Technical studies for operations with real-time communications in robotic missions

Daniel Weber[*] , Rossella Falcone[†] , Marcin Gnat[‡] , Armin Hauke[§] and Felix Huber[¶]

DLR, Oberpfaffenhofen, Bavaria, 82234, Germany

Robotic telepresence operations between earth and space are of high research value for science as they enable operators on ground to perform physical tasks in space without the need of human presence. Real-Time telepresence with haptic-feedback and stereoscopic imaging, however, poses new requirements to physical parameters of the communication channel like loss, delay and jitter as well as to the protocols spoken between the participants. To meet the new requirements, past robotic missions like ROKVISS chose to use specialized and dedicated communication channels while bypassing the established ground station network infrastructure. However, performing robotic and standard TM/TC operations in parallel was impossible because the Space Link could only be locked by either of the communication chains. For future missions, we present a setup that multiplexes robotic science data and standard TM/TC into one physical channel. Real-time requirements are met because the setup makes use of several FPGAs that forward UDP packets in synchronization with a common master clock. We present test results and test measurements of this technology and compare the proposed setup to a software based solution. Furthermore we present general approaches, tools and techniques for real-time related tasks. Finally we discuss the use of Space Link and Space Link Extension protocols in the communication chain and their impact on the real-time requirements. Operational aspects of the new setup and protocols are discussed as well.

I. Introduction

One of the most important challenges for future spacecraft operations in low and geostationary earth orbit will be the removal and avoidance of space debris. It has been estimated that during the next 100 years the number of debris particles larger than 10 cm will more then double. The number of debris particles larger than 1 cm will even increase by a factor of five. If no countermeasures are taken, the usage of earth orbits for satellite missions will become impossible.[1,2]

As this poses a serious threat to the future of space flight, it will be necessary to actively deorbit space debris or to move it to a graveyard orbit. By doing so, the number of dangerous particles can be reduced to a number at which spacecraft operation will still be possible. One of the discussed ways to accomplish this task is the usage of robots. They could either perform servicing of satellites to increase their lifetime or they could deorbit space debris to clear near earth orbits.[3–5]

Promising candidates for this task are telerobotic missions because no autonomous logic has to be installed onto a satellite. Instead they enable a human operator on ground to take decisions in real-time. Experiments have already demonstrated the feasibility of telerobotic operations in low earth orbit. Recent experiments of the ESA Telerobotics Laboratory even suggest that telerobotic operations are possible at time delays of several seconds when using model-mediated control approaches. This would enable telerobotic operations at even higher distances or under more difficult communication conditions.[6–8]

Unfortunately all the experiments make use of custom specialized setups which bypass the established world-wide network of ground station interfaces for satellite operations. Currently there is no standardized

[*]Software Engineer, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Daniel.Weber@dlr.de
[†]Software Engineer, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Rossella.Falcone@dlr.de
[‡]Ground Data Sys. Manager, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Marcin.Gnat@dlr.de
[§]Software Group Manager, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Armin.Hauke@dlr.de
[¶]Institute Director, Space Operations and Astronaut Training, DLR Oberpfaffenhofen, Germany, Felix.Huber@dlr.de

interface available for performing telerobotic operations with any ground station in the world. However, such an interface will play a major role in case of increased and regular activities with telerobotic missions. In particular it will be mandatory to do station handovers during telerobotic missions to increase the duration of telerobotic operations.[6, 9, 10]

The ROKVISS mission for example used a specialized hardware to generate a signal that was modulated onto a radio frequency link. It also used a dedicated leased SDH communication line to a near ground station. The round-trip delay to the ISS was in the range of $20 - 30$ ms which is a very good value. However this very good value came at the cost of bypassing the ground station network and creating a specialized setup only usable with one specific ground station, one hardware device and one dedicated SDH communication line. Furthermore this solution occupied the entire space link to be only used by the robotic telepresence operation. Housekeeping TM/TC had to be transferred to and from the ISS via a separate space link. Due to the lack of a world-wide interface also no ground station handover was possible.[6]

Kontur-2 for example made use of the NASA DTN (Delay Tolerant Network) of the International Space Station. This helped to evaluate how network technology can be used for the purpose of telerobotic operations. However DTN networks are optimized and designed for overcoming disruptions in connectivity. Thus they bundle network packages and use a store and forward mechanism to overcome connection interrupts. Both mechanisms create unpredictable delay and jitter and therefore are not a good choice for real-time applications. The solution bypasses the world-wide ground station interfaces and is a specialized solution for the ISS only. Results showed delays of lower than 750 ms in 95 % of the cases, but revealed problems with higher delays at around 5 % of the cases due to the DTN technology. Also this approach suffers from the low bandwidth that is limited to 100 bps. It is not possible to transfer video streams for missions combining stereoscopic imaging with robotic telepresence operations.[9]

As can be seen, these solutions are highly specialized and the question arises how a general world-wide interface to ground stations for telerobotic missions must look like. It has to be evaluated whether it is better to use hardware based approaches like in ROKVISS or to use software based approaches like in Kontur-2. Furthermore it has to be investigated how a real-time protocol can be integrated into the existing protocols and whether the ground station hardware needs to be changed for this purpose. In particular the space link must be able to support both standard TM/TC and real-time science data in parallel. Thus it has to be analyzed how a multiplexing between both data streams can be performed.

To address these problems a setup for the DEOS mission will be analyzed that includes several FPGAs. The FPGAs perform multiplexing of robotic science data with housekeeping TM/TC. Furthermore a software system will be set up that shall perform the same task with standard Linux computers. The results will be compared with each other. By doing so it will be possible to deduce the properties of a future interface to all ground stations for robotic telepresence operations.

In addition different tools and techniques are being developed that are used in real-time related application testing. This includes a WAN-Simulator that is developed to simulate different types and values of delay, loss and jitter and to simulate the effects of a space link. Also the real-time behaviour of standard Linux systems is discussed more deeply to achieve a better understanding for future applications.

II. Comparison of a FPGA- and a software-based design to implement a ground station interface for real-time telerobotic missions

II.A. A FPGA based approach

Fig. 1 shows a setup proposed for the DEOS mission.[11] In order to avoid the use of two separate space links, real-time science data is multiplexed with standard housekeeping data. To do so a FPGA based approach is used. Dedicated hardware FPGAs enable full timing control over the setup. This is supposed to result in good communication parameters.

II.A.1. Uplink

In order to merge real-time science TC Frames and standard housekeeping TC Frames into a single communication chain, the setup in Fig. 1 makes use of a FPGA called "H/W Merger". The TC Frames are encapsulated into CLTUs and are sent to the "H/W Merger" via UDP. The "H/W Merger" alternatingly receives these UDP packets at two different network interfaces. In the setup in Fig. 1 either CLTUs from the standard Spacecraft Operating System (SCOS) or the Payload Control System (PCS) are taken and merged

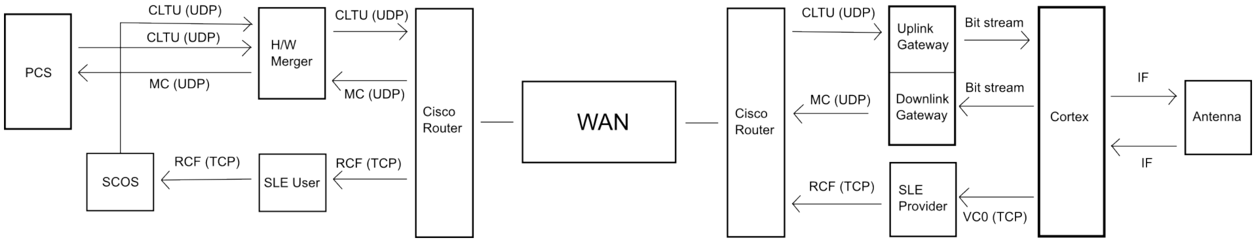


Figure 1. CLTUs from PCS or SCOS are merged at the "H/W Merger" (FPGA) into a UDP data stream. The UDP packets are forwarded to an Uplink Gateway (FPGA) and are modulated onto the Intermediate Frequency (IF). Telemetry is encapsulated into UDP packets by a Downlink Gateway (FPGA) and sent back to PCS. In parallel telemetry can be received via a SLE User/Provider setup.

into the uplink data stream. The output of the "H/W Merger" consists of UDP packets containing CLTUs that are sent at a fixed interval of 2.5 ms. They are forwarded through a Wide-Area-Network (WAN) to the ground station. A FPGA Uplink Gateway at the ground station receives the UDP packets and creates a bit stream that is directly modulated onto the space link.[12]

II.A.2. Downlink

The telemetry data consists of two parts. On the one hand there is real-time science data from the robot, on the other hand there is housekeeping TM data. To enable the Payload Control System (PCS) to get the real-time data as fast as possible a FPGA is used as a Downlink Gateway. It generates UDP packets containing the TM Frames of the Master Channel (MC) and forwards them directly to the "H/W Merger". The "H/W Merger" sends the data back to the PCS. In parallel a standard SLE User/Provider setup is used for operations with housekeeping TM Frames. To receive the telemetry information the Return Channel Frame Service (RCF) of SLE can be used.[13]

II.A.3. Discussion of the FPGA based approach

An important design choice of this setup is the use of FPGA devices. One at the control center and two at the ground station side. An idea is to synchronize the devices to a common GPS clock in order to reduce the jitter of the setup to a very low amount. Problems of this setup however arise as the "H/W Merger" is located at the sender side and it is a FPGA device. That means that each user of a robotic mission would have to install such a device. Furthermore the "H/W Merger" in this design will not create accept and transmit acknowledges like in a SLE communication chain. The usage of the Uplink/Downlink Gateway FPGA devices is a good choice as the direct modulation of a real-time signal onto the RF-Link bypasses possible delays by the Cortex.[14] On the other hand one might think of a solution that is based on an approach only using software like it will be discussed in the next section.

II.B. A software based approach

The software based approach benefits from the existing ground station network infrastructure. It minimizes the installation of additional specialized FPGAs. Thus the setup is more flexible and easier to install at ground stations worldwide. This is of high importance as successive handovers to other ground stations are necessary when doing telepresence operations over longer periods of time.[15]

II.B.1. Uplink

TC Frames from the Payload Control are packed into CLTUs. The CLTUs are forwarded to the Mission Control Center Interface (MCCI) via UDP. The MCCI consists of a SLE User and an additional interface for UDP communication. Instead it could also consist of an updated version of SLE that supports UDP communication. At the MCCI, the UDP packets are forwarded to the Ground Station Interface preferably by a thread running with real-time priority. A thread also running with real-time priority receives the UDP packets and creates data that is to be modulated onto the space link. In parallel the Housekeeping TC will be transferred via the existing SLE Protocol via TCP/IP.

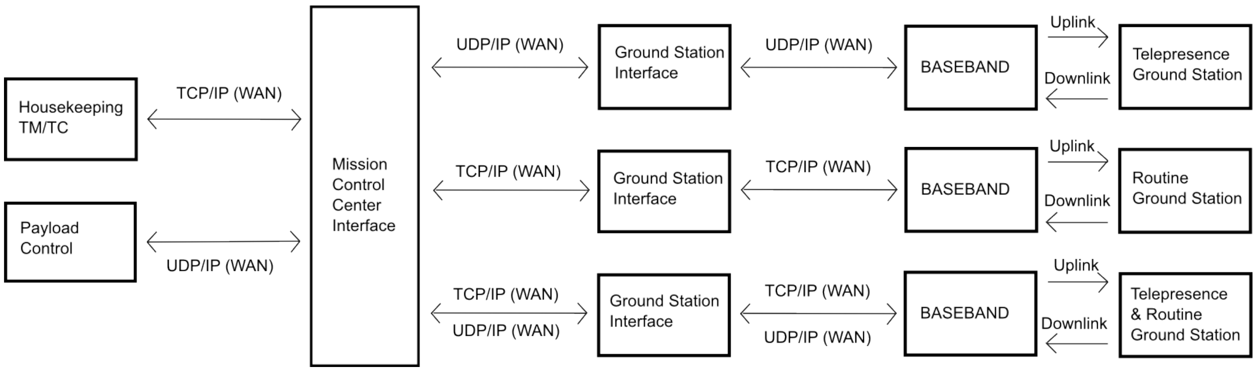


Figure 2. A proposed software based setup for robotic telepresence missions including station handovers.[15] Housekeeping TM/TC is transferred via the established SLE User/Provider Scheme which is based on TCP/IP. Real-time science data from the Payload Control is sent via UDP/IP either via a separate interface or via an updated version of SLE.

II.B.2. Downlink

In case of housekeeping telemetry, the TM Frames are received via the established SLE communication chain. Thus the interface between the Ground Station and the Mission Control Center is a SLE User / Provider setup. However, an additional interface must be implemented for the reception of real-time science data. This can be either an extension of the existing SLE protocol with a UDP interface or it can be a separate UDP interface. In any case the UDP packets are generated by real-time threads for increased performance.

II.B.3. Discussion of the software based approach

All available SLE applications basically use the TCP/IP protocol which inherently can be a source of jitter due to retransmissions. Thus the setup adds a UDP communication chain to the SLE User/Provider scheme. In general UDP is connectionless and there are no acknowledges. This ensures a low delay and a low jitter. As there is no information on how many packets are delivered or lost, an additional TCP/IP chain could be set up that informs the user about the quality of service.

II.C. Discussion of both setups

In the software based setup there are no FPGAs present that ensure a certain defined timing and they cannot be synchronized to a common GPS signal like in the hardware setup. Also it is unknown if timings can be achieved by the software that are comparable to the FPGA setup. In the FPGA based setup one would need to ship a FPGA to every user of the system. Is the possibly better timing worth the effort or is it better to use a software based solution? Is it sufficient to not give any acknowledges and only use UDP or should an additional TCP/IP line give at least some non-real-time information on quality of service? Maybe also the usage of RTP would be a good candidate instead of UDP in order to at least allow retransmissions. In order to draw further conclusions on which system will be the superior choice, further tests will be performed. In the next section, one of the components of the real-time chain — the "H/W Merger" — is compared to a software based version.

III. Timing behaviour of a FPGA based and a software based merger to multiplex housekeeping with real-time science data

In order to compare the timing behaviour of a software based and a FPGA based solution we take the merging functionality as an example and see how well the merging of standard TM/TC and real-time science data can be performed on both systems.

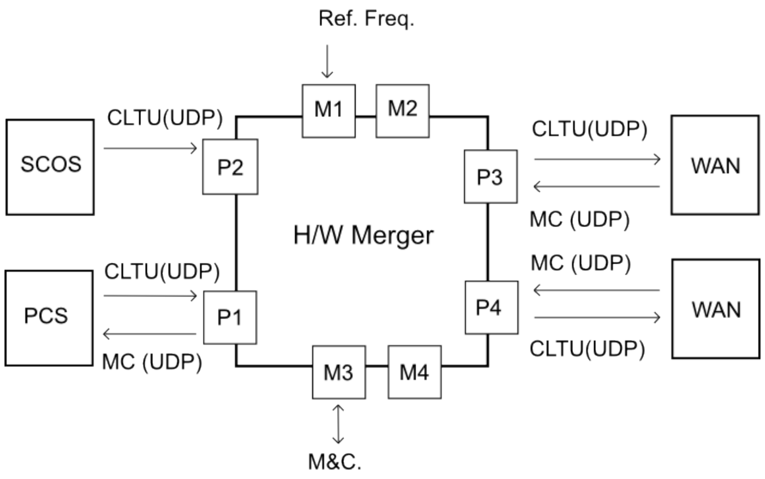


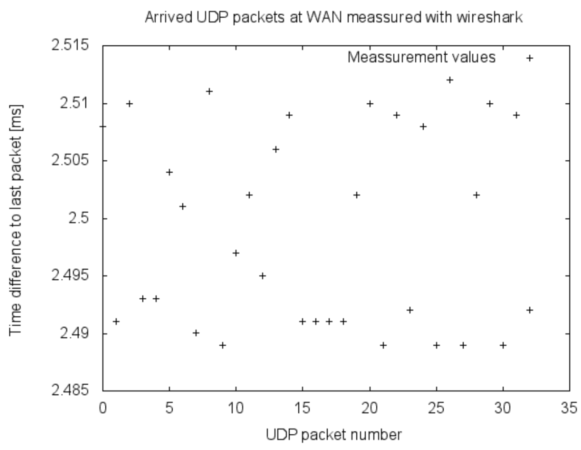
Figure 3. FPGA-Merger that takes incoming UDP packets from the Satellite Control System (SCOS) at P2 and the Payload Control System (PCS) at P1 and merges them into an output data stream at P3. The port P4 can be used as a backup port. Ports M3 and M4 allow the configuration of the device. Port M1 and M2 can be used to connect the device to an external reference frequency.

III.A. Timing analysis for the FPGA based "H/W Merger"

III.A.1. Test setup

The FPGA merger receives UDP packets on the Ethernet ports P1 and P2 and multiplexes them into an Ethernet output at P3. UDP packets generated at P3 have a fixed interval of 2.5 ms between each other. These packets contain CLTUs that are either from P1 or from P2. Further UDP packets are received at port P3 and sent back to the Payload Control System. These UDP packets contain TM Frames received from the satellite.

III.A.2. Timing analysis



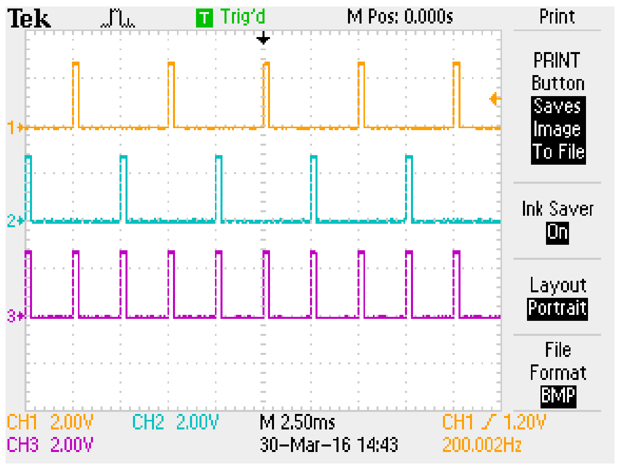


Figure 4. Left: Timing interval in between two successive UDP packets measured at the output of P3A using wireshark. Right: Oscilloscope measurement of the timers at the FPGA. CH3 (pink) is the send timer with an interval of 2.5 ms. CH1 (yellow) and CH2 (blue) go high when data is being copied from the receive buffer of P1/P2 to the send buffer of P3A. It goes low when the copy process finishes. The measurement shows that there is no jitter at the FPGA itself.

The analysis of the 2.5 ms interval shows that this interval can be achieved very precisely. The standard deviation of the 30 measurement values in this case is 2.50 ms $\pm$ 0.01 ms. Oscilloscope measurements in Fig. 4 also show that the FPGA timers work correctly and don't introduce this jitter. Thus the jitter must be introduced by other components.

III.A.3. *Conclusion*

As expected a very good timing can be achieved by the usage of an FPGA device. The jitter is as low as ± 0.01 ms and can probably still be reduced to even lower amounts.

III.B. Timing analysis for the merging performed by software

For a software system a value is expected that performs much worse than a FPGA device. The system runs on a Linux system where a scheduler creates periodic interrupts. The most important question is how the best timing can be achieved and which parameters of a Linux system can be changed in order to achieve a better timing behaviour.

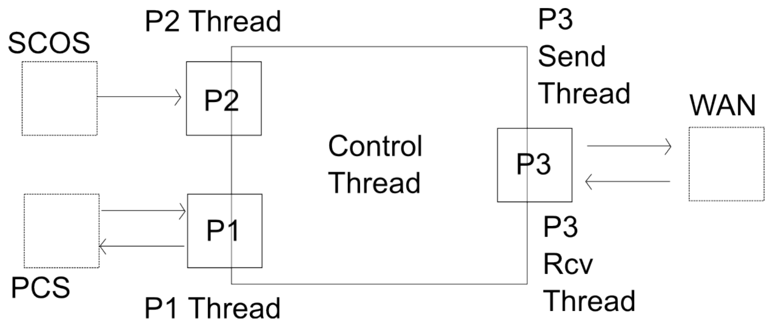


Figure 5. Software-Merger that takes incoming UDP packets from the Spacecraft Operating System (SCOS) at the P2 Thread and that takes incoming UDP packets from the Payload Control System (PCS) at the P1 Thread. The packets are forwarded to a P3 Thread which is a real-time thread on a Linux system to multiplex the incoming packets into a data stream with a fixed time interval of 2.5 ms.

III.B.1. *Test setup*

The software is written using the ACE Framework. It consists of several threads running on a Linux system. The threads have different priorities. The P3 Thread is a Linux real-time thread with the highest priority, whereas the other threads are also real-time threads but have a lower priority. It is important for P3 to have the highest priority as the output shall generate UDP packets at an interval of exactly 2.5 ms like in the setup before. This interval must not be interrupted by any other thread or process.

III.B.2. *Information on Linux for real-time related tasks*

In order to set up the Linux system one has to know several things about the timing behaviour of the system. The timing of the Linux system highly depends on the Linux scheduler that starts and stops processes in order to distribute the CPU resources. There are several kernel parameters that define how the scheduler operates. These parameters are listed in Table 1.

Table 1. Kernel Parameters and their impact on timing

Parameter	Impact
CONFIG_HZ=1000 or 250 or other values	Select the frequency of the kernel scheduler
CONFIG_NO_HZ=yes or no	Select if scheduling intervals may be omitted
CONFIG_HIGH_RES_TIMERS=yes or no	Enable the usage of high resolution timers

It is very important that these parameters are set to appropriate values, otherwise a precise timing behaviour cannot be obtained. The same software running on a system with different Linux kernel parameters will perform differently. The CONFIG_HZ parameter defines a fixed frequency of the Linux scheduler that is used in case CONFIG_HIGH_RES_TIMERS=no and CONFIG_NO_HZ=no. On laptops this is often set to a default of 250 Hz to save battery power and it is often set to 1000 Hz on desktop computers. Thus it will be impossible to send UDP packets on a default laptop in intervals of 1 ms while it is possible to do so on a default desktop PC. By enabling CONFIG_NO_HZ, some scheduling intervals may be omitted, for example in case of an idle CPU. This can contribute to reduce power consumption of a computer. Another important aspect is the support of high resolution timers that are built into most systems today. If they are enabled, the scheduler can make use of them for scheduling purposes and thus generate precisely timed interrupts.

A good setting here is to enable the high resolution timers and to allow the omission of some scheduling intervals by setting CONFIG_NO_HZ to yes. Always make sure that the software is not run on a system with a fixed frequency that is lower than the desired frequency of a task. Another point that has to be addressed is the priority of processes. The priority should be so high that the process cannot be interrupted by other processes. Here the Linux Real-Time-Tasks are used. They are created by using the command *chrt* in front of the name of the command to be executed. However this requires root privileges. When using a RT-Task it can be ensured that the scheduler always prioritizes a given process. Only other RT-Tasks or hardware interrupts are able to interrupt the process.

The Linux kernel can also be patched by a real-time PREEMPT patch. This kernel patch further reduces the latency of the kernel and it even enables to set priorities of hardware interrupts. Thus a process can even be told not to be interrupted for example by a hard drive read. However this is not considered necessary in the tasks performed here. The latencies and jitter introduced by the Linux system would be much lower than any network latencies and jitter. Thus the network application would not benefit from this patch.[16]

III.B.3. Timing Analysis

For the measurements the same setup as in the measurements with the FPGA device is used. The time interval of output UDP packets at P3 is measured with wireshark.

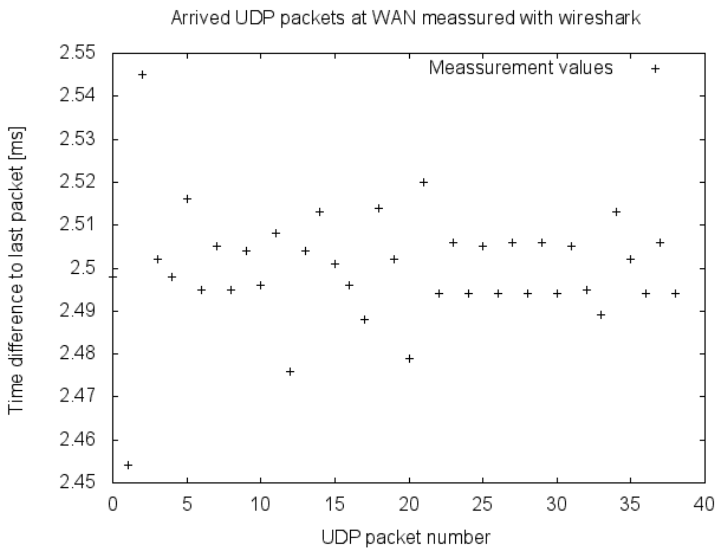


Figure 6. Time interval between two successive UDP packets at the output Thread P3 measured using wireshark.

The results show that the achievable timing is in the same range as the FPGA device. The mean value and standard deviation that can be achieved is 2.5 ms $\pm$ 0.01 ms. However one has to really take care of the correct settings. Real-Time tasks have to be used and also the kernel parameters have to be set accordingly. If one would run the same software on a laptop with CLOCK_HZ = 250, one would only be able to achieve a lowest interval of 4 ms. Also note that if no RT-Tasks would be used, then the software would suffer from scheduling interrupts. During the test and development process, intervals of around 30 ms were observed due to interruptions of the scheduler when no RT-Tasks were used.

III.C. Conclusion

As can be seen, the FPGA technology achieves very precise timing. For tasks like creating a signal to be modulated onto an uplink, the FPGA is definitely the way to go. Thus the usage of FPGAs will be necessary for tasks like implementing an Uplink or Downlink Gateway like in Figure 1. Also the FPGA has a low power consumption (around 5 Watt). Further advantage of the FPGA is that it is inherently safe against hacking as the firmware cannot be changed easily.

However, for a world-wide network interface for future telerobotic missions, a software based approach benefits from its high flexibility. The comparison of the FPGA and software merger has shown that a similar timing can be achieved by the software. A software based solution increases the probability that the existing infrastructure can be used to run the software. Packet inspection and the implementation of protocols is also much easier to be done in software. A software based solution is therefore preferable as a general network interface for future telerobotic missions.

It also has to be kept in mind that successful experiments of the ESA Telerobotics Lab and the Meteron Project show that telerobotic operations can also be performed with latencies of several seconds. Often the benefit of small latency reductions are low compared to the high delay introduced by other components.[8]

In the next section different tools and techniques are shown that are usable for the tests and implementation of a software based system. Furthermore it is necessary to have a look at possible changes of the protocols between ground stations and control centers.

IV. Development of a WAN-Simulator to simulate effects of a wide area network or of an RF link on the communication chain

In order to test the performance of telerobotic operations under different communication conditions there are several techniques that are used by our group.

IV.A. The Linux Traffic Control

The Linux Traffic Control is a subsystem of the Linux kernel that can assign scheduling algorithms to different kinds of network traffic. The traffic that will be affected can be selected by filter rules. For maximum efficiency and timing accuracy, the system is integrated within the Linux kernel. To configure the Traffic Control system different queueing disciplines — called qdiscs — can be set up on a given network interface. These qdiscs are basically schedulers that decide how traffic will be scheduled. Typical queueing disciplines are the FIFO (First-In First-Out) qdisc, the SFQ (Stochastic Fair Queuing) qdisc or a TBF (Token Bucket Filter) qdisc. Queuing disciplines can also be defined to be classfull. Classfull qdiscs can have a filter attached to them that defines which kind of network traffic will be scheduled by the queueing discipline.

One classfull queuing discipline is the netem qdisc. The term netem is an abbreviation for network emulator and it comes shipped with every Linux kernel. The netem queueing discipline can be used to emulate a wide range of communication parameter. Delay or jitter can be defined and packet loss or corruption can be added.[17]

The timing precision of netem is defined by the kernel parameters CLOCK_HZ, CLOCK_NO_HZ and by the parameter CONFIG_HIGH_RES_TIMERS. If for example CLOCK_HZ is set to 250 then only delays in multiples of four milliseconds can be implemented. Thus the kernel parameters play an important role.

In order to develop a tool that simulates characteristics of a wide area network or of a radio-frequency link, one has to alter the communication parameters over time. For example at the end of a satellite pass, the packet loss in the communication chain will increase to 100%. Thus a system has to be implemented that changes the communication parameters over time to simulate different kinds of communication characteristics. In this way telerobotic missions can be simulated in a wide range of communication conditions.

For further information on the Linux Traffic Control system refer to the Traffic Control HOWTO or the Linux Advanced Routing and Traffic Control HOWTO.[18,19]

IV.B. Test setup for simulating different kinds of communication characteristics in telepresence operations

The software that is used on the WAN-Simulator is a software based on the in-house framework for ground station monitoring and control. It can set the queueing disciplines according to the desired simulation parameters via a GUI front-end. The simulation parameters can also be changed over time.

In order to measure the performance of the WAN-Simulator some test measurements are carried out. Only the traffic of the uplink is selected by specific filter rules and queueing disciplines are set up to implement

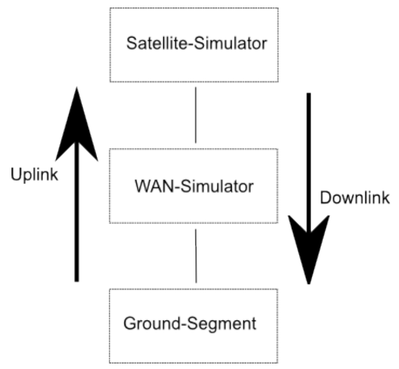


Figure 7. Test setup for telerobotic operations. A satellite simulator acting as the robotic payload is connected to a WAN-Simulator and sends TM Frames to the Ground Segment. The Ground Segment sends TC Frames to the satellite simulator. All Frames are sent via UDP and are routed through the WAN-Simulator.

a given delay on the uplink only. Then network packets from the Ground Segment are sent to the Satellite Simulator. The results depend on the used kernel parameters as shown in Table 2 and Table 3. The results show that for the system using a CLOCK_HZ = 300 configuration, a Round Trip Time (RTT) of around 13 ms is measured if the delay is configured to be 10 ms. The precision of a WAN-Simulator running with high resolution timers is much higher as shown in Table 3. The standard deviation in the measurements is also higher for the system with the lower resolution of 300 Hz.

Table 2. Example RTT measurements with a WAN-Simulator using CLOCK_HZ = 300 **and CONFIG_NO_HZ** = n **and CONFIG_HIGH_RES_TIMERS** = n **.**

Uplink Delay (ms)	#Packets	RTT Min (ms)	RTT Avg (ms)	RTT Max (ms)	RTT Std. Dev. (ms)
10	100	10.7	13.6	32.1	2.4
20	100	20.6	23.6	26.2	1.5
500	100	500.5	505.1	506.5	1.0

Table 3. Example RTT measurements with a WAN-Simulator using CLOCK_HZ = 1000 **and CONFIG_NO_HZ** = y **and CONFIG_HIGH_RES_TIMERS** = y **.**

Uplink Delay (ms)	#Packets	RTT Min (ms)	RTT Avg (ms)	RTT Max (ms)	RTT Std. Dev. (ms)
10	100	10.2	10.4	10.5	0.1
20	100	20.3	20.4	20.5	0.2
500	100	500.1	500.3	500.5	0.7

The same setup is also used to measure the precision of different values of jitter. The results are shown in Table 4.

V. Conclusion

The tests have shown that it is possible to use Linux systems instead of a FPGA to achieve a similar timing performance for the purpose of network traffic altering and generation. This is shown by the implementation of a software based merger and the implementation of a WAN-Simulator.

A future world-wide ground station interface for telerobotic missions should therefore be implemented as a software based interface. A software based solution will be much easier to install at all ground stations around the world than a FPGA based solution. For this purpose, a new interface for UDP communication must be implemented at the ground stations. This can be an update of SLE or it can be a new interface. Furthermore it will be important that the devices in the communication chain will be configured correctly. For Linux this means that the kernel settings have to be set correctly and that RT-Tasks have to be used.

Table 4. Example jitter measurements with a WAN-Simulator using CLOCK_HZ = 1000 and CONFIG_NO_HZ = y and CONFIG_HIGH_RES_TIMERS = y .

Uplink Delay +/- Jitter (ms)	Measured RTT (ms)	#Packets
100 ± 30	101.2 ± 29.9	20000
100 ± 10	101.4 ± 10.0	20000
300 ± 15	301.3 ± 15.0	20000
600 ± 5	601.4 ± 5.1	20000
10 ± 1.5	11.1 ± 1.5	20000

Appendix A : Acronym List

ACE	ADAPTIVE Communication Environment
CLTU	Command Link Transmission Unit
CPU	Central Processing Unit
DEOS	Deutsche Orbitale Servicing Mission
DTN	Delay Tolerant Network
ESA	European Space Agency
FPGA	Field-programmable gate array
GPS	Global Positioning System
H/W Merger	Hardware Merger
IF	Interface
IF	Intermediate Frequency
IP	Internet Protocol
ISS	International Space Station
MC	Master Channel
NASA	National Aeronautics and Space Administration
PCS	Payload Control System
RCF	Return Channel Frame (SLE Service)
ROKVISS	Robotic Components Verification on the ISS
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
RT-Task	Real-time Task
SCOS	Spacecraft Operating System
SDH	Synchronous Digital Hierarchy
SLE	Space Link Extension
TC	Telecommand
TCP	Transmission Control Protocol
TM	Telemetry
TTL	Transistor-transistor logic
UDP	User Datagram Protocol
VC	Virtual Channel
WAN	Wide area network

References

[1]Klinkrad, H., *ESA space debris mitigation handbook*, European Space Agency, 2003.

[2]Rex, D., "Wird es eng im Weltraum? Die mögliche Überfüllung erdnaher Umlaufbahnen durch die Raumfahrt," *Carolo-Wilhelmina Mitteilungen II/1996, Braunschweig*, 1996.

[3]Landzettel, K., Preusche, C., Albu-Schaffer, A., Reintsema, D., Rebele, B., and Hirzinger, G., "Robotic on-orbit servicing-DLR's experience and perspective," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006, pp. 4587–4594.

[4]Sellmaier, F., Boge, T., Spurmann, J., Gully, S., Rupp, T., and Huber, F., "On-Orbit Servicing Missions: Challenges and solutions for Spacecraft Operations," *AIAA SpaceOps Conference*, 2010.

[5]Kaiser, D., Bellido, E., and Hofmann, P., "Space debris mitigation using on-orbit servicing solutions," *IAC 2010 conference*, 2010.

[6]Hirzinger, G., Landzettel, K., Reintsema, D., Preusche, C., Albu-Schäffer, A., Rebele, B., and Turk, M., "Rokviss-robotics component verification on ISS," *Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, 2005.

[7]Hirzinger, G., Brunner, B., Dietrich, J., and Heindl, J., "Sensor-based space robotics-ROTEX and its telerobotic features," *Robotics and Automation, IEEE Transactions on*, Vol. 9, No. 5, 1993, pp. 649–663.

[8]ESA-Telerobotics-Laboratory, http://esa-telerobotics.net/uploads/documents/Interact Brochure - Online.pdf, Jan. 2016.

[9]de Frescheville, F. B., Martin, S., Policella, N., Patterson, D., Aiple, M., and Steele, P., "Set-up and validation of METERON end-to-end network for robotic experiments," *ASTRA Conference*, 2011.

[10]Artigas, J. and Hirzinger, G., "A Brief History of DLRs Space Telerobotics and Force-Feedback Teleoperation," *Acta Polytechnica Hungarica*, Vol. 13, No. 1, 2016.

[11]"DEOS - Deutsche Orbitale Servicing Mission," http://www.dlr.de/rb/desktopdefault.aspx/tabid-6815/11182_read-40188/.

[12]"TC SYNCHRONIZATION AND CHANNEL CODING," *BLUE BOOK - CCSDS 231.0-B-2*, 2010.

[13]"SPACE LINK EXTENSION - RETURN CHANNEL FRAMES SERVICE SPECIFICATION," *BLUE BOOK - CCSDS 911.2-B-2*, 2010.

[14]Gnat, M., Falcone, R., Hauke, A., Ohndorf, A., and Eberle, S., "Technical and operational investigations of the real-time communication for robotic missions." *SpaceOps 2014 Conference*, 2014, p. 1825.

[15]Gnat, M. and Willburger, P., "Operations for parallel satellite support," *SpaceOps 2012*, 2012, pp. 11–15.

[16]Fu, L. and Schwebel, R., "Linux RT PREEMPT HOWTO," https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO.

[17]"netem," http://www.linuxfoundation.org/collaborate/workgroups/networking/netem, 2009.

[18]Brown, M., "Traffic Control HOWTO," http://www.tldp.org/HOWTO/Traffic-Control-HOWTO, 2006.

[19]Hubert, B., "Linux Advanced Routing and Traffic Control HOWTO," http://lartc.org/howto/, 2012.