Performance Optimization of Free-Space Optical Communication Protocols based on results from FSO Demonstrations

Bernhard Epple*[a]
[a] German Aerospace Center (DLR), Institute of Communications and Navigation, 82234 Wessling, Germany;

ABSTRACT

The mobile free-space optical channel mainly suffers from relatively long link outages, produced by short-term blockings of the line-of-sight (obstacles, clouds), pointing- and tracking-errors or deep signal-fades caused by index of refraction turbulence effects. This paper discusses the applicability of commonly used communication protocols like UDP, TCP, ARQ and the SCPS-TP from the Space Communications Protocol Standards (SCPS) in various scenarios. The performance of the protocols in the selected scenarios is evaluated using the simulation software OMNeT++. The simulations are based on channel measurements from the three FSO demonstrations FASOLT (61 km Ground – Ground link), KIODO (LEO satellite downlink), and ATENAA (land-mobile link) and from ongoing measurements at the German Aerospace Center (DLR) (short-range Ground - Ground) as part of the MINERVAA project. Based on the simulation results, recommendations for protocols in free-space optical communication scenarios are given.

Keywords: Free-space optics, mobile optical links, TCP, UDP, SCPS, ARQ, performance comparison, FASOLT, KIODO, ATENAA, OmNet, Communication Protocols.

1. INTRODUCTION

Data services for mobile applications are becoming part of people's lives and raise the need for high-speed communication technologies. To this end, free-space optical (FSO) communications technology has the potential to outperform radio frequency (RF) systems, especially in the area of backbone traffic. The mobile FSO channel mainly suffers from relatively long link outages. Causes of these outages are short-term blockings of the line-of-sight (LOS) by obstacles or clouds, pointing- and tracking-errors or deep signal fades caused by index of refraction turbulence effects. The optical channel also suffers from problems caused by the high data-rates and the long link distances that are typical for FSO application scenarios. This paper discusses the applicability of commonly used communication protocols like UDP, TCP, ARQ and the SCPS-TP from the Space Communications Protocol Standards (SCPS) in various scenarios (Ground – Ground, Space – Ground, land-mobile - stationary). The named protocols are used for data transfer since many years and great efforts have been made in the past to adapt these protocols to emerging areas, e.g. high data-rate links. In the first part of this paper the error characteristics of the FSO channel will be illustrated based on measurement results from FSO demonstrations. Some of the problems that occur for data communication over the optical channel also apply to modern communication systems. These problems will be discussed and common solutions will be shown. These solutions will be implemented within the OMNeT++ simulation framework [2] and evaluated for their applicability in the FSO environment. Based on the simulation results, recommendations for protocols in FSO communication scenarios are given.

Most work related to this paper has been done within the ATENAA project and its successor project MINERVAA [1]. Both projects are funded by the European Union as part of the European Community 6th Framework Program. Both projects have the goal of establishing FSO in the field of commercial aviation.

* Bernhard.Epple@dlr.de; phone +49 (0) 8153 2816; fax +49 (0) 8153 2844; www.dlr.de/kn/

2. ERROR CHARACTERISTICS OF THE OPTICAL CHANNEL

The optical channel is mostly characterized by its fading behavior. The received signal has strong variations in its power over time. Sources for these fades can be index of refraction turbulence effects in the atmosphere, pointing- and tracking-errors at sender and receiver or short-term blockings of the line-of-sight (LOS). The experienced fades have mostly time durations of several milliseconds, but they can also have time durations of over hundred milliseconds. During a fade the bit error probability increases significantly due to the reduced receive power. Unlike errors caused by additive white Gaussian noise (AWGN) on the wired-channel, bit errors induced by fades appear in bursts. For this reason the bit errors in the data stream transmitted over an optical link are not evenly distributed and the measured bit error ratio (BER) does not have a constant value. Instead, the quantity of received errors depends on the current state of the channel.

2.1 Measuring the Bit Error Ratio

A general approach of measuring the bit error ratio of any channel is to send a known sequence of data over the channel and to count the received bit errors. The ratio of bit errors over transmitted bits measured during the integration time is the BER. In most literature only one BER value is given for measured channels. Because of the fading behaviour of the channel this value can only be seen as the long-term mean value of the bit errors. For designing communication protocols this value should be used with precaution because statistical information about the error distribution is not included. To take the error distribution into account, the terms Short-Term and Long-Term BER shall be introduced. It can be said that during the duration of one fade the BER remains constant. Therefore the BER during one fade shall be called Short-Term BER (BER_{ST}). The BER measurement over multiple fades shall be called Long-Term BER (BER_{LT}). BER_{LT} can be understood as the mean value of all BER_{ST} that lie within the measurement interval. Measuring the BER_{ST} using common bit error measurement devices is nearly impossible. Neither can they be triggered on the start of a fade for initiating a new measurement nor are they build for such short measurement durations and high measurement frequencies. In [3] a way is shown how the BER_{ST} can be calculated from received power measurements.

2.2 Calculating BER_{ST} from Received Power Measurements

Hardware for measuring and logging the received power is widely available and offer sampling rates of several kHz. This sampling rate is fast enough to measure isolated fades since the typical fade duration can be assumed to be several milliseconds, as already mentioned. According to [3] the calculation of BER_{ST} from received power measurements can be done using Formula (1) and (2):

$$\mathrm{BER}_{\mathrm{ST}}(P) = Q\left[\frac{P/P_0}{1+(1+\xi_0 P/P_0)^{1/2}}\right] \tag{1}$$

and

$$\mathrm{Q}(x) = \frac{1}{\sqrt{2\pi}}\int_x^{\infty}\exp[-(t^2/2)]dt \tag{2}$$

The form factor ξ_0 and characteristic power P_0 are parameters of a receiver model described in [4]. For the Fujitsu FRM5W621KT/LT module, the values ξ_0=0.8 and P_0 =1.35 nW have been found. It is important to mention that the receiver model is bound to the data rate of the receiver and can not be scaled to other data rates. The module operates at a wavelength of 1550 nm and a data rate of 622 Mbit/s and shall be used as reference receiver throughout this manuscript.

2.3 Received Power Measurements and their Short-Term BER

At the German Aerospace Center (DLR) several successful FSO demonstrations have been accomplished in the past. These demonstrations have also been used to measure the characteristics of the atmospheric channel. So results from several power measurements for different channel types are available. The measurements from four demonstrations have been chosen as data source for characterizing the optical channel. The first selected demonstration is the FASOLT demonstration where an optical link between a receiver at the DLR site in Oberpfaffenhofen near Munich and a transmitter on the Wallberg, a mountain in 61 km distance, has been shown in 2002 [5]. The second measurement is taken from the KIODO project held in June 2006. KIODO was a cooperation with the Japan Aerospace Exploration

Agency (JAXA). In this project optical LEO-downlinks from the Japanese OICETS satellite to DLR's optical ground station in Oberpfaffenhofen (OGS-OP) have been demonstrated [6]. In 2007 a demonstration within the European Union funded project ATENAA was held. In cooperation with other partners from throughout Europe a link between an aircraft simulator at Oberpfaffenhofen's airport and the OGS-OP has been demonstrated over a distance of 1400 meters. During the measurement from the ATENAA demonstration, the aircraft simulator was driving with a velocity of 20 km/h on the taxi way, while some obscurations (trees, bushes, poles) frequently blocked the line of sight between sender and receiver [7], [8]. The last of the four selected measurements is taken from an ongoing measurement campaign at DLR's premises in Oberpfaffenhofen where a stationary link between two buildings over 500 meters has been set up. From the available measurements made in this testbed setup, a measurement has been selected which has been taken on a hot sunny day with strong winds which caused strong atmospheric turbulences. The data from these measurements together with (1), (2) and the characteristic receiver values from above have been used to calculate the BER_{ST} for the optical links of these demonstrations. For making the measurements comparable to each other they have been scaled to result in a Long-Term BER of 10^{-6} for the given receiver. This scaling can be done because the scaling has a similar effect as changing the transmit power and does not change the dynamic of the receive power. A BER_{LT} of 10^{-6} can generally be assumed as a typical value for experimental setups. Systems that are used for data transmission generally have a BER_{LT} of 10^{-9} or lower. The ATENAA measurements had to be treated in a special way because of the obscurations in the line of sight. These obscurations are part of the communication scenario, but not part of the channel characteristics. Therefore drops in receive power caused by obscuration had to be removed for the scaling process and reinserted afterwards for the simulations. In Figure 1 the calculation results are shown.

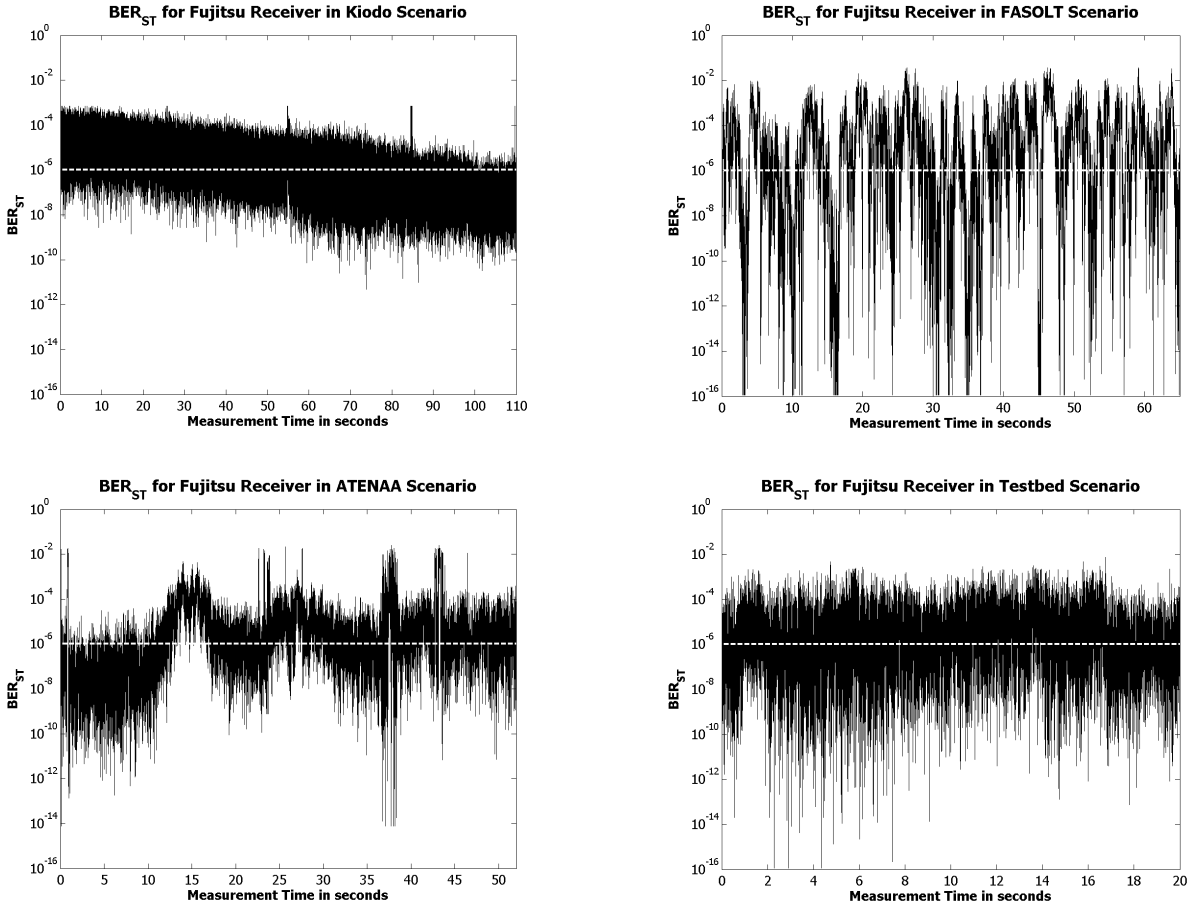

Fig. 1. The calculated Short-Term BERs for the selected measurements, based on the Fujitsu receiver model. The white dashed line marks the BER_{LT} of 10^{-6}. Since all measurements belong to different scenario types they show different error characteristics. For example the FASOLT measurement has long and strong fades and the testbed measurement is very unsteady due to the extreme weather conditions.

The KIODO measurements depict a very stable channel with only slight changes in BER_{ST}. As one can see BER_{ST} is getting better towards the end of the measurement. This behaviour is caused by the fact that the rising edge of the satellite pass at the OGS-OP has been selected and therefore the link distance decreases during the satellite pass. You can also see three spikes in BER_{ST} calculations at around 30, 60 and 90 seconds which are cause by tracking problems, when the signal got lost for a very short duration. The FASOLT link suffers from very long and deep fades compared to the other measurements. It is supposed that these fades are caused by strong turbulences at the sender. The measurements have been done during winter and there was a large temperature difference between the room (with an opened window) in which the sender was located and the outside air. The channel in the land-mobile scenario ATENAA is frequently changing its over all state due to the motion of the aircraft simulator and the blockings of the line of sight. In the measurements taken at the DLR testbed the challenging weather condition can clearly be seen in the variance of the BER_{ST}.

3. GENERAL PROTOCOL ASPECTS FOR LOSSY HIGH-BANDWIDTH LONG-DELAY CHANNELS

Links in the free-space optical channel have besides their special error characteristics some other characteristics that cause problems for communication protocols. These problems are caused by the typical high-bandwidth and the long propagation delays of these links and shall be discussed later. First, methods for reliable data transmission on lossy channels will be summed up.

3.1 Reliable unidirectional data transfer

One of the most known protocols for unidirectional data transfer is the User Datagram Protocol [9]. UDP is a connectionless and unreliable protocol. Unreliable because it does not guarantee the successful delivery of the data to the receiver. Because of its connectionless design and its broad application in communication networks UDP is a suitable candidate for implementing data streaming over a unidirectional FSO. Nevertheless in several scenarios containing unidirectional links it is desirable to guarantee data delivery. Due to the absence of a back channel, forward error correction (FEC) is the only possible solution. For FEC a coder at sender side inserts redundant data into the data stream which enables a decoder at the receiver side to recover lost data from the available redundancy information. The easiest way to implement FEC is to transmit every packet n-times in the hope that one of these redundant packets will not be lost. This approach is only feasible as long as enough bandwidth is available. It is also a quite inefficient method. In cases when the channel is in a good state and the first packet arrives at the destination there will also be n-1 unnecessarily sent packets arriving at the receiver. More sophisticated versions of FEC use special codes like Reed-Solomon, Hamming, or Turbo codes to encode and decode the transmitted data for a more efficient use of the available bandwidth. One problem of FEC is that the amount of lost data has to be guessed before the data is transmitted. This might lead to a waste of bandwidth if the guess has been too high or a data loss if the guess has been too low. Guessing the right amount of needed redundancy is nearly impossible. Therefore, if a backchannel is available, it is advisable to use the backchannel for reliable data transfer. More information about FEC over the optical channel can be found in [10].

3.2 Reliable bidirectional data transfer

If a backchannel is available for communication this channel can be used to inform the communication partner about lost data. Protocols exchanging information about sent and received data for a reliable transfer belong to the group of Automatic Repeat reQuest (ARQ) protocols. In general there are four types of ARQ protocols. The first type is the Stop-and-Wait protocol which sends one datagram and waits until it receives an acknowledgment packet (ACK) from the receiver. If the ACK does not arrive within a certain amount of time, the packet is retransmitted. If an ACK arrives the sender can send the next packet. This protocol is easy to implement but very inefficient because the sender spends a lot of time by waiting for ACKs. The second protocol type is the Go-back-N protocol. With this, the sender maintains a window of data that can be sent without waiting. As long as space within the window is available the sender transmits data. If the window has been sent the sender waits for the ACKs of the sent packets. As soon as the packet at the lower end of the window gets ACKed, the window is advanced and a new packet is sent. If an ACK does not arrive within a certain amount of time the complete window is resent. This approach reduces significantly the time spent waiting but it wastes bandwidth because the complete window is resent even if only one packet got lost. The third protocol type is the Selective-Repeat-ARQ. This type is similar to the Go-back-N protocols but it reduces the amount of retransmissions by only retransmitting not acknowledged packets from the window. The implementation complexity of this type is rather high compared to the other ARQ types because a non sequential buffer management has to be implemented at the

receiver. The amount of memory needed for the reordering of the received buffer depends on the amount of data that can be en route between sender and receiver, but on lossy channels it can become a multiple of this value if the window size is not set to be a limiting factor. For this reason Go-back-N has been preferred over Selective-Repeat in times when memory was a bottleneck to communication systems in terms of amount, cost and performance. But this should have changed now for most applications. A further optimization of the ARQ protocols is the addition of negative ACKs (NAK) which signal the sender a missing packet before its retransmission timeout has been reached. It is also possible to have no retransmission timer at the sender and completely rely on successful delivery of NAKs. In this case a retransmission timer at the receiver has to be implemented for retransmitting the NAKs if the requested data does not arrive in a certain time interval. Since in most cases the number of successfully transmitted packets should be larger than the number of lost packets, the use of NAKs should reduce the numbers of unnecessarily retransmitted packets due to wrong set retransmission timers and lost ACKs. The fourth and last type in this group is hybrid ARQ which is a combination of FEC and Selective-Repeat ARQ. Since these protocols are very complex structured they would exceed the focus of this paper.

3.3 Protocol Features needed for efficient communication over FSO channels

The Transmission Control Protocol (TCP) is one of the most used protocols for the bidirectional and reliable data transfer. It is part of the core protocols of the internet stack and was first defined in RFX 793 in 1981 [11]. TCP belongs to the group of Go-back-N protocols and many attempts have been made to improve its performance especially over lossy high-bandwidth long-delay channels; like today's internet with broadband access in many homes. Because the same characteristics also apply to the optical channel, the proposed solutions can be transferred from the wired internet to the FSO channel. A good collection of the most promising improvements is the document "TCP Extensions for High Performance" [12]. Also this document is tailored on the performance of TCP the fundamental concepts can be transferred to all ARQ based protocols. Another good source for improvements to TCP is the Transport Protocol from the Space Communications Protocol Specifications (SCPS-TP) [13] which is designed for high latency microwave satellite connections.

3.3.1 The Minimum Window Size

FSO links have typically a high bandwidth and in many cases also a long link distance which is troublesome for maintaining the send window in ARQ protocols. The send window is holding all bytes that are sent to the receiver and have to be stored for a possible retransmission request. The minimum window size WND_{min} (if the send window should not become a limiting factor) can be calculated for FSO communications as

$$WND_{\min} = RTT \times D \tag{3}$$

with

$$RTT = \left(\left[2 \times \frac{d}{c} \right] + t_{proc} \right) \tag{4}$$

Where RTT is the so called round trip time, d = link distance, c = speed of light, t_{proc} = processing time of the protocol stack at the partner and D = link bandwidth. As example, for a LEO downlink with a maximum link distance of 2500 km at low elevation angles and a data rate of 1 Gbit/s this comes to a window size of at least 2.084.775 byte that have to be managed by the protocol stack. TCP for example foresees only a window size of up to 16 bit which will not be sufficient for addressing the window data of most FSO applications without limiting the throughput of the protocol. Therefore a windows size up to 32 bit should be implemented. The window size is also defining the minimum size of the buffer at the receiver. In systems with low available memory the windows size can be used as flow control to reduce the amount of needed buffering memory at the receiver.

3.3.2 Accurate calculation of the Round Trip Time

In TCP all retransmissions are triggered by the retransmission timer, so it is important to set these timers very accurate. If the timer is set on a value too low it will result in a retransmission of the packets before the ACKs can have reached the sender and if it is set too high the sender will waste transmission time by unnecessarily waiting for the timeout in case the ACK got lost. Both errors will result in a loss of useful throughput. The timing in most TCP implementations is done by noting the send time for every packet and calculating *RTT* from this value when the corresponding ACK arrives. As TCP is only acknowledging the last correct received packet, this ACK based clock will stop and no recalculation of the timeout will be done until this packet gets successfully delivered. The clock will be on hold for at least *RTT* and might cause unnecessary retransmissions. Since t_{proc} is part of *RTT* and the receiver might behave different as long as it is waiting for missing packets, the *RTT* calculations might be incorrect until the complete window has been retransmitted and the receiver has settled. A frequently suggested improvement to this behaviour is the addition of timestamps to every packet. To prevent the need of synchronizing clocks between communications partners, the last received timestamp of a sender should be echoed in the answer from the receiver. If the protocol gets enhanced by additional features the implementer has to take care to use the correct timestamp to echo. For example in case of delayed ACKs the time stamp of the oldest packet in the acknowledgement list should be used although intuitively one might want to echo the last received time stamp. If the last received timestamp would be echoed, the first packets in the ACKed list would always time out before the delayed ACK arrives. [14] gives details about the round trip time calculation. One fact to mention here is that [14] gives a value of three seconds as initial round trip time estimation. This value is used until the first measurement based on timestamps can be done. This means at least one transmission window of data will be scheduled with a wrong timeout. Depending on the link scenario a different choice for the initial *RTT* estimate will be advisable.

3.3.3 Optimal Packet Size Selection

The selection of the packet size can have crucial impact on the performance of a protocol. If the packet size is set too small, the ratio between header length *l'* and user data length *l* becomes unfavourable for reaching a high goodput (i.e. user data arriving at the receiver per second). If the packet size is set too high the packet loss probability p_{PL} will increase and also lead to a reduced goodput as shown by (5) for the AWGN channel [15].

$$p_{\mathrm{PL}} = 1 - \left(1 - BEP\right)^{l+l'} \tag{5}$$

Where *BEP* is the bit error probability for the channel. For ARQ protocols also the loss of an ACK is interpreted as a loss of the corresponding packet, (5) should be extended to

$$p_{\mathrm{PL}} = 1 - \left(1 - BEP\right)^{l+l'} + 1 - \left(1 - BEP_{Back}\right)^{l_{ACK}} \tag{6}$$

Where BEP_{Back} is the bit error probability of the backchannel and l_{ACK} is the length of an ACK packet. A feature used in most ARQ implementations is to piggyback the ACK as a flag within a data packet. This is done for reducing the number of transmitted packets. Because data packets are generally longer than ACK packets, this optimization increases the risk of losing a packet due to a lost ACK. Based on formulae developed in [15] and [16] formulae (8), (9) and (10) for the normalized goodput *G* can be derived for the three main classes of ARQ protocols. For simplicity the given formulae neglect the time duration of ACK packets and the processing time at the receiver since both time values can be expected to be very small for high bandwidth systems. They also neglect the fact that in most implementations the transmission window is limiting the data flow and they assume that the channel is in a saturated state, i.e. there is continuous traffic on the channel. In the following the round trip time capacity C_{RTT} is defined as the maximum number of packets that can be on the channel at the same time.

$$C_{RTT} = \frac{RTT \times D}{l + l'} \tag{7}$$

The use of C_{RTT} allows to model the time spent waiting in Stop-and-Wait and Go-back-N protocols as “wasted bits” that reduce the goodput. The goodput *G* can then be defined as the ratio between the one successful transmitted packet and the “waste packets” (waiting time, retransmissions) that had to be transmitted for this successful transmission. This simplifies the calculation of the goodput and leads to following formulae:

Goodput Selective-Repeat ARQ G_{SR}:

$$G_{SR} = (1 - p_{PL}) \times \frac{l}{l + l'} \tag{8}$$

Goodput Stop-and-Wait ARQ G_{SaW}:

$$G_{SaW} = \frac{1}{1 + C_{RTT}} \times G_{SR} \tag{9}$$

Goodput Go-back-N ARQ G_{GbN}:

$$G_{GbN} = \frac{1}{1 + C_{RTT} \times p_{PL}} \times G_{SR} \tag{10}$$

In the formulae for Stop-and-Wait and Go-back-N *RTT* and *D* are directly influencing the goodput of the protocol. This means that these protocols will behave differently in the selected scenarios due to the different link distance. From the three given formulae it can be seen that the goodput of the Selective-Repeat ARQ is an upper bound for the performance of all three protocol classes. In the following the effect of changing the packet length *PL=l+l'* will be illustrated for the three protocol classes. For the calculations made, the header size has been fixed to 20 bytes which is a typical value for modern communication protocols. As data rate the 622 Mbit/s of the reference receiver have been used. Figure 2 shows the calculation results for Stop-and-Wait protocols in the testbed and the FASOLT scenario. For the testbed scenario the goodput is mainly influenced by the ratio between *l* and *l'*. This is due to the fact that the testbed has a link distance of only 500 meters, so *RTT* is rather short and the sender has not to spend much time waiting for ACKs. For small packet lengths the goodput drops not only because of a bad ratio between *l* (variable) and *l'* (fixed to 20 byte) but also because of relatively more time spent waiting if the transmission time of a packet is only a fraction of *RTT*. If the link distance is increased like for the FASOLT scenario, the ratio between *RTT* and the time needed for transmitting the packet becomes the determining factor. The time needed for transmitting a packet can be seen as useful time spent on the channel while the remaining time of *RTT* can be seen as the amount of time wasted by waiting for a reply from the receiver. When the link distance is increased to the 61 kilometers of the FASOLT scenario, the goodput degenerates heavily for all packet lengths because the time spent waiting is also increased. Figure 3 shows the calculation results for Go-back-N protocols in the testbed and the FASOLT scenario. For the testbed scenario the goodput is again mainly influenced by the ratio between *l* and *l'* but the influence of *RTT* on the goodput if shorter packets are used is reduced. That *RTT* is still an influencing factor can be seen in the plot for the FASOLT scenario. Another observation from these plots is that the use of packet lengths longer than 1000 byte does not result in a significant increase of the goodput. Finally Figure 4 shows the calculation results for Selective-Repeat ARQ. For this type of protocol *RTT* is no longer a factor and the calculations made are valid for all scenarios. The plot looks similar to the plot of Go-back-N over the testbed link. Therefore in scenarios with shorter link distances the implementation of a Go-back-N protocol might be favorable over a Selective-Repeat protocol due to its easier implementation. The second plot in Figure 4 show the goodput over the packet length for various bit error ratios. It can be seen that also for the Selective-Repeat ARQ the goodput is not increasing significant when packets longer than 1000 bytes are transmitted. This explains why in most communication protocols the packet length is limited to a value between 1000 and 2000 bytes, e.g. Ethernet limits the maximum frame length to 1500 bytes. For Selective-Repeat ARQ a formula for the optimal length of the user data l_{opt} can be obtained by differentiating (8) and setting the derivative to 0.

$$l_{opt} = \frac{l'}{2} \times \left(\sqrt{1 - \frac{4}{l' \times \ln(1 - BEP)}} - 1 \right) \tag{11}$$

With (11) it is possible to adjust the packet length to the BER_{ST} to receive an optimal goodput. But it is most unlikely that BER_{ST} measurements will be available to the protocol stack during runtime and if there should be such measurements they will be outdated immediately due to the channel dynamics. It would be a desirable feature to calculate or even predict the BER_{ST} during runtime for optimizing the packet length.

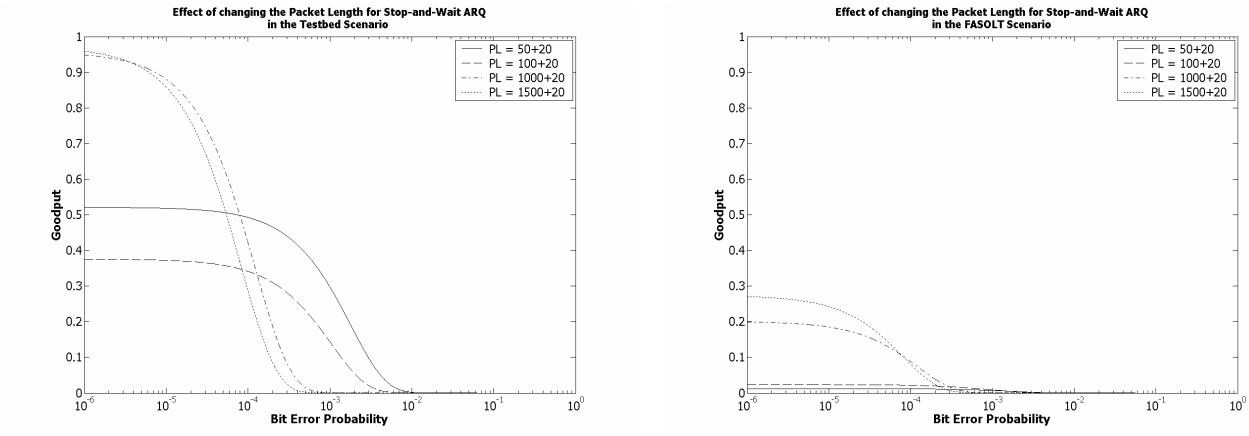

Fig. 2. Effect of changing the packet length on the goodput of Stop-and-Wait ARQ protocols. The long distance in the FASOLT scenario (right) results in an inefficient data communication due to the long waiting times.

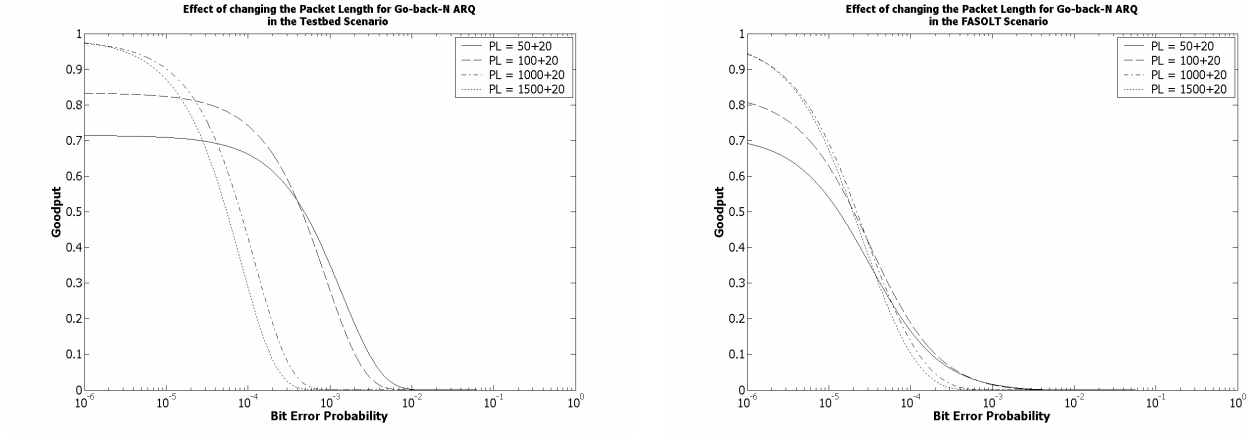

Fig. 3. Effect of changing the packet length on the goodput of Go-back-N ARQ protocols. The long distance in the FASOLT scenario (right) results in an obvious change in goodput. Packet lengths >1000 byte do not seem to change the goodput.

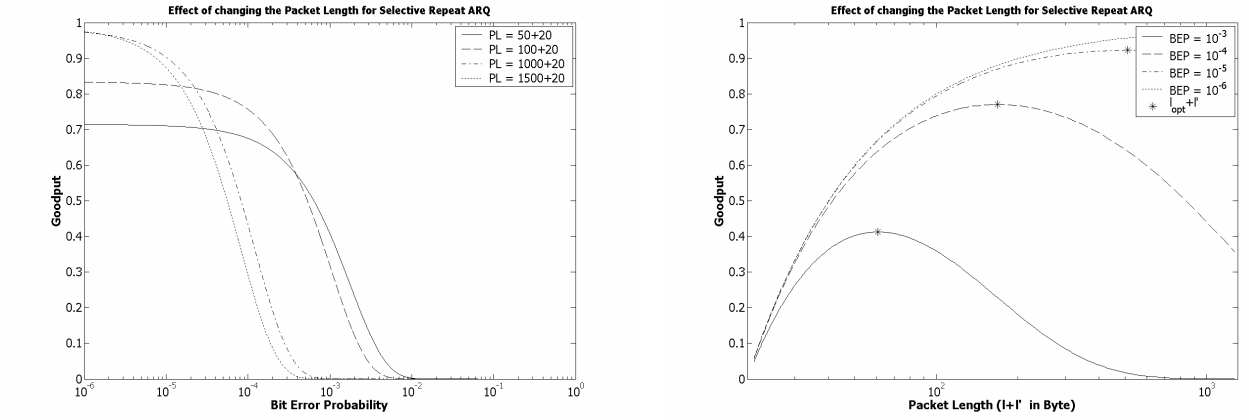

Fig. 4. Effect of changing the packet length on the goodput of Selective-Repeat ARQ protocols. On the left the goodput is plotted over the BER for various packet lengths. On the right it can be seen that packet lengths >1000 byte do not significantly increase the goodput. The optimum packet lengths according to (11) are marked in the plot by *.

One characteristic that all protocol types have in common is that at bit error ratios larger than 10^{-5} the goodput starts to drop significantly for all packet lengths. Therefore a BER_{LT} less than 10^{-5} should be the design goal of every communication system. Based on the observations made in this section, the packet lengths used later in the simulations will be limited to the range from 500 to 1500 byte, to limit the number of needed simulation runs.

3.3.4 Selective Acknowledgements and other ARQ depended features

It is commonly understood that the use of selective Acknowledgements in ARQ protocols can significantly improve the protocol performance. This is especially the case if communication is done over lossy channels. For a correct handling of selective ACKs the correct received parts of the data stream have to be communicated between sender and receiver. This can either be done by sending the start and end positions of the correct received data blocks or by sending the start and endpoints of the gaps in the data stream. This means that the number of acceptable fragments has to be limited to a certain amount; otherwise there must be space for a huge amount of fragments in the header of each packet. The maximum number of fragments is half the window size as it can easily be seen. If the selective ACK feature should be added to standard TCP the available space in the TCP options field allows for transmitting the positions of four data fragments. All other fragments created during data transmission will have to be rejected by the receiver until the first created fragment could be accepted by the receiver. Therefore the noisier the channel gets the less the performance gain by this feature will be.

In environments where the delay is more important than the correct delivery of the data, the number of retransmissions can be limited to a desirable value. This feature is only useful if there is error correction implemented on one of the higher layers sitting on top of the protocol layer. This is the case for voice transmission or similar. Such a feature can, for example, be activated in SCPS-TP. Other features implemented by SCPS-TP are selective negative acknowledgement for further optimizing the request strategy and multiple transmissions of packets for forward error correction. A performance comparison of TCP and SCPS-TP can be found in [17].

3.3.5 Congestion control

Not specifically contained in the original TCP definition congestion control is one of the most important features added to TCP. In networks with many sources using the same links, the senders have to use the channel cooperative for not jamming the channel by trying to send more data than the channel can bear. For avoiding such situations, lost packets are generally interpreted as a sign of congestion and the sender detecting these packet losses will reduce its transmission rate to reduce the load on the channel. On lossy channels the base assumption that packet losses are caused by congestion is wrong and therefore congestion control algorithms will unnecessarily reduce the load on the channel. The SCPS-TP standard supposes to implement Explicit Congestion Notification as defined in [18] for being able to distinguish between channel congestion and packet corruption. [17] shows that it is also a suitable option to disable congestion control at all to prevent any problems. In point to point links it is suggested by the author to disable congestion control because it should not be needed.

4. EVALUATION OF PROTOCOL ENHANCEMENTS BASED ON SIMULATION RESULTS

In the following, the timestamp and the Selective-Repeat feature have been chosen for evaluation because these seem to be the base features that are generally proposed tuning protocol performance. The evaluation of these two features is done based on simulations within the OmNeT++ simulation frame work. The used simulation model consists of one point to point link between a sender and a receiver. Although the modeled links between the two partners are bidirectional links, one partner has been determined to be a sender, the other one to be the receiver. On each connection between the two communication partners a module has been placed that inserts errors into the transmitted packets based on the Short-Term BERs gathered from the measurements. The propagation delay between the partners has been set to fit the scenario statistics where the selected measurements originate from. In this way the two communication directions are modeled to have the same error characteristics during the communication. It is known that this assumption does not hold in the general case but the simulation results should still be accurate enough to evaluate the performance gain caused by the protocol enhancements. The reference protocol used for the evaluation is the standard TCP as defined in RFC 793 with no congestion control implemented. In the simulation model no further error correcting layer is implemented below TCP so the simulated protocol resides on the data link layer and not, as in the ISO/OSI model, on the transport layer. In a real world scenario TCP will behave slightly different because of the underlying layers doing their work. The selected features were implemented into the TCP protocol for keeping the simulation results comparable to each other. For adding

features to TCP a special option field is reserved in RFC 793 which can add up to 40 bytes to the header size. For implementing the timestamp feature 10 additional bytes are needed for storing the timestamps and marking the option. For implementing Selective-Repeat for each lost packet a new addressable block of 8 bytes has to be added to the header. For marking the option 2 byte are needed. Without the timestamp feature a total of four data fragments can be addressed by selective acknowledgements, if compatibility to TCP should be kept. If the timestamp feature is added to TCP, only three addressable blocks are available in the header for selective acknowledgements. To evaluate the correctness of these implementations, some test runs over channels with AWGN have been made and the results have been compared to the values gained from theory presented in section 3.3.3. See Table 1 for test results. Because in the simulation results the initialization and completion of the communication is contained (i.e. the simulated channel is not always in a saturated state), the simulated goodput values differ slightly from the calculated goodput values. Another source for differences between simulation results and values gained from theory is that in the simulation unnecessary retransmissions will be sent due to bad *RTT* calculations and the scheduling of retransmissions will not be ideal. This will further reduce the goodput.

4.1.1 Evaluation of the Timestamps Feature

For the KIODO scenario it had been thought that this feature would give a great improvement due to the varying propagation delay during the transmission. The biggest improvement had been expected on the descending edge of the satellite pass where the growing propagation delay should lead to an underestimation of *RTT* and therefore to many unnecessary retransmission timeouts. The simulation of this case has been done by using the available measurement in reversed order. An improvement in goodput if the timestamps are used could only be seen if also an aggressive calculation algorithm for *RTT* had been used. But still several unnecessary timeouts occurred, which reduced the goodput. For a better performance the *RTT* calculation algorithm had to be changed to a more conservative calculation method which tends to overestimate *RTT*. By this change the number of too early timeouts could be heavily reduced, but the goodput gain caused by the timestamp feature has been also reduced. The additional header bytes for the timestamps caused also a slightly reduced goodput. The best results for the KIODO downlink could be reached with the timestamps disabled and the use of a conservative *RTT* calculation algorithm. The second scenario where the timestamps should have given a goodput gain is the ATENAA scenario where the motion of the aircraft simulator causes small jitter in the propagation delay. Also in this case the improvement caused by the timestamps could not recover the loss caused by the increased header. For all four simulated point to point links the timestamp feature could not give any improvement in goodput, so it is not advisable to implement it. For networks this observation will change because the experienced delay jitter in networks will be much stronger.

4.1.2 Evaluation of TCP with Selective-Repeat

Because it had been decided not to evaluate the timestamps option any further, four data blocks could be implemented in the TCP header for simulation of TCP with Selective-Repeat. The first simulation runs have been done for evaluating the protocol implementations against the given theory for transmission over the AWGN channel. They have been named AWGN Testbed and AWGN KIODO to show that these two simulation scenarios are using the channels from these demonstrations but with AWGN instead of errors caused by fading. Table 1 lists the goodput values for all simulated scenarios. For comparison the calculated values from the AWGN channel with a BER_{LT} of 10^{-6} in the same scenario are given in brackets.

Table 1. Goodput comparison of simulation results to AWGN theory. The results from calculation for an equivalent AWGN channel are given in brackets. The used header size is 20 byte for TCP and 60 byte for Selective-Repeat ARQ. The BER_{LT} for AWGN calculations is 10^{-6}. The last two columns are showing the throughput (percentage of bandwidth used from the available bandwidth) in the simulation for the backchannel. The backchannel traffic is solely caused by ACKs.

Scenario	User Data *l* in byte	Goodput TCP Simulation Results (AWGN calculation)	Goodput Selective-Repeat Simulation Results (AWGN calculation)	Throughput Backchannel	
				TCP	Selective-Repeat
AWGN Testbed	500	0.9306 (0.9562)	0.8880 (0.8893)	0.0528	0.1066
	1000	0.9479 (0.9710)	0.9347 (0.9359)	0.0270	0.0561
	1500	0.9347 (0.9735)	0.9479 (0.9501)	0.0181	0.0379

AWGN KIODO	500	0.0174 (0.0874)	0.2712 (0.8893)	0.0528	0.1065
	1000	0.0200 (0.0874)	0.2857 (0.9359)	0.0270	0.0560
	1500	0.0218 (0.0873)	0.2905 (0.9501)	0.0181	0.0379
Testbed	500	0.8222 (0.9562)	0.8506 (0.8893)	0.0507	0.1021
	1000	0.7755 (0.9710)	0.8712 (0.9359)	0.0254	0.0523
	1500	0.7557 (0.9735)	0.8656 (0.9501)	0.0167	0.0347
ATENAA	500	0.8245 (0.9520)	0.8362 (0.8893)	0.0499	0.1004
	1000	0.8218 (0.9667)	0.8561 (0,9359)	0.0249	0.0514
	1500	0.8163 (0.9692)	0.8531 (0.9501)	0.0164	0.0341
FASOLT	500	0.4944 (0.7706)	0.6291 (0.8893)	0.0428	0.0875
	1000	0.0421 (0.7799)	0.6693 (0.9359)	0.0202	0.0439
	1500	0.0644 (0.7811)	0.6758 (0.9501)	0.0131	0.0288
KIODO	500	0.0163 (0,0874)	0.1942 (0.8893)	0.0518	0.1041
	1000	0.0186 (0.0874)	0.1969 (0.9359)	0.0261	0.0537
	1500	0.0206 (0.0873)	0.1978 (0.9501)	0.0173	0.0357

The simulation results for the AWGN Testbed channel show that the protocol simulation works as expected. The simulation results are well in range of the theoretical values. For the AWGN KIODO channel things are a little bit different. The TCP implementation is fine, but the TCP with Selective-Repeat does not reach the goodput level given by theory. The reason for this is the limitation to only four data blocks that can be used for Selective-Repeats. The given theory assumes unlimited available blocks and it also neglects the increased header size for every added block. In case all available blocks are occupied, the implementation will have a goodput similar to TCP until space for a new block becomes available. To ensure the experienced drop in goodput for the KIODO channel is really caused by this limitation of data blocks, simulation runs with unlimited blocks and not increasing headers have been done. In these runs the theoretical values could be reached. For implementing Selective-Repeat ARQ with a near theoretical goodput over the AWGN channel, the average number of required blocks $N_{\mu Blocks}$ in a given communication scenario can be calculated by:

$$N_{\mu Blocks} = C_{RTT} \times \sum_{i=1}^{\infty} p_{PL}^{i} = C_{RTT} \times \left(\frac{1}{1 - p_{PL}} - 1 \right) \tag{12}$$

Then $N_{MaxBlocks}$, the maximum number of blocks needed, can be calculated by using the cumulative distribution function:

$$N_{MaxBlocks} = \sigma \sqrt{2} \times erf^{-1}(2p_C - 1) + N_{\mu Blocks} \tag{13}$$

With p_C being the confidence probability of the calculated maximum value and σ being the standard deviation of the AWGN channel. In the FASOLT and the testbed channel, the best goodput by TCP is reached if a packet size of 500 byte is used. If the packet length is increased, the goodput decreases. In the FASOLT scenario increasing the packet length leads to a drastic drop in goodput. This behavior can only be explained by the specific fading characteristic of these channels, which seem to favor shorter packages. For all other channels, the goodput is not significantly increased if longer packets are transmitted; therefore shorter packets should be preferred in the general case. In Table 1 also the throughput (percentage of bandwidth used from the available bandwidth) of the backchannel is given for all simulation runs. A common rule of thumb for TCP says that the backchannel should have at least a bandwidth of 10 % of the forward channel bandwidth. This rule also holds for Selective-Repeat ARQ as long as packets not shorter than 500 byte are transmitted. In the general case for Selective-Repeat ARQ it can be said that the backchannel should have at least a bandwidth of 20 % of the forward channel bandwidth.

CONCLUSION

It has been shown that Long-Term BERs are not suitable to simulate or to predict the performance of communication protocols over the FSO channel. An important factor for protocol design is the fading characteristic of the channel which might favor selected packet sizes. This coupling should be subject to further research for an improved performance of ARQ protocols over the optical fading channel. At the moment it can only be said that shorter packets (<1000 byte) should be used and that calculations for a comparable AWGN channel give the upper bound of the expectable protocol performance. For protocol design, the Short-Term BER is an important measure which can be calculated from receive power measurements. The possibility of calculating BER_{ST} in real time during communication would be desirable for adjusting the packet length to the current state of the channel. It is assumed that such dynamic packet lengths would increase the goodput. Another made observation is that over short link distances (Testbed, ATENAA) Go-back-N ARQ performs as good as Selective-Repeat ARQ and might be preferred because of its easier implementation. Over longer distances the implementation of Selective-Repeat ARQ should be considered because of its much higher goodput. For LEO downlinks the constant transmit power results in a variation of the receive power over time. For a more efficient power use and a constant link quality it is advisable to adjust the transmit power according to the elevation angle at the receiver.

REFERENCES

1. *European Community 6th Framework Program*, "MINERVAA, Mid-term Networking Technologies In-Flight and Rig Validation for Avionic Applications".
2. OMNeT++, Discrete Event Simulation System, http://www.omnetpp.org
3. N. Perlot, "Evaluation of the Scintillation Loss for Optical Communication Systems with Direct Detection", *Optical Engineering*, 46(2), 025003 (2007), 2007.
4. F. David, "Scintillation Loss in Free-Space Optic IM/DD Systems", *Proceedings of the SPIE*, Vol. 5338, 2004.
5. F. David, et. al., "Overview of the FASOLT Experiment and Final Results", *Proceedings of the SPIE*, Vol. 4975, 2003.
6. N. Perlot, et. al., "Results of the Optical Downlink Experiment KIODO from OICETS Satellite to Optical Ground Station Oberpfaffenhofen (OGS-OP)", *Proceedings of the SPIE*, Vol. 6457, 2007.
7. C. Fuchs, et. al., "Broadband Communications for Aeronautical Networks: The ATENAA Outer Optical Link Validation", *1st CEAS Air and Space Conference, Berlin, Germany, 10th-13th September 2007*, 2007
8. *European Community 6th Framework Program,* "ATENAA, Advanced Technologies for Networking in Avionic Applications", CEC contract number AST3-CT-2004-502843, URL: www.atenaa.org.
9. J. Postel, "User Datagram Protocol", *RFC 768*, http://tools.ietf.org/html/rfc768, 1980.
10. H. Henniger, F. David, D. Giggenbach, and C. Rapp, "Evaluation of FEC for the atmospheric optical IM/DD channel", *Proceedings of the SPIE*, Vol. 4975, 2003.
11. Information Sciences Institute University of Southern California, "Transmission Control Protocol", DARPA Internet Program, *RFC 793*, http://tools.ietf.org/html/rfc793, 1981.
12. V. Jacobson, et. al., "TCP Extensions for High Performance", *RFC 1323*, http://tools.ietf.org/html/rfc1323, 1992.
13. The Consultative Committee for Space Data Systems, "Space Communications Protocol Specification (SCPS)-Transport Protocol (SCPS-TP)", *Blue Book: Recommended Standards*, CCSDS 714.0-B-2, October 2006.
14. V. Paxon and M. Allman, "Computing TCP's Retransmission Timer", *RFC 2988*, http://tools.ietf.org/html/rfc2988, 2000.
15. M. Schwartz, "Telecommunication Networks: Protocols, Modeling and Analysis", *Addison-Wesley Series in Electrical & Computer Engineering*, Addison-Wesley Publishing Company, 1987.
16. T. Saadawi, M. Ammar and A. El Hakeem, "Fundamentals of Telecommunication Networks", *Wiley Series in Telecommunication and Signal Processing*, John Wiley & Sons Inc., 1994.
17. J. Muhonen and R. Durst, "Space Communications Protocol Standards (SCPS) FY97 DOD", *MITRE Technical Report*, The MITRE Corporation, 1998
18. R. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", *RFC 3168*, http://tools.ietf.org/html/rfc3168, 2001.