SDP Memo 098:
SDP experience with the SPEAD protocol

Document number……...SKA-TEL-SDP-0000199
Document Type……MEMO
Revision……..……..1
Author……Chris Broekema
Release Date……2019-03-25
Document Classification…….…. Unrestricted

Lead Author	Designation	Affiliation
Chris Broekema		ASTRON
Signature & Date: Chris Broekema Chris Broekema (Mar 18, 2019)		

SDP Memo Disclaimer

The SDP memos are designed to allow the quick recording of investigations and research done by members of the SDP. They are also designed to raise questions about parts of the SDP design or SDP process. The contents of a memo may be the opinion of the author, not the whole of the SDP.

Table of Contents

Introduction

The data traffic pattern into the Science Data Processor (SDP) is characterized by a continuous and large number of of uni-directional data streams, emitted from the Central Signal Processor (CSP) Correlator and Beamformer. These streams are transported using 100 Gigabit Ethernet technology, which is split up into four 25 Gigabit Ethernet links into the SDP servers.

The interface between CSP and SDP, as documented in the two relevant ICDs [RD03, RD04], uses the unreliable UDP/IP protocol. On top of this, an application specific data format, SPEAD, is used.

Considering the unreliable nature of the protocols used, and the unprecedented bandwidths and distances involved, it is useful to investigate the performance and usability of the currently available SPEAD implementations. The SDP consortium has done a number of experiments using both existing and newly developed codes, which are summarized in this memo. We will only consider the SKA SDP side of the equation, which concerns itself with receiving large volumes of SPEAD packets. Sending of packets is not considered, apart from generating packets for experimental purposes.

This memo was written to address SDPCDR-374.

List of abbreviations

CASPER	Collaboration for Astronomy Signal Processing and Electronics Research
CSP	Central Signal Processor
ICD	Interface Control Document
SDP	Science Data Processor
SPEAD	Streaming Protocol for Exchanging Astronomical Data
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP/IP	User Datagram Protocol / Internet Protocol

References

Reference Documents

Reference Number	Reference
RD01	SDP Memo 046: Experiences with the SPEAD protocol, R. Tobar et al
RD02	SKA1 SDP Integration Prototype (SIP) Report, B. Mort et al
RD03	SKA1 LOW SDP - CSP INTERFACE CONTROL DOCUMENT, F. Graser et al
RD04	SKA1 MID SDP - CSP INTERFACE CONTROL DOCUMENT, F. Graser et al
RD05	https://github.com/ska-sa/spead2
RD06	*Energy-Efficient Data Transfers in Radio Astronomy with Software UDP RDMA*, Przemyslaw Lenkiewicz, P. Chris Broekema, and Bernard Metzler, *Future Generation Compute Systems, Volume 79, Part 1, February 2018, pages 215-224.* *http://www.astron.nl/~broekema/papers/FGCS-SoftiWARP/1703.07626.pdf*
RD07	https://github.com/SKA-ScienceDataProcessor/integration-prototype/blob/master/sip/science_pipeline_workflows/ingest_visibilities/recv_c/README.md

SPEAD design and implementations

The Streaming Protocol for Exchanging Astronomical Data (SPEAD) was developed as a joint project between SKA South Africa and UC Berkeley in the CASPER (Collaboration for Astronomy Signal Processing and Electronics Research) project as a single self-describing data transmission protocol to bridge the gap between different accelerator technologies. It is designed to be the default ‘on wire’ protocol in active use in the MeerKAT radio telescope.

Limitations in the current implementation

The current state of the art implementation of the SPEAD protocol is spead2 [AD05]. This uses a combination of Python and C++ to allow both the use of high level programming languages, and highly performance optimized codes.

Both the SIP experiences [RD02] and the experiments by the ICRAR team [RD01] documented some limitations in this implementation. Most notably, there appears to be no mechanism to trigger re-transmits of missing data, nor is there a timeout mechanism to give up on receiving a particular heap. While re-transmits of data at a user-level are probably not feasible considering the data rates and distances involved in the SKA, one of the two mechanisms is essential for proper functioning of a receiver pipelines. There is a configurable window, which allows the application to give up on trying to receive a partial heap once X newer heaps have been observed, which may be a suitable alternative that needs further investigation.

While by default the loss of a packet means the loss of the entire heap, there is an option to get a list of payload byte ranges that were received for partial heaps. Using this effectively requires a priori knowledge of the exact heap layout. More research into the exact ways loss of entire heaps may be avoided may be needed.

SKA1 Integration Prototype (SIP) and SPEAD

Within the SKA1 Integration Prototype, SPEAD was explored. The main goal of this exercise was to test the interface with CSP, to test the Bulk data network in the P3 prototype system, and to gain experience in achieving high throughput data communication.

A main consideration in the design of the experiment was the choice of SDP to extensively employ containers as a way to orchestrate large numbers of tasks. While this technology is extensively used, the high volume data transports in SKA1 make it challenging. While the compute overheads of containers are generally low, they do tend to use virtualized networks, with potentially large performance impacts.

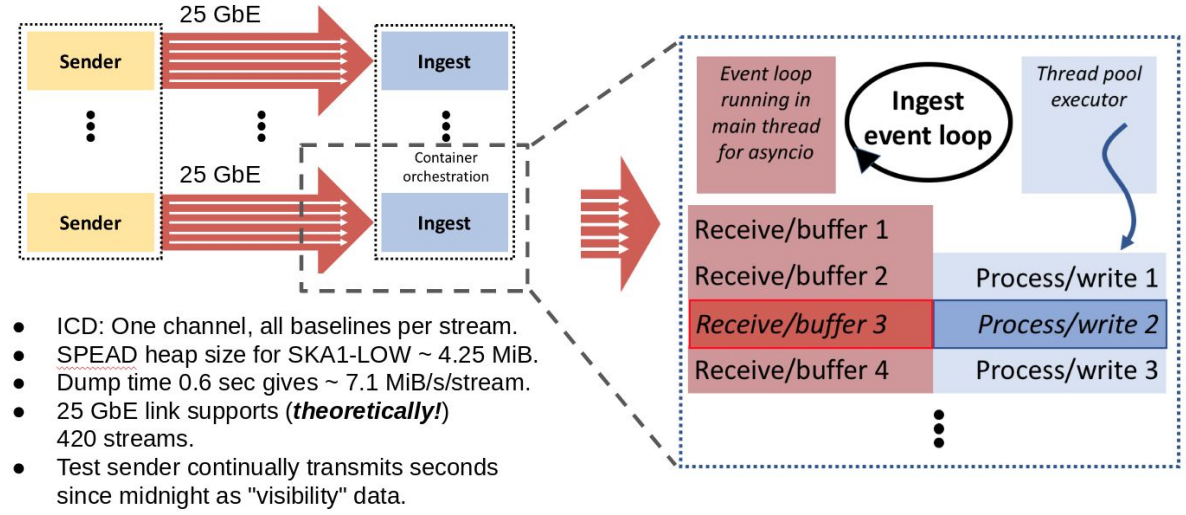

Figure 1: an overview of the SIP SPEAD experiments.

It was shown that achieving good performance is challenging and requires extensive tuning. This is a well known problem. Some particular problems were found in the overhead parsing headers. The experiment was configured to send descriptions every heap, to help facilitate quick recovery from receiver failure. While this makes the data essentially self-describing, the nature of our data flows is static, and sending the descriptions once, or once every few seconds, would significantly reduce this overhead.

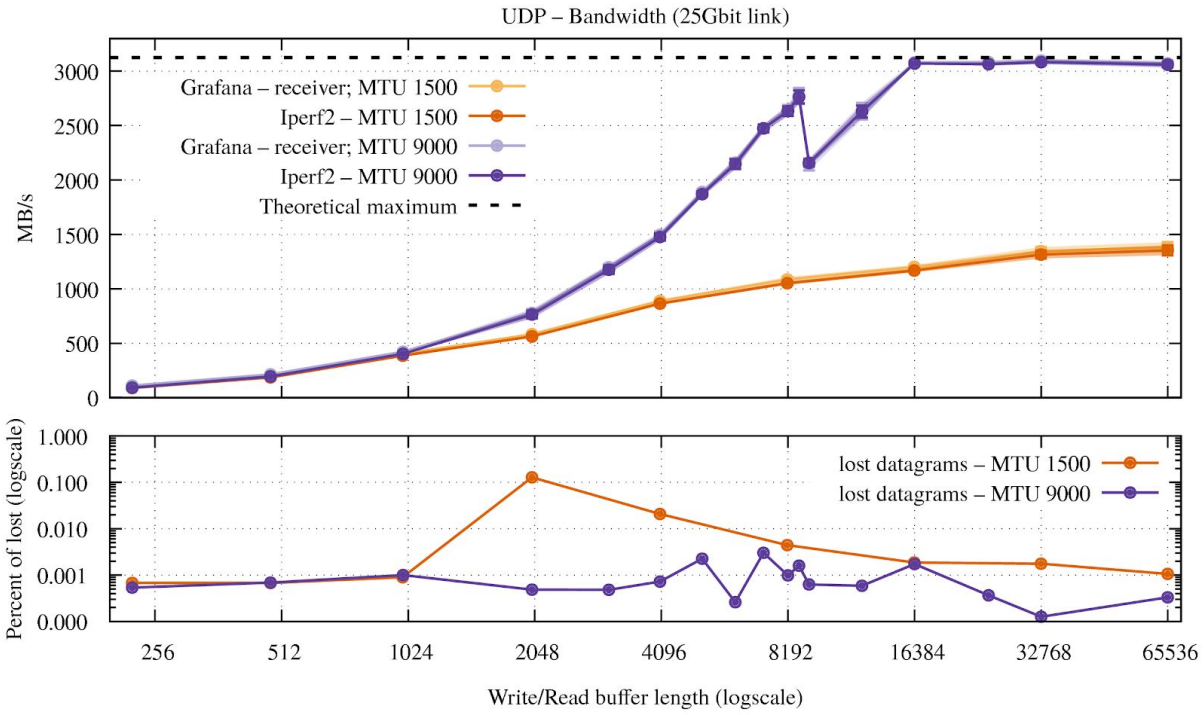

Figure 2: Bandwidth graphs showing performance of spead2 vs iperf2 and loss of datagrams for various MTU sizes.

Considering the difficulty in achieving reasonable performance with the available implementations, the SIP team implemented a raw UDP/IP receive application, that essentially skips all SPEAD parsing and only reads the packet payload [RD07]. While this negates many of the potential advantages of using SPEAD over a simpler transport protocol such as those used in for instance the LOFAR telescope, overheads are much reduced.

SPEAD experiments by ICRAR

To gain experience with the SPEAD protocol over the long distances between the SKA1 correlator and beamformer on the telescope site, and the Science data processor in Perth and Cape Town, the ICRAR team experimented with and compared performance of a local loopback device, a short-range 1 Gigabit Ethernet connection, and the existing 100 Gigabit Ethernet link between Murchison Radio Observatory and Perth that is close match, both in distance and used technology, to the expected SKA1 links. The experiments were designed to measure performance and loss of data using SPEAD over UDP/IP using the spead2 code [RD05]. Furthermore, a new TCP/IP based SPEAD implementation was added and extensively tested.

We will not reproduce the results obtained by the ICRAR team here, but instead refer to their excellent memo on the subject [RD01]. Figure 3 shows the results of their final experiment over a long-distance SKA1 like 100 GbE link.

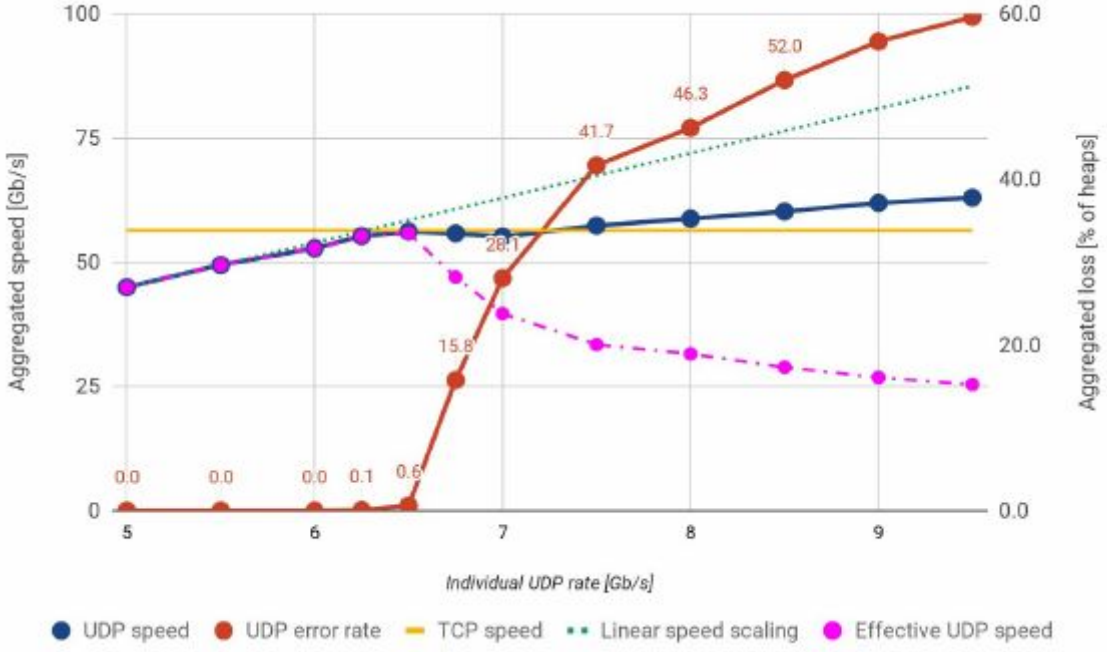

Figure 3: Performance of all nine 10 [Gb/s] connections being sent through the same 100 [Gb/s] link between MRO and Perth. Dark blue points are aggregated UDP speeds, red dots are UDP error rirates, the green dotted green line is the linear scaling of the individual UDP rates, and the yellow line is the aggregated TCP speed. Like in the previous experiment, UDP aggregated speeds stop scaling and saturate at around 64 [Gb/s]. Aggregated UDP speeds with <1% data loss rates again agree with the aggregated TCP speed with zero loss. The purple line represents the ‘effective’ UDP transfer speed as observed by the receiver, which is defined as the total volume of heaps received divided by the total transfer time (or, equivalently, the aggregated UDP speed times the UDP error rate).

An important conclusion from the ICRAR report is that the long-distance 100 GbE link is saturated at about 64 Gbps. This represents a load factor of 64%, which is significantly lower than the maximum load factor per link of 80% currently specified in the ICD. While the document does not go into much detail on the cause of this limit, it is noteworthy that both TCP/IP and UDP/IP reach the same maximum effective throughput, suggesting this is an infrastructural limit, either in the network, or on the receiving node, not one based by the protocol per se. Nevertheless, this is clearly an issue that warrants further investigation.

Further discussions with the ICRAR team revealed that the limiting factor may have been the Linux kernel on the receiving node not being able to keep up with the data stream. This is a well known problem that has been extensively studied (but not well documented) in the past. The spead2 code includes the ability to bypass the Linux kernel with Remote Direct Memory Access (RDMA) using either ibvers or netmap, although the latter should be considered abandoned. While this offers extensive performance and energy consumption advantages (see [RD06] for a discussion on the latter), this may put hardware constraints on the network interface cards on the receiving nodes. The ibverbs implementation is only tested on Mellanox ConnectX-based cards, and while others may work, this is not guaranteed.

It is also important to note that RDMA based communication bypasses kernel safety features Traversing the IP stack is expensive because of the strict separation between kernel space and user space. RDMA gives remote machines direct access to user space memory, which is only viable in a trusted environment. However, the performance benefits are enormous, and the use of this technique is highly recommended.

They also conclude that the use of UDP/IP offers no benefit and would require careful tuning to avoid loss of data and optimal data throughput. The author of this memo would like to note that the choice for UPD/IP over TCP/IP is likely driven by the implementation effort on the FPGAs in the CSP correlator and beamformer, rather than purely on the basis of expected superior performance. The reliable nature of the TCP/IP protocol means that the sending peer has to keep each sent packet in memory until confirmation of receipt is received, which takes up valuable and sparse memory resources. Furthermore, TCP/IP is much more complex than UDP/IP to implement, driving up firmware development time and cost.

The bandwidth limit shown is likely to be caused by the inability of the kernel to keep up with the data steam, as mentioned above. This may also have caused the loss of packets, as demonstrated in the memo. It would be wise to repeat the experiments, taking particular care in turing kernel performance. Loss of packets may be considerably less, as is the experience with the LOFAR data transmission system.

Work external to SKA SDP

SPEAD is in production use in the MeerKAT array, but apart from mentions in slides on the engineering of the telescope, no documented experience is available. We've consulted with the SKA South Africa team in the preparation of this memo, and some of the suggestions in this memo were made by that team.

In general, anecdotal evidence suggests that the SPEAD protocol works well within the MeerKAT telescope, and is in widespread use. High performance implementations are available but the knowledge and know-how is currently concentrated in persons, not documents. It was suggested that this will possibly change during bridging, but this would require active effort and tracking from the side of SKA. While we consider this is a slight risk, our questions on the subject were promptly and openly answered, and, as shown in the acknowledgement section, feedback from the SKA South Africa team was given on this memo.

Summary and conclusions

Considerable work has been done in the SDP consortium to gain experience with the SPEAD protocol and to exercise the available implementations and the SKA interfaces that use this protocol. Some challenges have been identified, and some potential solutions have been offered. Some of these are:

- No retransmit or timeout mechanism for lost packets
  - Use configurable window of observed received packets instead
- Loss of a packet means loss of an entire heap
  - Reconstruct partial heaps with a priori information and list of byte ranges received
- High overhead of parsing headers
  - Only send headers every couple of seconds instead
- Significant loss of packets at high occupancy of links using UDP/IP
  - Use TCP/IP instead (this may require significant work on the part of CSP Correlator and Beamformer though)
- Inability to fully utilise the available bandwidth
  - Kernel performance on the receiving node: use RDMA instead
  - Alternatively: read raw data packets, as shown by SIP [RD07]

While some problems have been identified, and solutions have been offered, significant work still remains. Chief of these is that all experiments were done using CSP emulators, and not FPGA based hardware. As soon as suitable hardware is available this should be corrected.

It is recommended that such experiments are done on a multi-switch test environment to more accurately simulate the complex system the SKA will be. Furthermore, we recommend that the features available in the spead2 implementation, as identified in this memo, are further explored.

Acknowledgements

We thank Fred Dulwich, Bruce Merry and Rodrigo Tobar for their feedback and suggestions on this memo.