Effective, Robust TCP/IP over SpaceFibre

Albert Ferrer Florit
STAR-Barcelona S.L.
Barcelona, Spain
albert.ferrer@star-dundee.com

Marti Farras Casas
STAR-Barcelona S.L.
Barcelona, Spain
marti.farras@star-dundee.com

Steve Parkes
STAR-Dundee Ltd.
Dundee, Scotland, UK
steve.parkes@star-dundee.com

***Abstract*— Spacecraft data-handling systems continue to demand ever-increasing performance. The use of Commercial Off-The-Shelf (COTS) devices, screened for space applications, is increasingly common, because of the performance and cost advantages they offer.**

SpaceFibre is the next generation of the widely used SpaceWire network technology. SpaceFibre provides multi-Gbit/s on-board networking over both copper and fibre optic cables. SpaceFibre's built-in quality of service (QoS) and fault detection, isolation, and recovery (FDIR) capabilities deliver high reliability and availability which are critical for spacecraft operations. SpaceFibre was specifically developed for space applications, providing the capabilities needed for space systems. COTS processors do not support SpaceFibre but do generally have PCIe and Ethernet interfaces, which could be used to connect to SpaceFibre.

This paper describes how SpaceWire and SpaceFibre networks can be used with both COTS and radiation-tolerant SoC devices, including the use of a TCP/IP stack. It further demonstrates how commercial hardware and standard software network stacks can be combined with the enhanced capabilities of SpaceFibre technology, including Quality of Service (QoS), link reliability and multi-lane redundancy.

Keywords— SpaceFibre, On-board Networking, On-board Data Handling, Ethernet, TCP/IP, SpaceWire, Software Device Driver, System-on-Chips, Embedded Systems

I. Introduction

Spacecraft data-handling systems continue to demand ever-increasing performance. The use of Commercial Off-The-Shelf (COTS) devices, screened for space applications, is increasingly common, offering the higher performance of the latest commercial processors. In parallel with this, differing approaches to risk have also been developed, spurred on by CubeSat developers. Spacecraft-level redundancy, single-string, unit level redundancy and redundancy inside a unit, are all available to help balance cost of development/deployment against risk/mission-life. Space Enhanced Plastic (SEP) components have been developed in response to the use of COTS, to enable a better balance between reliability, cost and performance. Very high performance FPGAs like the Versal have been developed for space applications to support demanding processing and data-handling requirements. However, the drive to use COTS for space applications remains strong because of the performance and cost advantages they offer.

Reliable and efficient real-time performance remains paramount in space. Traditional Ethernet does not inherently guarantee deterministic latency or robust fault-tolerance; it typically depends on additional layers (e.g., TCP, additional redundancy mechanisms) to manage errors and congestion. Many COTS devices (e.g., computers running Linux) rely on the familiar TCP/IP stack and do not implement SpaceFibre links [1].

By creating a bridge between Ethernet (commonly used by COTS devices) and SpaceFibre (readily adopted by space-qualified FPGA designs), engineers can deploy a single, integrated network with minimal duplication of cables or protocol infrastructure.

Previous work has focused on encapsulating transport protocols over SpaceWire [2] and SpaceFibre [3]. Instead of encapsulating IP packets via an IP gateway, the proposed solution encapsulates complete Ethernet frames within SpaceFibre packets (and vice versa), with IP packets, when present, remaining inside the Ethernet payload. This approach supports use cases that do not rely on the IP protocol and makes the implementation simpler and more efficient, since no IP layer processing is required and the entire forwarding path can be easily implemented in hardware to sustain very high data rates. A similar approach is used by Ethernet over InfiniBand (EoIB) [4].

In this paper, we present a practical implementation of this concept, bridging Ethernet and SpaceFibre directly at data link layer to support both raw Ethernet applications and the TCP/IP stack of COTS and radiation-tolerant SoC devices. At the software level, one or more virtual network interfaces expose SpaceFibre and Ethernet endpoints to the operating system standard TCP/IP stack as an IP network organized into two subnets: one dedicated to native SpaceWire/SpaceFibre traffic (e.g., RMAP) and another to standard IP traffic (e.g., TCP).

Either subnet may span SpaceFibre and Ethernet links, as is the case with Wi-Fi/Ethernet bridging [5]. However, broadcast is intentionally unsupported for simplicity and robustness; SpaceFibre excludes packet broadcasts to prevent broadcast storms in redundant topologies, so IP addressing is configured statically, and the bridging logic behaves as a constrained Layer-2 gateway rather than a transparent Layer-2 bridge.

When Ethernet endpoints are present, this software abstraction is complemented by hardware support, using a SpaceFibre Routing Switch extended with Ethernet ports and bridging logic, as described in the next section.

II. SpaceFibre Routing Switch with Ethernet Ports

The STAR-Dundee SpaceFibre Routing Switch [6], enhanced with Ethernet ports (SpFi Router), serves as the hardware backbone of the proposed unified network. The Ethernet ports bridge Ethernet and SpaceFibre networks, providing effective tunnelling of Ethernet frames over SpaceFibre, and vice versa, at the data link (Layer-2) level.

The requirements for this bridging logic can be derived from the analysis of data flows among COTS processors, SoC FPGAs, and standalone FPGAs. A simple topology is shown in Figure 1. To support these data flows, the Ethernet ports of the SpFi Router must be capable of supporting two distinct classes of packet payloads: conventional IP-based Ethernet traffic and native SpaceFibre traffic.

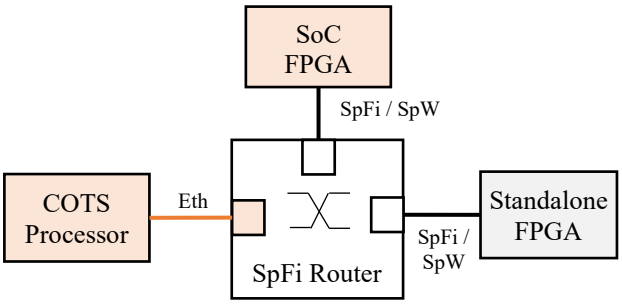

Fig. 1. SpaceFibre Routing Switch with Ethernet ports

A. Supported Packet Payloads

As stated, the unified network must transparently transport both Ethernet frames carrying IP packets, and SpaceFibre packets carrying native protocols such as RMAP or raw payload data.

1) IP-based traffic

In our solution, Ethernet frames are always used to carry IP packets with TCP/UDP or other transport protocols. The corresponding data flows in Figure 1 are:

a) From COTS processor to the SoC FPGA:

The Ethernet frames from the COTS processor are encapsulated into SpFi packets by the Ethernet port of the SpFi Router. At the destination, the SoC FPGA decapsulates them into Ethernet frames, which are then processed by the standard OS network stack.

b) From SoC FPGA to COTS Processor:

The network interface of the SoC FPGA sends SpaceFibre packets containing Ethernet frames. The SpFi packets are routed to the Ethernet port, which decapsulates the Ethernet frames, so they can be forwarded to the COTS Processor.

2) Native SpaceFibre traffic

SpaceFibre packets are used for raw data or other protocols, such as RMAP packets, which can be processed by hardware:

c) From COTS processor to standalone FPGA:

The network interface of the COTS Processor encapsulates the SpFi packets into Ethernet frames, which are then decapsulated by the Ethernet port and routed to the destination FPGA.

d) From standalone FPGA to COTS processor:

The SpaceFibre packets arrive at the Ethernet port, which encapsulates them into Ethernet frames. The network interface of the COTS processor obtains the application data, e.g., RMAP read reply, from the encapsulated SpFi packets.

e) From SoC FPGA to standalone FPGA and vice versa:

SpaceFibre packets are not encapsulated and follow standard SpaceFibre addressing, but the user application within the SoC FPGA can access the SpFi network using standard IP addressing with UDP sockets.

The following sections describe the functionality implemented in the Ethernet ports in order to support these data flows.

B. Ethernet-to-SpaceFibre

When an Ethernet frame arrives at an Ethernet port of the SpFi Router, the Ethertype field is inspected. If it denotes a SpaceWire/SpaceFibre fragment, this frame and any related frames with more fragments are reassembled into a complete SpFi packet, including SpFi addressing. Otherwise, the full frame is encapsulated in a SpFi packet (Figure 2). The Ethernet PCP field of the VLAN tag is translated into a SpaceFibre Virtual Channel number (VC), which is then mapped to a Virtual Network (VN) depending on the SpFi Router configuration. This provides SpaceFibre QoS to Ethernet frames.

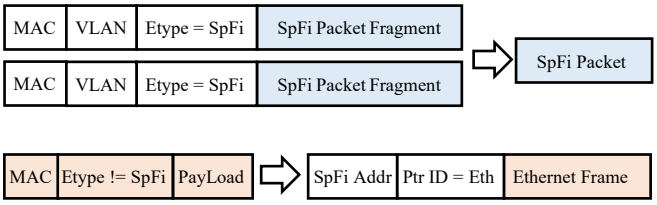

Fig. 2. Ethernet to SpaceFibre bridging depends on EtherType.

C. SpaceFibre-to-Ethernet

Conversely, when a SpFi packet is routed to an Ethernet port, the SpFi Protocol ID field is inspected. If it indicates an Ethernet frame, the Ethernet frame is extracted. Otherwise, the SpFi packet is encapsulated within one or multiple Ethernet frames with the Ethertype field indicating they contain a SpFi packet fragment (Figure 3).

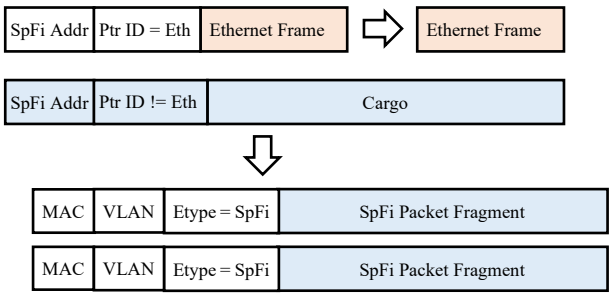

Fig. 3. SpaceFibre to Ethernet bridging depends on SpFi protocol ID.

Table 1 describes the fields of a SpFi packet fragment in least-significant-byte-first order. The "Start Flags" and the "Sequence" fields detect if an Ethernet frame was lost, providing data integrity by truncating an outgoing SpFi packet with an EEP. The "Length" field is used only when padding is required due to the payload length being smaller than the minimum Ethernet frame, after accounting for header bytes.

TABLE I. SPFI PACKET FRAGMENT FIELDS

Field	*Bits*	*Description*
Start Flags	8	Bit 0 is set if it is the first fragment.
Length	8	Payload data length when padding is used, otherwise set to zero.
Sequence	32	Sequence number incremented for every fragment sent.
Payload	variable	SpW/SpFi packet data
End Flags	8	Bit 0 set if fragment ends with EOP Bit 1 set if fragment ends with EEP

D. MAC Address translation

The SpFi Router translates destination MAC address to destination SpFi logical address (LA) and vice versa, either through direct mapping between a MAC address byte to the SpFi address byte or via a configurable look up table.

Since MAC values can be configured in most systems, we adopted the former approach for simplicity, using the format "`02:00:XX:XX:LA`". The first byte "0x2" enforces unicast traffic and sets the locally administered (private) bit, ensuring no collision with globally assigned MAC addresses. The last byte corresponds to the SpaceWire logical address. The "`XX`" fields are don't care values. For source MAC addresses, the second byte is always zero, but for destination MAC addresses it may be assigned to identify a specific Ethernet port when multiple ports are connected to the same switch. This byte is cleared by the Ethernet port during frame forwarding.

For the encapsulation of SpFi packets into Ethernet frames, the source MAC address is obtained from a configurable value assigned to a specific SpFi Router Ethernet port. The system's network management can statically or dynamically configure these mappings.

E. Network Topology

For the deployment of typical space applications requiring redundant paths, the Ethernet ports of the SpFi Router should be connected only to endpoints and not to Ethernet switches. Ethernet networks are susceptible to broadcast storms unless complex managed switches implement additional control protocols. In contrast, SpaceFibre uses broadcast messages with a built-in mechanism that prevents broadcast storms. SpaceFibre does not allow packet broadcast, and its use of static routing tables inherently prevents packets from looping indefinitely.

As the on-board network topology is typically predetermined, ARP and similar protocols are not implemented in the SpFi Router's Ethernet ports, reducing complexity. Instead, the software network interface can either load a preconfigured IP and MAC address setup information or discover the network using configuration data from the SpFi Router configuration port zero.

F. Implementation Results

The previous bridging rules between Ethernet and SpaceFibre were designed to minimize implementation complexity while maximizing performance. Table 2 presents the resource usage of the Ethernet port bridging logic implemented on a radiation-tolerant PolarFire FPGA (RTPF500ZT) without including an Ethernet MAC IP. This implementation supports all previously described features, including fragmentation and reassembly of SpaceFibre fragments. The achieved maximum operating frequency of 200 MHz enables support lane rates exceeding 6.25 Gbps.

TABLE II. RESOURCE USAGE OF ETHERNET PORT BRIDGING LOGIC

	DFF	***LUT***	***RAM***
RTPF500ZT	789 0.17%	2845 0.57%	6 0.4%

Figure 4 shows the hardware setup used to verify that no bottlenecks are present in the implemented bridging logic. The Ethernet MAC IP used by the Ethernet port of the SpFi Router was replaced with a loopback logic. A STAR-Dundee STAR-Ultra Interface PCIe EGSE unit was used to send SpFi packets containing either Ethernet frames or raw data.

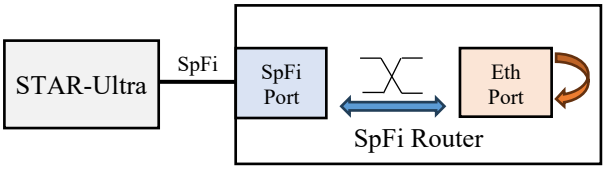

Fig. 4. Hardware setup of the performance test.

Figure 5 shows the data rate achieved when the STAR-Ultra is sending raw data using 3 KB packets with a single-lane SpFi interface. The measured throughput of 4.4 Gbps is very close to the theoretical maximum of 4.77 Gbps for a 6.25 Gbps lane rate after accounting for SpFi encoding and protocol overhead. In this configuration, the Ethernet port is generating Ethernet frames containing SpFi packets and then decapsulating them back into SpFi packets.

Fig. 5. Performance test results at 6.25 Gbps SpFi lane rate.

When transmitting SpFi packets carrying Ethernet frames, the measured data rates remain nearly identical. In this case, the Ethernet port first decapsulates the frames and subsequently re-encapsulates them into SpFi packets.

III. VIRTUAL NETWORK INTERFACES

For SoC-based endpoints, commercial processors, or other software-driven systems connected via SpaceFibre or Ethernet, we implemented virtual network interfaces that allow user applications to exchange both IP-based traffic and native SpaceFibre packets through standard TCP/UDP sockets. From the application's perspective, communication occurs over a standard IP network, while the virtual network interface directs the traffic onto Ethernet or SpaceFibre packets. This allows existing applications to use standard networking APIs without modification.

A virtual network interface is a software construct that sits directly beneath the operating system's (OS) networking stack. Each virtual interface is configured with an IP address (or subnet) and a maximum transmission unit (MTU). The MTU defines the maximum packet size supported by the interface; if a packet exceeds this value, the kernel automatically fragments it before transmission. Likewise, fragmented packets are reassembled by the OS networking stack before delivery to the application, ensuring transparency for higher-layer protocols and applications.

When an application opens a TCP or UDP socket, the OS stack generates the corresponding IP packets and forwards them to the appropriate virtual interface, depending on the source and destination IP address of the socket configuration.

The interface encapsulates these IP packets into link-layer packets and transmits them across either Ethernet or SpaceFibre links. In the reverse direction, received packets are decapsulated, reconstructed as IP packets, and injected back into the OS stack.

A virtual network interface can be realized through different mechanisms, ranging from user-space interfaces (e.g. TUN interface) to fully integrated kernel-level drivers, depending on performance and deployment requirements.

Figure 6 shows the network stack supporting the proposed unified network architecture. Applications operate over distinct IP subnets depending on whether they generate standard IP traffic or native SpFi traffic, which are managed by different types of network interfaces. Also, the virtual network interface is configured to either use a SpFi or Ethernet hardware device driver. To support SpFi QoS, a dedicated virtual network interface can be assigned to each VC, or for Ethernet links, to the equivalent PCP field.

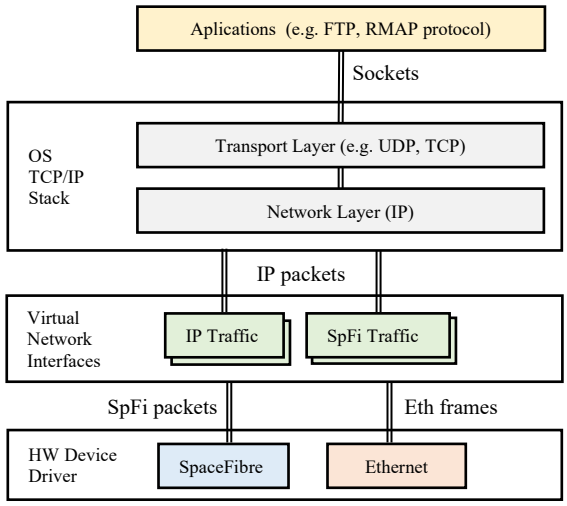

Fig. 6. Unified Network Stack

Under the hood, these network interfaces implement among other functions:

- The address translation, so that IP addresses map correctly to MAC or SpFi logical addresses, based on direct translation, static configuration or dynamically using network discovery algorithms. Different subnets can be used for each SpFi virtual network. For simplicity, an IP address can also contain the actual SpFi destination logical address being used.

- For outgoing standard IP traffic, the network interface first encapsulates the IP packets within Ethernet frames. In case of a SpFi link, these frames are then encapsulated into SpFi packets. IP traffic is always encapsulated in an Ethernet frame even when there are no Ethernet links involved. This allows the source of IP packets to be agnostic of the destination using SpFi links or Ethernet.

- For incoming standard IP traffic, a standard Ethernet network interface is used for Ethernet links. For SpFi links, a custom network interface extracts the IP packet from the Ethernet frame encapsulated within the SpFi packet received.

- For outgoing native SpFi traffic, applications use the UDP protocol to send a complete SpFi packet. The network interface extracts the SpFi packet from the UDP payload within the IP packet. For Ethernet links, the SpFi packet is then encapsulated into an Ethernet frame.

- For incoming native SpFi traffic, the SpFi packet is received either directly over a SpFi link or from an Ethernet frame over an Ethernet link. The packet is then encapsulated into a UDP datagram within an IP packet and delivered to the OS networking stack.

This layered architecture integrates all endpoints, including COTS sensors, SoC-based processors, and FPGA-only instruments, into a unified network, allowing each device to operate seamlessly with either standard IP protocols or SpFi protocols tailored for space applications.

IV. Evaluation

To implement and evaluate the overall solution, the network depicted in Figure 1 was built using the following hardware: a PC equipped with an Ethernet card emulates the COTS processor; another PC with a STAR-Dundee STAR-Ultra SpFi interface emulates the SoC FPGA; and a VCK190 Evaluation Kit with a Versal FPGA [7] implements the upgraded STAR-Dundee SpFi Router IP with an Ethernet port. The standalone FPGA shown in Figure 1 is expected to handle native SpFi protocols, for example, a mass-memory unit processing RMAP commands in hardware. For simplicity, this use case is evaluated by just reading the routing table of the SpFi Router using RMAP read commands.

A. TUN interfaces

The virtual network interfaces are implemented as TUN interfaces. There is a TUN interface per VC for SpFi endpoints and one per PCP value for Ethernet endpoints. The OS provides a TUN kernel driver that exposes IP packets from the network stack through read and write file descriptors. The functionality of each virtual network interface is implemented in a user-space process that accesses these file descriptors. This approach greatly simplifies the implementation, although it may not achieve the performance of a kernel-space network driver [8].

B. IP Address translation

In this implementation, a direct translation is used between IP, MAC and SpFi logical addresses, using IP addresses with the following format:

IP traffic: `10.100.<VN>.<LA>`

SpFi traffic: `10.100.<VN+128>.<LA>`

Where `<VN>` is the SpFi virtual network number (VN) and `<LA>` is the SpFi destination logical address. For SpFi traffic using path addressing, `<LA>` is set to zero.

Each endpoint in the unified network is assigned a unique SpFi LA, even for endpoints connected through Ethernet links. This LA serves as a common identifier across all network layers. It is embedded in the MAC address of Ethernet endpoints (using the format defined in Section II-D) and, together with the VN, determines the IP address assigned to the TUN interface of each virtual channel or PCP.

C. *Example Results*

Figure 7 shows a screenshot of the STAR-Ultra Link Analyser capturing an IP packet received from a SpFi link. The IP packet was generated by an endpoint with a standard Ethernet interface and was encapsulated within a SpFi packet by the Ethernet port of the SpFi Router. The packet begins with SpFi LA value 0x29, followed by the SpFi protocol ID byte indicating that the SpFi packet contains an Ethernet frame (value 0xEE is used in our implementation). After the Ethernet and IP headers, the actual IP payload is highlighted.

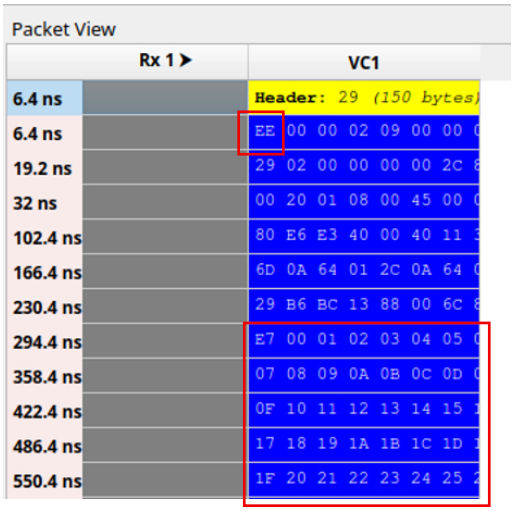

Fig. 7. IP packet received by a SpFi link

We have previously described how virtual network interfaces enable the use of UDP to send native SpFi traffic via either SpFi or Ethernet links. Figure 8 shows a simple Python script that sends a UDP packet with an RMAP read command. The SpFi path address is included in the UDP payload, so the destination IP address is used only to select the interface corresponding to the specified VN.

```
import socket

SRC_IP   = "10.100.128.44" # SpFi Traffic on VN0
DST_PORT = 0 # UDP port is not used

PATH_ADDR = 0 # SpFi Path Address 0 (Router config port 0)
DST_IP   = "10.100.128.0" # LA not used.
RMAP_rd = [254,1,73,32,0,0,0,1,254,0,0,0,0,0,64,2,0,0,4,138]
packet = bytes([PATH_ADDR] + RMAP_rd)

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_PRIORITY, 1)
s.bind((SRC_IP, 0))
s.sendto(packet, (DST_IP, DST_PORT))
```

Fig. 8. Python script to send an RMAP packet using UDP

Figure 9 shows a screenshot of the Wireshark software with a UDP packet received over an Ethernet link, containing the corresponding reply packet from the SpFi Router configuration port. The data field begins with the protocol ID byte 0x1, indicating an RMAP packet. The RMAP command reads a 32-bit register with the ASCII value “SpFi”, as shown in the highlighted RMAP data field.

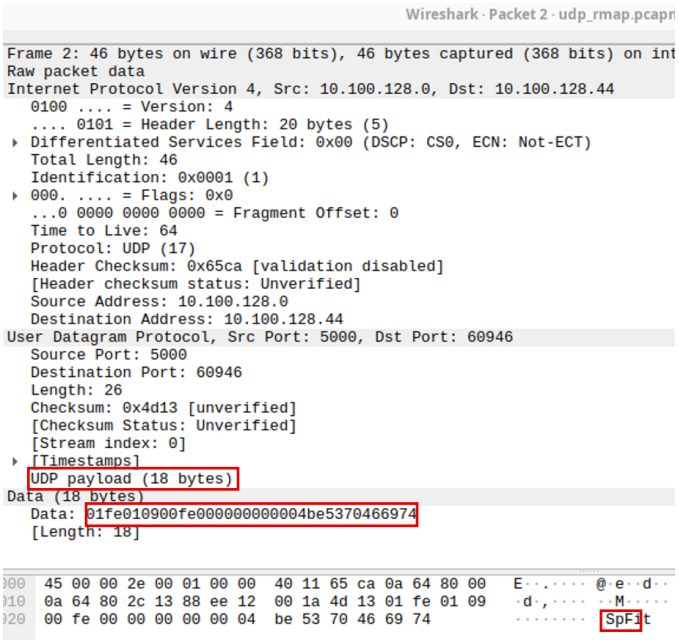

Fig. 9. UDP packet with RMAP reply data

Finally, Figure 10 shows a file transfer between two endpoints, across a network with both SpFi and Ethernet links. A standard FTP application is used, highlighting the advantages of integrating the standard OS network stack into SpFi networks.

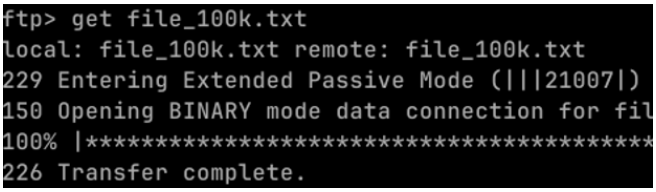

Fig. 10. FTP file transfer across SpFi and Ethernet links

D. *Future work*

The evaluation was performed using Gigabit Ethernet MACs and TUN virtual interfaces. However, the Ethernet port of the SpFi Router supports much higher speeds, and the MAC is currently being upgraded to 10 GbE. Future work will focus on the development of kernel-space network drivers to achieve maximum throughput and lower latency, building on our previous work on high-performance driver architectures [9].

V. Conclusions

In contrast to Ethernet related protocols such as TSN and TCP, SpaceFibre links can operate without software intervention, are not constrained by a maximum packet length and provide a reliable link that transparently mitigates single-event effects produced by radiation events [10]. The flexibility of the SpaceFibre packet format and the robustness of its broadcast mechanism enable custom redundancy mechanisms and precise time distribution without the risk of broadcast storms that can occur in Ethernet networks [11,12]. The proposed solution allows SpaceFibre technology to integrate COTS devices with Ethernet interfaces and leverage the standard TCP/IP stack to deliver even more capabilities.

REFERENCES

[1] ECSS, *SpaceFibre – Very High-Speed Serial Link*, ECSS Standard ECSS-E-ST-50-11C, Issue 1, European Cooperation for Space Data Standardization, May 2019. [Online]. Available: http://www.ecss.nl

[2] S. Mills and S. Parkes, "TCP/IP over SpaceWire," in *Proc. DASIA 2003 – Data Systems in Aerospace*, vol. 532, 2003.

[3] D. Dymov, Y. Sheynin, and V. Olenev, "STP-ISS transport protocol application for SpaceFibre on-board networks," in *Proc. 7th Int. Conf. Control, Decision and Information Technologies (CoDIT)*, vol. 1, IEEE, 2020.

[4] E. Smirnov and M. Sennikovsky, "Ethernet over InfiniBand: Current solutions," in *Proc. 14th Annu. OpenFabrics Alliance Workshop*, Apr. 2018.

[5] IEEE Computer Society, *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*, IEEE Std 802.1D™, 2004.

[6] A. Gonzalez Villafranca, A. Ferrer Florit, M. Farras Casas, and S. Parkes, "SpaceFibre IP cores for fast adoption of next-gen FPGA communication architectures," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, USA, Mar. 2025.

[7] AMD, *AMD Versal Adaptive SoCs*. [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal.html

[8] P. Emmerich, M. Pudelko, S. Bauer, S. Huber, T. Zwickl, and G. Carle, "User space network drivers," in *Proc. ACM/IEEE Symp. Architectures for Networking and Communications Systems (ANCS '19)*, 2019.

[9] D. Gibson, S. Mills, A. MacLennan, and S. Parkes, "Efficient high data rate networking using remote direct memory access over SpaceFibre," in *Proc. Eur. Data Handling & Data Processing Conf. (EDHPC)*, Juan-les-Pins, France, 2023, pp. 1–5, doi: 10.23919/EDHPC59100.2023.10396179.

[10] A. Gonzalez, A. Ferrer, M. Farras, S. Parkes, P. Maillard and K. O'Neill, "Characterisation of AMD Versal FPGA Transceivers Under Heavy-Ion Radiation," 2025 25th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Antwerp, Belgium, 2025.

[11] S. Parkes, A. Ferrer, A. Gonzalez, and D. Gibson, "SpaceFibre onboard interconnect: From standard, through demonstration, to space flight," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, USA, Mar. 2025.

[12] "Performance evaluation of Ethernet LAN broadcast and point-to-point," *Comput. Commun.*, vol. 10, pp. 290–298, 1987, ISSN 0140-3664.