order to allow the general purpose modeling of SpaceWire-D networks, an interactive constraint application needs to be built to allow network engineers to design these types of networks. Ease of use is important for this application as the users may not be familiar with constraint programming.

References

1. ECSS: SpaceWire - Links, nodes, routers and networks. Standard, ECSS (Jul 2008), ECSS-E-ST-50-12C
2. Gecode Team: Gecode: Generic Constraint Development Environment (2006), available from http://www.gecode.org
3. S. Parkes, A. Ferrer, S.M., Mason, A.: SpaceWire-D: Deterministic Data Delivery with SpaceWire. Third International SpaceWire Conference, St Petersburg (2010)
4. Smith, B.M.: Symmetry and search in a network design problem. In: Proc. 2nd Int. Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR), LNCS 3524. pp. 336–350. Springer (2005)
5. T. Yuasa, e.a.: A Deterministic SpaceWire Network Onboard the Astro-H Space X-Ray Observatory. Third International SpaceWire Conference, San Antonio (2011)