An End-to-End flow control mechanism for improvement data delivery reliability in SpaceWire

Alexey Abramov, Pavel Volkov
St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190000, St. Petersburg Russia
alexey.abramov@guap.ru, volkov@aanet.ru

Abstract

Data reliability and validity of the data are one of the main problems of transmitting information over a wired communication channel in real time nowadays. At the same time we must do not forget that the redundancy that occupied due to the use of different methods (like noise-coding), increases the volume of transmitted information, that in turn leads to an increase in the time of data transmission. With a large volume of data transmitted and the requirements of immediate response, some methods to improve the reliability of information transmission are not effective.

This article describes one of the methods for improving the reliability of data transmission over SpaceWire standard [1].

Index Terms: SpaceWire, Protocol, Reliable Data Delivery, Flow control, End-to-End.

I. INTRODUCTION

Overload is key issue that must be addressed when designing computer networks. Solving this problem is a complex task. Overloading occurs when the number of packets transmitted over the network, starts to approach the value of allowable bandwidth. The best way to solve this problem is to keep the number of packets in the network below the level at which the bandwidth begins to fall sharply.

SpaceWire is a spacecraft communication network based in part on the IEE 1355 standard of communications. It is coordinated by the European Space Agency (ESA) in collaboration with international space agencies including NASA, JAXA and RSA.

A SpaceWire system comprises several units connected together with SpaceWire links, either directly or indirectly via one or more SpaceWire routers. The SpaceWire links are high-speed, bi-directional, point-to-point communication links operating at a baud rate of between 2 and 400 Mbits/s. SpaceWire runs over a cable containing four twisted pairs. At each end of a SpaceWire link is a coder/decoder (CODEC) [2] which encodes packets of data to be transmitted into a serial bit-stream and decodes an incoming serial bit-stream into a data packets. The serialized data is encoded using a data-strobe (DS) technique where the strobe changes state at a bit interval whenever the data remains constant. This allows simple clock recovery in the receiver by XORing the data and strobe signals together and provides better skew tolerance than data-clock encoding. The DS signals are transmitted using low-voltage differential signaling (LVDS). SpaceWire CODECs are used in both nodes and routers and consequently form an important element of any SpaceWire system.

A. Flow control mechanism in SpaceWire

The data is serialized in the transmitter and encoded through the SpaceWire link using DS encoding. Data is only transmitted when the CODECs at each end of the SpaceWire link have reserved buffer space using the SpaceWire flow control mechanism. When space for eight more data characters in the receive buffer is available then a FCT (Flow Control Token) is transmitted to allow eight data characters to be transferred. In this way buffer overruns are detected as a link error by the CODEC. The serial data-strobe encoded bit-stream is received, error checked and decode into SpaceWire characters in the receiver. Receiver data characters which were requested using FCT are placed in the receiver buffer.

The recovery scheme described in the SpaceWire standard is implemented internally in the CODEC. Errors detected in the CODEC include incorrect parity bit received, escape character sequence error, receiver disconnection error, receiver credit error (data character received when not expected), transmit credit overflow and character sequence error at startup.

When an error is detected the SpaceWire link is disconnected and the tail of the packet currently being received is assumed to be lost. To preserve the packet structure used in SpaceWire system an EEP (Error End of Packet) character is added to the tail of the packet. This indicates the packet was partly received but an error occurred before the normal packet EOP (End Of Packet) marker was received. As the SpaceWire link has been disconnected then the FCT characters which were transferred to the other end of the link, denoting receive buffer space, are assumed to be invalid and the internal receiver FCT pointer are update to free any reserved buffer space. The transmitter error recovery scheme clears the tail of the packet being transmitted when the error was detected from the transmitter FIFO. Therefore when the next link start-up and connection occurs the next data character to be transmitted is the head of the next SpaceWire packet. In this way the error recovery system is transparent to the host system which needs only to perform the SpaceWire packet level protocol. So, there is no mechanism in SpaceWire standard to retransmit the packet is assumed to be lost. Apparently, it is an urgent issue for specific tasks [3], and thus, there is necessity to create a protocol for reliable data delivery over SpaceWire.

II. Reliable data delivery

The main requirements for Reliable Data Delivery Protocol (RDDP) are to utilize SpaceWire capabilities to provide a packet delivery protocol that is able to detect and recover lost packets. The protocol is also required to be simple as possible and flexible so that it can be adapted as needed to different host data throughput requirements.

To increase data delivery reliability RDDP is based on one of the retransmitted packets method – sliding window method with window size equals 1, also known as method of idle time of a source [4].

RDDP specifies a packet format than is consistent with the standard SpaceWire packet. RDDP does not dictate packet routing through a SpaceWire network be it point-to-point or composed of multiple routers. Packet routing is handled by the SpaceWire layer, so protocol does not try to improve on SpaceWire packet routing. The one-byte destination logical address (DLA) is sufficient to get a SpaceWire packet to its destination though a variety of network configuration.

Figure 1 illustrates RDDP perspective for packet routing. For RDDP, there is no “path” to the remote end point, only a DLA. Protocol does not have, and does not require, knowledge of how a packet gets to its destination. This keeps protocol simple and allows any implementation using this protocol to remain independent of possible SpaceWire network changes. Figure 2 shows protocol packet format.

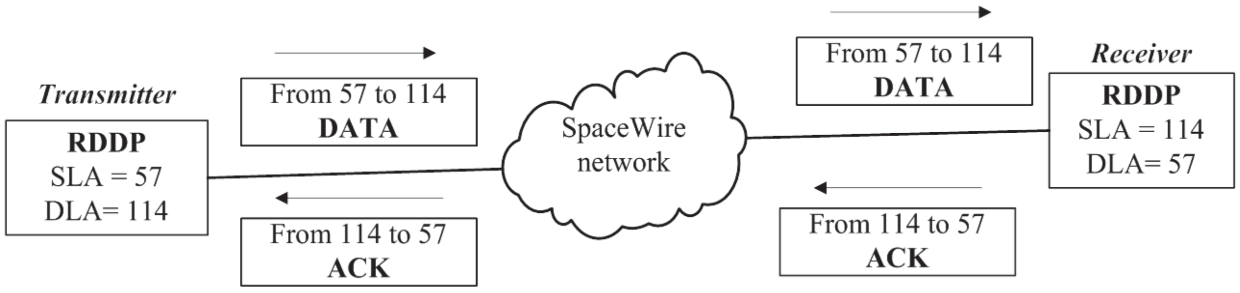

Figure 1. Protocol Packet Routing

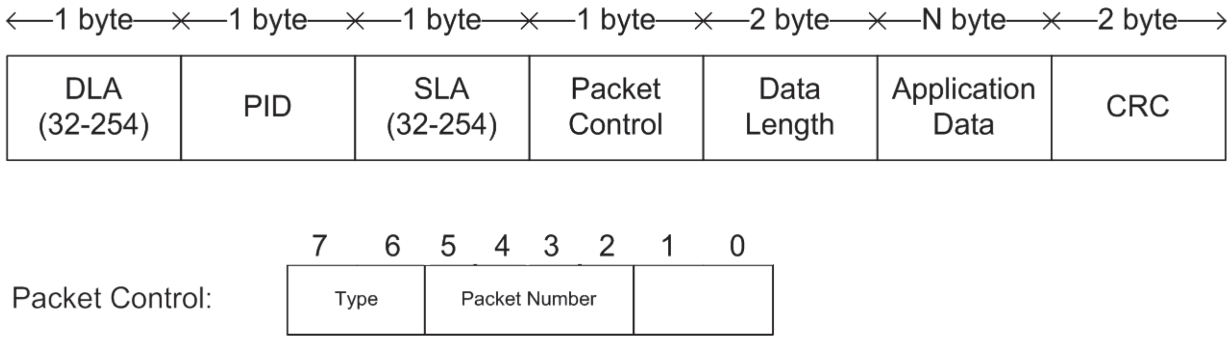

Figure 2. Protocol Packet format

Packet’s field description:

- DLA – Destination logical address of receiver. The range of values from 32 to 254.
- PID – Number of protocol (ECSS-E50-11 draft B). It equals 254.
- SLA – Source logical address of the transmitter. The range of values from 32 to 254.
- Packet Control – Packet control information.
- Data Length – length of “Application Data”.
- Application Data – it‟s a “sending information” of sending packet; this field does not use for ACK or NACK.
- Type – Type of message (0 – Data, 1 – ACK, 2 – NACK, 3 – reserved).
- PN – Number of packet. Without this field it is not clear to understand what we have received:
- It’s a copy of the packet due to retransmit because of expired timeout.
- Transmitter itself specifically sends it a second time.

Transmitter defines a sequence number for each sending packet. RDDP uses a positive acknowledgment (ACK) for each correctly packet transmitted. If an ACK is not received within a timeout interval, then packet is retransmitted. After a maximum number of retries have been exhausted for a packet, the transmitter will declare that there is an error occupied while sending the packet. The requirement for the receiver to acknowledge each packet allows the transmitter to detect lost packet if an ACK is not received. On the receiver end, a packet is acknowledged if it has a valid Cycle Redundancy Check (CRC) character and header. Moreover RDDP uses a negative acknowledgment (NACK) to change a current sequence number on the transmitter. RDDP state machines of transmitter and receiver are present on Figure 3 and Figure 4 below.

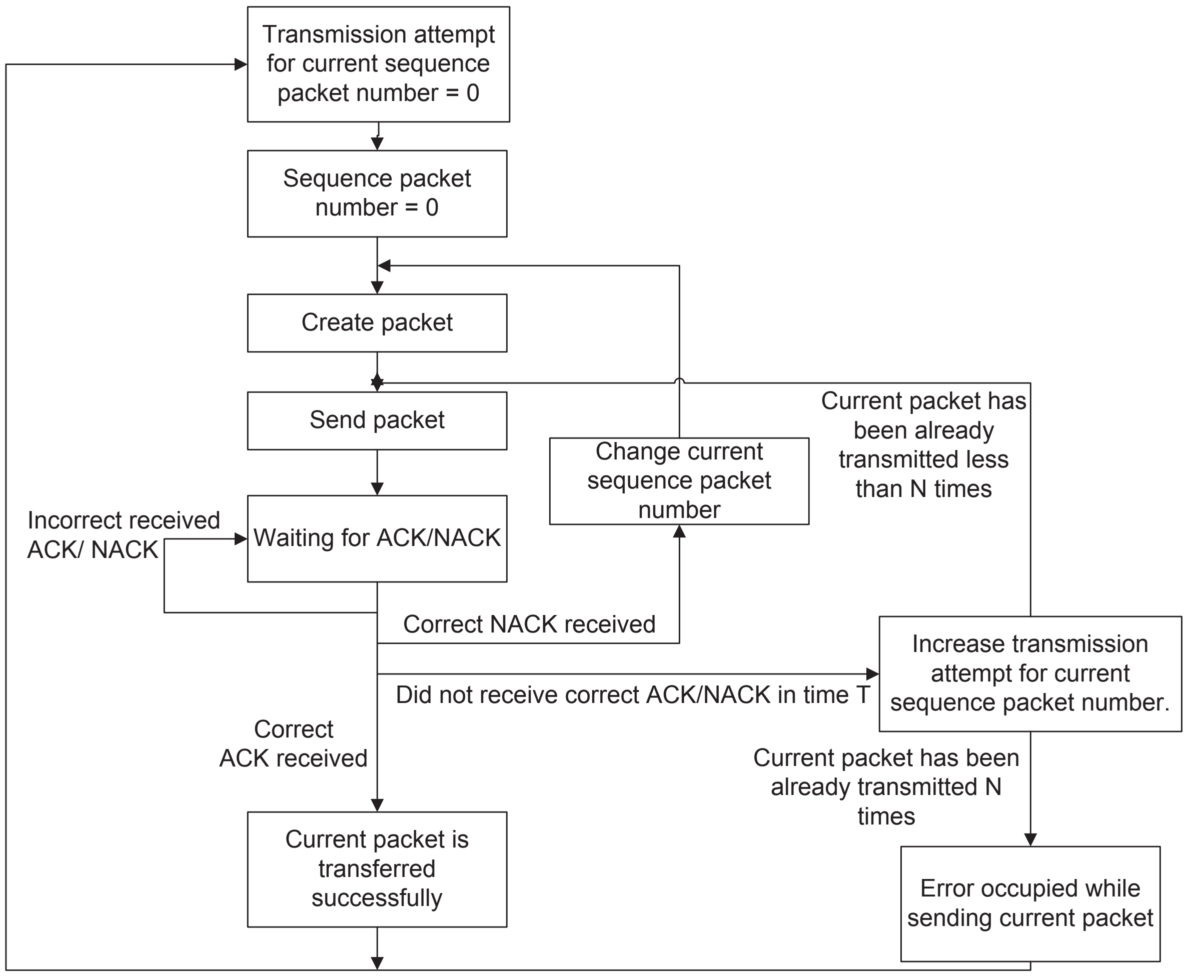
Transmission attempt for current sequence packet number = 0
Sequence packet number = 0
Create packet
Send packet
Change current sequence packet number
Current packet has been already transmitted less than N times
Incorrect received ACK/ NACK
Waiting for ACK/NACK
Correct NACK received
Increase transmission attempt for current sequence packet number.
Did not receive correct ACK/NACK in time T
Correct ACK received
Current packet has been already transmitted N times
Current packet is transferred successfully
Error occupied while sending current packet

Figure 3. RDDP State machine of the transmitter

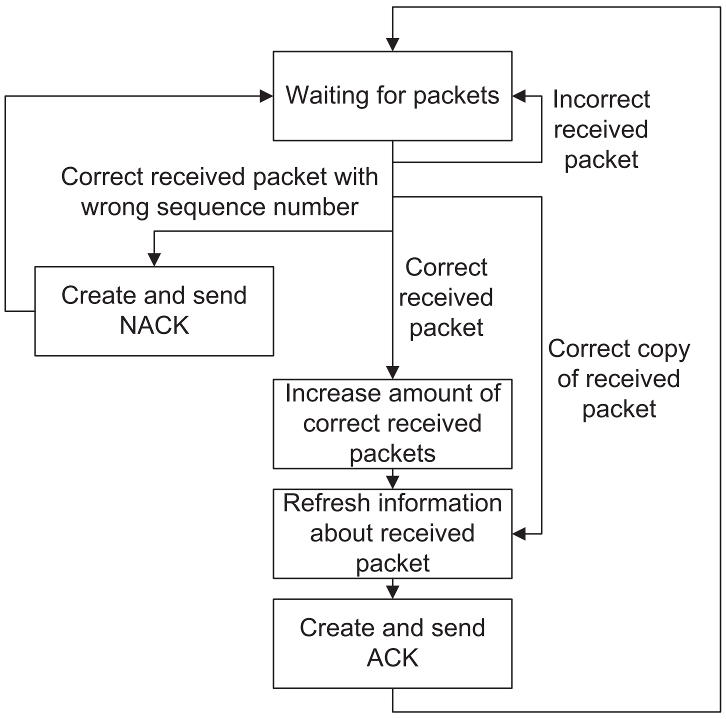
Waiting for packets
Incorrect received packet
Correct received packet with wrong sequence number
Create and send NACK
Correct received packet
Correct copy of received packet
Increase amount of correct received packets
Refresh information about received packet
Create and send ACK

Figure 4. RDDP State machine of the receiver

III. RDDP IMPLEMENTATION

RDDP was implemented as a module on top of two different drivers, such as:

1. Driver MCB-01.2.ASIC ver. 1.411 (without OS, MIPS32 architecture)
2. Driver MCB-01.2m.FPGA ver. 1.63 (without OS, MIPS32 architecture).

It performs the functions of the transport layer of the OSI model. Module is written in C. Timeout interval is fixed (there is no adaptive definition mechanisms of timeout interval).

A. RDDP usage requirements

Several requirements to protocol usage are:

- Module provides reliable data delivery between two SpaceWire devices end-to-end connected.
- Module needs to be used on both sides at the same time.
- Maximum size of packet depends on the host system architecture.
- Programmer needs to keep in mind about the maximum size of the packet when writing an application. Thus he is in charge to “cut the information to pieces” which must be fit in the packet.
- In a network is admissible to use only logic addressing with no heading removal.
- It is impossible to use simultaneously module at one device and directly driver functions on other for packet transmitting.
- Before working with module it should be initialized.
- After working with module it should be reinitialized.

B. Module functions

Three functions of the module are accessible to programmer, who is writing an application:

1. **unitDataACK_init() –** function to initialize the module.
2. **unitDataACK_sendPacket(**[*dstLA*], [*srcLA*], [channelNum], [packetAddress], [sizeInBytes]**) –** function to transmit the packet from one device (host) to another. It does not return management to the application until the package will not be transmitted or the is no way to transmit the packet.
3. **unitDataACK_deinit() –** function to deinitialize the module.

Details can be found in the protocol documentation [5].

C. Tested devices

Module has been written for several SpaceWire:

1. SpaceWire MCB-01 board with the SpaceWire bridgeMCB-01.2.ASIC (ver. 0645) with 4 SpaceWire links (400 Mbit/s)and processor CPOS_2 (100 MHz) on board.
2. SpaceWire MC24EM board with the SpaceWIre bridge MCB-01.2m.FPGA with 4 SpaceWire links (182 Mbit/s) and processor MC__24 (50 MHz) on board.

IV. CONCLUSION

RDDP does not attempt to duplicate or improve on the considerable capabilities provided by SpaceWire. Certainly, lacks of method of idle time of a source are especially appreciable on low-speed links, but on the other hand it is flexible and simple as possible. It built on top of SpaceWire the ability to recover lost packets, and to ensure to higher lever processes that packets are as error free as possible. The implemented module based on the RDDP ready to be applied to SpaceWire applications, which could be run on SpaceWire MC24EM board and SpaceWire bridge MCB-01.2m.FPGA board.

REFERENCES

[1] “SpaceWire: Links, nodes, routers and networks”, ECSS-E-ST-50-12C, 31 July 2008.

[2] Chris McClements, Steve Parkes, Agustin Leon, “The SpaceWire CODEC”, International SpaceWire Seminar (ISWS 2003) 4-5 November 2003, ESTEC Noordwijk, The Netherlands.

[3] A. Krimchansky, W.H. Anderson C. Bearer, “The Geostationary Operational Satellite R Series SpaceWire Based Data System Architecture”, International SpaceWire Conference 22-24 June 2010, St. Petersburg, Russia.

[4] N. Olifer, V. Olifer, "Computer Networks: Principles, Technologies and Protocols for Network design", Wiley India Edition, First edition.

[5] MiT, “Reliable Data Delivery Protocol”, ver. 1.2, 13 October 2010.