CCSDS Packet Transfer Protocol for space applications

A. Dmitrov

St.Petersburg University of Aerospace Instrumentation
67 B.Morskaya st., 190 000, St.Petersburg, Russia
andrey.dmitrov@gmail.com

Abstract

The aim of the Consultative Committee for Space Data Systems (CCSDS) Packet Transfer Protocol is to transfer CCSDS Packets across a SpaceWire network. The CCSDS Packet Transfer Protocol has been designed to encapsulate a CCSDS Space Packet into a SpaceWire packet, transfer it from an initiator to a target across a SpaceWire network, extract it from the SpaceWire packet and pass it to a target user application. This protocol does not provide any means for ensuring delivery of the packet nor is it responsible for the contents of the packet being a CCSDS Space Packet. This Standard has been prepared by the ECSS-E-ST-50-53 Working Group, reviewed by the ECSS Executive Secretariat and approved by the ECSS Technical Authority. The paper proposed the implementation of this Protocol under OS Linux.

Index Terms: SpaceWire, Linux, CCSDS PTP.

I. SERVICES ARCHITECTURE

Linux offers embedded designers an inherently modular operating system that can be easily scaled down to compact configurations suitable for embedded design. Plus, Linux is the fastest growing server operating system and is rapidly moving into embedded applications.

For chips manufactured by ELVEES RnD Center with built-in SpaceWire [1] channels has been developed software to work effectively in the OS Linux [3] environment:

• Drivers for SpaceWire channel controllers, which allow the use of SpaceWire channels as usual network devices. Each channel is represented by its network interface with IP address, so all TCP/IP applications, using Berkeley Software Distribution (BSD) POSIX sockets API, would work over SpaceWire without any change.

• For use in distributed and parallel systems, implemented a Time Access Service (TAS), that provides applications with a consistent interface to a local time source that is correlated to some centrally maintained master onboard time source. The time values provided by this service might typically be used by the application to schedule some operation, such as the acquisition of an image or to time stamp locally generated telemetry data.

• Distributed interrupts service (DIS) is a service, meant for operating with SpaceWire distributed interrupts. Interface for applications is a standard POSIX real-time signals interface.

• To provide operability TCP / IP stack designed ARP for SpaceWire networks.

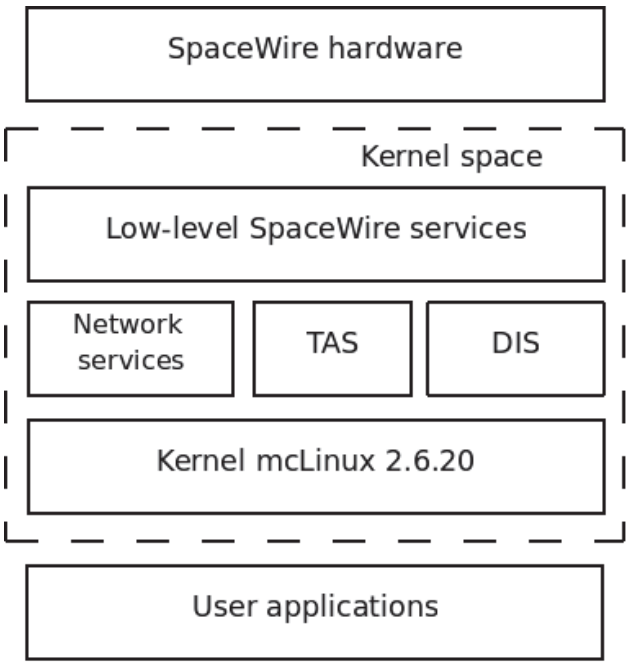

Figure 1: OS Linux architecture with SpaceWire services

II. NETWORK SERVICES

Each SpaceWire channel is represented as linux network device with its own IP address, which supports data transmission over TCP/IP. User applications are provided with standard POSIX socket interface, so a lot of network applications can be used over SpaceWire without any change. Among them http, ftp, telnet clients and servers, and a wide range of standard utilities for network configuration and diagnostics, as ifconfig, route, ping, nuttcp etc.

III. CCSDS PACKET TRANSFER PROTOCOL

This Standard is one of the series of ECSS Standards intended to be applied together for the management, engineering and product assurance in space projects and applications. ECSS is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the purpose of developing and maintaining common standards.

The CCSDS Packet Transfer Protocol has been designed to encapsulate a CCSDS Space Packet into a SpaceWire packet, transfer it from an initiator to a target across a SpaceWire network, extract it from the SpaceWire packet and pass it to a target user application. This protocol does not provide any means for ensuring delivery of the packet nor is it responsible for the contents of the packet being a CCSDS Space Packet.[2]

Figure 2 illustrates the location of the CCSDS Space Packet transfer Protocol in a typical onboard protocol stack. The CCSDS Space Packet transfer Protocol provides a unidirectional data transfer service from a single source user application to a single destination user application through a SpaceWire network.

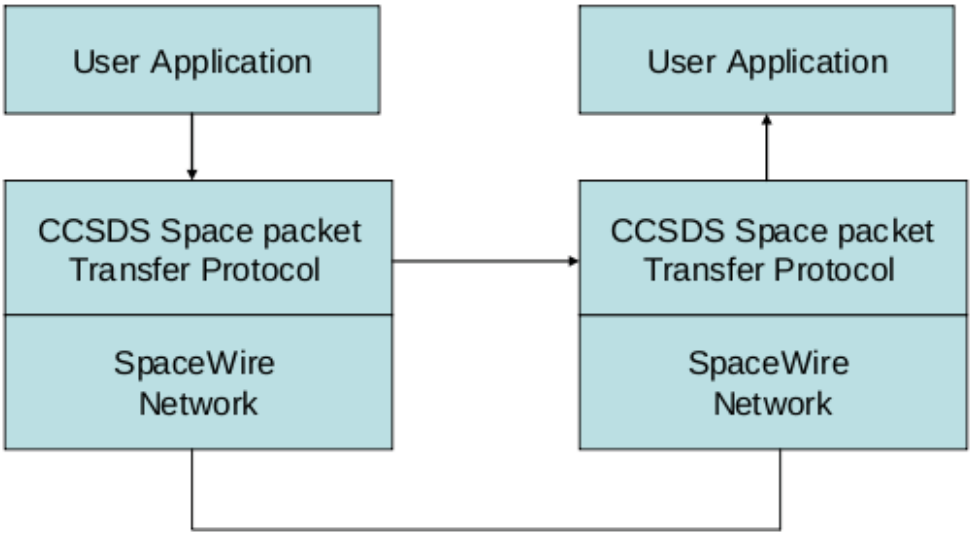

Figure 2: Protocol configuration

A. Service features

- Unidirectional (one way) data transfer service.
- Asynchronous Service. There are no predefined timing rules for the transfer of service data units supplied by the service user. The user may request data transfer at any time it desires, but there may be restrictions imposed by the provider on the data generation rate.
- Unconfirmed Service: the sending user does not receive confirmation from the receiving end that data has been received.
- Incomplete Services. The services do not guarantee completeness, nor do they provide a retransmission mechanism.
- SDU format: the service does not check the format of the submitted CCSDS Space packet.

- Non sequence Preserving Service. The sequence of service data units supplied by the sending user may not be preserved through the underlying network.
- The Space Packet Transfer Protocol does not provide any mechanisms for guaranteeing a particular quality of service.

B. Service parameters

- The value of the packet length shall be equal to at least 7 and at most 65542 octets.
- The Target SpaceWire Address parameter shall be used to define the path to the Target when SpaceWire path addressing is being used.
- The Target Logical Address parameter shall be used to define the logical address of the Target that is to receive the CCSDS packet.
- The User Application Value shall be an 8 bit value which is transferred along with the CCSDS packet to the Target.
- The Protocol Identifier field shall be an 8 bit field that contains the Protocol Identifier. It shall be set to the value 2 (0x02) which is the Protocol Identifier for the CCSDS Packet Encapsulation Protocol.
- The Status code parameter shall be used to indicate the validity of the packet to the receiving service user and take one of the following values:
  1. 0x00 indicates that the packet is ok
  2. 0x01 indicates packet arrived terminated by EEP
  3. 0x02 indicates reserved field was non zero

The CCSDS Packet Transfer Protocol packet shall contain the fields shown in Figure 3.

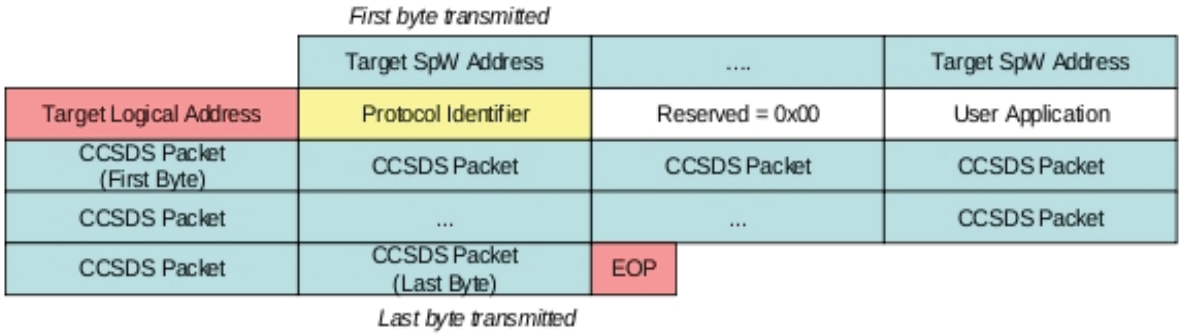

Figure 3: Encapsulated CCSDS Packet format

IV. Comparison of UDP and PTP

UDP is well known analog of PTP. UDP protocol meets all the features of the protocol PTP, but it additionally provides an opportunity to transmit data in both directions with packet header containing return address.

It is impossible get the information about SpaceWire end of packet (EOP) symbol from the UDP packet. UDP has no Status code.

Size of PTP packet header is 4 bytes, minimum size of UDP packet header is 28 bytes. The difference is 24 bytes, if packet contains 1500 bytes overhead is approximately 1.5%.

V. Protocol implementation

The protocol was implemented as a kernel module for OS Linux. As TCP / IP protocol stack, it performs the function of the transport and network levels of the OSI model. Function proto_register(), which adds a protocol to the existing list, is performed after loading the module into memory. Also the sock_register() function is called by a protocol handler when it wants to advertise its address family, and have it linked to the socket module. It creates an entry for this

protocol in the net_families table. The net_families contains the protocol list and all the protocols are registered here. Due to this, it is possible to access the functions of the protocol using a standard BSD sockets interface.

The socket API is a general interface for Unix networking and allows the use of various network protocols and addressing architectures. The network structure of Linux kernel with the PTP protocol is shown in Figure 4.[4]

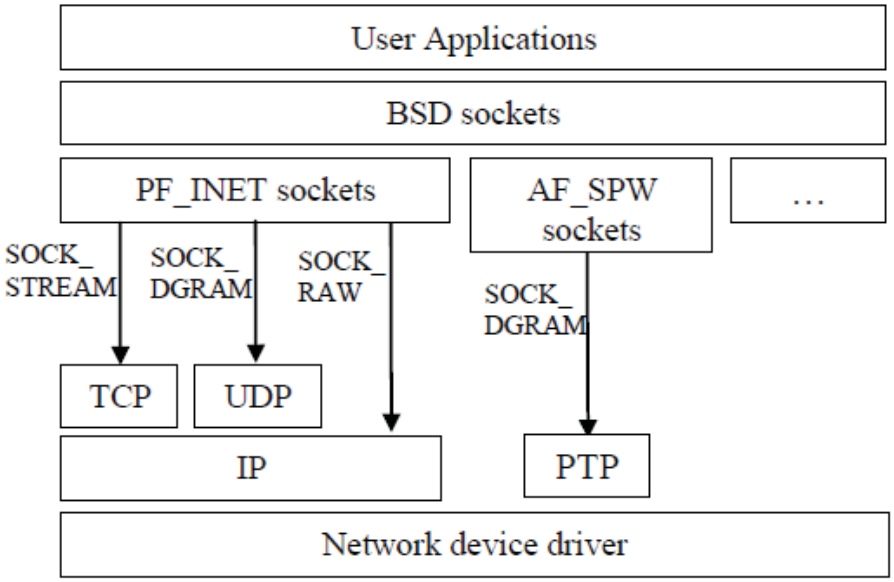

Figure 4: The structure of sockets in the Linux kernel

The User Application field specifies the value of virtual channel that identifies the receiving application. This value is assigned once when creating a connection. Hardware address is determined in accordance with the value of User Application.

Before sending or receiving packets connection endpoint using the function socket() has to be created. After that, UserApplication value (for receiving and transmitting packets) must be passed to the protocol. Transfer UserApplication values is performed by the bind() function. After performing the described steps data transmission over the network can be started. Set of standard functions for reading / writing to BSD sockets must be used for this purpose.

VI. Characteristics of the Implementation

Series of tests were developed and performed to evaluate the characteristics of the implemented protocol. The obtained results were compared with the results of similar tests for popular protocols TCP/IP and UDP/IP. Total measurements are shown in Figures 5 and 6.

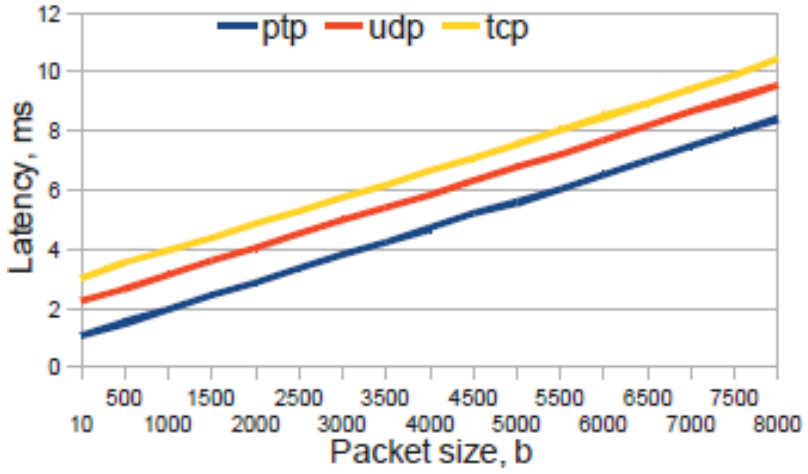

Figure 5: Protocols latency

Figure 5 illustrates the latency depending on packet size for the TCP, UDP and PTP. In Figure 6 we can see the values of throughput depending on the size of the packet.

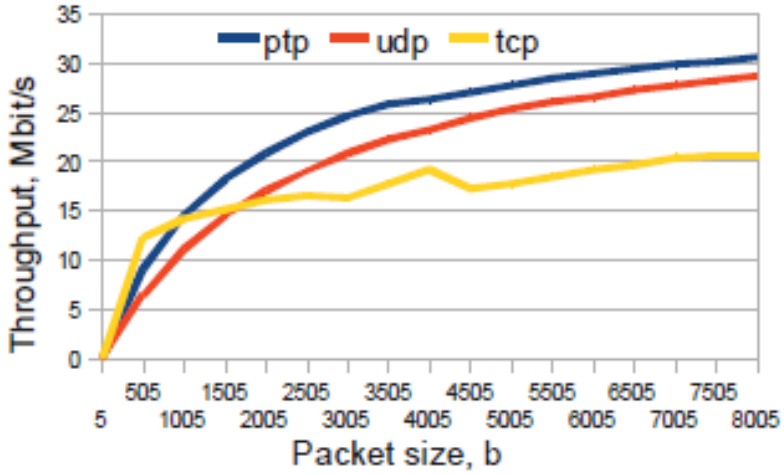

Figure 6: Protocols bandwidth

Obviously, the implementation of the CCSDS PTP protocol has the advantage in latency and bandwidth of data transmission. This is result of simplicity of the PTP, comparing to TCP and UDP. The protocol does not provide any guarantees on packet delivery or any additional services. This can be considered as protocol disadvantages.

VII. Conclusion

Implementation of the CCSDS PTP has shown good results in the tests. The developed protocol may find a use in the aerospace industry and can be used as well as popular TCP or UDP. A particular choice of protocol depends upon the task.

REFERENCES

[1] ECSS-E-50-12A “SpaceWire - Links, nodes, routers and networks”, European Cooperation for Space Standardization (ECSS), 2003, 124 p.
[2] ECSS-E-ST-50-53C “SpaceWire - CCSDS packet transfer protocol”, European Cooperation for Space Standardization (ECSS), 2010.
[3] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman, Linux Device Drivers, Third Edition - O'Reilly Media, Inc., 2007
[4] Klaus Wehrle, Frank Pahlke, Hartmut Ritter, Daniel Muller, Marc Bechler, The Linux Networking Architecture Design and Implementation of Network Protocols in the Linux Kernel – Pearson Education Deutschland GmbH, 2006