REAL-TIME TRANSPORT PROTOCOLS FOR TELEMETRY DATA AND SIGNALING

Nikki Cranley, Ph.D., Diarmuid Corry, M.Sc.
ACRA CONTROL INC., Maryland, USA

ABSTRACT
There are many network transport and application layer protocols that have been developed and designed for the delivery of time-sensitive and critical data over best-effort Ethernet networks. Choosing the appropriate protocols is crucial to ensuring the performance of the network for data transmission. Network layer protocols, such as TCP/IP or UDP/IP, determine the reliability of the data transmission while application layer protocols are designed to enhance the services of the underlying transport protocol. This paper will compare the Quality of Service (QoS) tradeoffs for both UDP and TCP. In this paper it is shown that UDP or TCP alone is insufficient to meet the transmission requirements of telemetry data and to overcome these limitations, an application layer protocol is required to enhance the delivery of real-time sensitive data. This paper presents the Real-time Transport Protocol (RTP) and proposes a RTP telemetry profile payload definition that provides complete payload self-description and up to 98% packetization efficiency.

1. INTRODUCTION

The Integrated Network-Enhanced Telemetry (iNET) initiative endeavours to define open standards for the interoperation of networked enabled systems [1]. The iNet working groups embrace the use of standard Ethernet-based network technologies and network protocols for data transmission and signaling. This paper is structured into two key sections: the first investigates the network layer IP protocols for transmitting data and compares the suitability of TCP/IP versus UDP/IP for data transmission. The second section of this paper describes the application layer protocol, the Real-time Transport Protocol (RTP), for telemetry data transmission.

There are two key network layer protocols used for the transmission of data. These are the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP provides reliable data transmission by defining acknowledgement, lost packet retransmission, and congestion control algorithms. UDP, on the other hand, is a simple best-effort service that does not provide any packet delivery guarantees. From the outset, TCP may seem like a preferred transport layer protocol, however there are a number of decidedly significant trade-offs between the use of TCP or UDP. Application layer protocols are designed to enhance the services of the underlying transport protocol (i.e. TCP/IP or UDP/IP) by adding application-specific features and services. One such application layer protocol is the Real-time Transport Protocol (RTP) that has been designed for the transmission of time-sensitive packet based data. RTP is the predominant application layer protocol employed for VoIP, video conferencing and video streaming services. RTP was purposely designed to be incomplete so that it can be adopted by different applications with real-time data transmission requirements. In this way it can be adapted to meet the unique requirements of telemetry data transmission whilst still being conformant to the standard through the definition of an RTP telemetry payload profile. This paper will show how the RTP protocol can be used for the transmission of telemetry data.

The remainder of this paper is structured as follows: Section 2 describes the challenges for packet-based real-time data transmission over Ethernet and the two predominant protocols for data transmission, that is TCP and UDP. Section 3 provides a comparison of TCP versus UDP for data transmission and show that there are negative behavioral aspects associated with the use of TCP for real-time data transmission. The following section outlines why UDP should be chosen as the underlying transport protocol. Section 5 provides and overview of the Real-time Transport Protocol (RTP) that was designed to enhance the UDP transport layer for real-time data transmission. RTP was designed so that it can be adapted to the needs of the application in question through the definition of RTP payload profiles. An RTP telemetry profile definition is proposed that provides a flexible and efficient means of carrying telemetry data in terms of parameters and samples.

2. Challenges for Real-time Data Transmission

The Ethernet Medium Access Control (MAC) is a best-effort networking technology and does not provide any guaranteed Quality of Service (QoS) [2]. The Ethernet makes a "best-effort" to deliver error-free frames. If the network is heavily loaded, collisions or bit errors may occur during transmission resulting in the Ethernet frame being discarded. It is the responsibility of the higher network and transport layers of the Open Systems Interconnect (OSI) reference model to overcome such events.

2.1. Best-Effort Delivery

There are many factors, which affect the delivery of real-time data traffic over best-effort Ethernet networks. These are heterogeneity; congestion; delay and jitter due to congestion, packet loss and/or retransmissions; and loss due to congestion.

There are two kinds of heterogeneity, namely, network heterogeneity and receiver heterogeneity. Network heterogeneity refers to the subnets in the network having different capacities and resources (e.g., processing, bandwidth, storage and congestion control policies). Network heterogeneity can cause each end-device to have different packet loss/delay characteristics. This is a problem for designing or predicting behavior of transmission over the network, as the transmitting device cannot make any assumptions about how the network will treat a particular packet. For example, if an application used prioritization of packets by using the Type Of Service (ToS) field in the IP header, various subnets in the network may interpret this ToS field differently. Receiver heterogeneity refers to end devices having different bandwidth requirements and/or processing capability.

Congestion occurs when the amount of data in the network exceeds the capacity of the network. As traffic increases, routers and switches are no longer able to cope with the load and this results in lost packets. Congestion can be caused by several factors.

- Queues can build up caused by high data rate devices. If there is not enough memory to hold all the data at the switch, packets will be lost. But even if queues had an infinite length, this cannot eliminate congestion, as in the case of TCP, by the time a packet is at the top of the queue, the packet may have already timed out and duplicates may have already been sent. All these packets are then forwarded onto the next router, increasing the load all the way to the destination.

- Slow processors can also cause congestion. If the router's CPU is slow at performing its tasks of queuing buffers, updating tables etc. queues can build up even though there is excess capacity.
- Bottleneck bandwidth links cause congestion.

There are many sources of jitter and delay in any networked transmission system. These include device delay and network delays. In the network itself, in addition to propagation delays, further delays are incurred at each switch along its path due to queuing and switching at various nodes in the network. At the end-points, delays are incurred in obtaining the data to be transmitted and packetizing it. Real-time data is particularly sensitive to delay, as the data packets require a strict bounded end-to-end delay. That is, every telemetry data packet must arrive at the receiving device with enough time to unpack the data, extract the parameters, process, record, and/or display the data. A variable delay on the IP networks is known as jitter. Jitter is mainly due to queuing and contention of packets at intermediate switches, but can also happen when packets take a different path to the destination.

Ethernet does not provide any guarantee with respect to data loss – it is "best-effort" and packets may be lost at any time. Real-time data applications (such as telemetry) generally have some low loss tolerance (say 0.01%) and a fixed latency (say 200ms) in which a packet must arrive. So, even if the network allows for re-transmission of lost packets, for example with TCP flows, the retransmitted packet must arrive before its processing time. Congestion at switches in the network often results in queue overflow, which generally results in packets being dropped from the queue. Often consecutive packets in a queue are from the same data source. So, when packets are being dropped, they often belong to the same source. Thus, during periods of congestion, this packet queuing characteristic manifests as bursty losses and delays. For instance, if packet *n* has a large delay, packet *(n + 1)* is also likely to do so. A similar effect occurs for packet loss.

2.2. Network Layer Protocols

TCP is a reliable connection-oriented protocol that guarantees the delivery of the data packets [3]. TCP comprises flow-control and congestion control mechanisms that regulate the senders' transmission rate to the available bandwidth and render the connection reliable by retransmitting lost packets. The transmission rate of TCP is bounded by a parameter called the congestion window, Cwnd. A TCP source may transmit Cwnd packets then it must wait until it has received an acknowledgement, ACK, for the transmitted packets.

TCP packets comprise a sequence number field in the header, which facilitates loss detection. To ensure reliable data delivery, the receiver acknowledges the receipt of TCP packets. Reliability in TCP is achieved through its ability to detect failed transmissions and re-sending lost packets. However, TCP cannot guarantee transmission since underlying TCP is IP, which is unreliable. There are two possible loss events in TCP that require retransmissions, the TCP data packet may be lost or the ACK may be lost.

UDP, on the other hand, is a lightweight connectionless protocol that does not guarantee the delivery of packets, nor does it implement any flow control or congestion control mechanisms [4]. UDP is considered a "greedy" protocol in that it transmits at the desired rate irrespective of network capacity or other competing traffic flows. For example, a UDP flow that consumes 75% of the link capacity will starve competing TCP flows forcing them to share the remaining 25% of the link capacity since the TCP flows adapt to the available capacity.

3. COMPARATIVE ANALYSIS OF TCP VERSUS UDP DATA TRANSMISSION

The ns2 simulator [5] was used to compare the performance of TCP and UDP using the same network topology as shown in Figure 1. The number of transmitting data sources was increased by one every 10 seconds until a maximum of 10 data sources was reached. The capacity of the Bottleneck link was limited to 10Mbps, making it easier to saturate the bottleneck link using fewer data sources. The switch is a typical store and forward switch with 10us switching delay. Two test cases are compared: all data sources transmit data using TCP; all data sources transmit data using UDP. The File Transfer Protocol (FTP) application was used as a source for TCP data packets. TCP will attempt to maximize utilization of the available network capacity and minimize the effects of congestion by dynamically adapting its congestion window. The UDP traffic is generated as a Constant Bit Rate (CBR) data flow with an offered load of 1Mbps per UDP data source. In both cases the packet payload size was 552B. In this section the QoS at the bottleneck link is investigated.

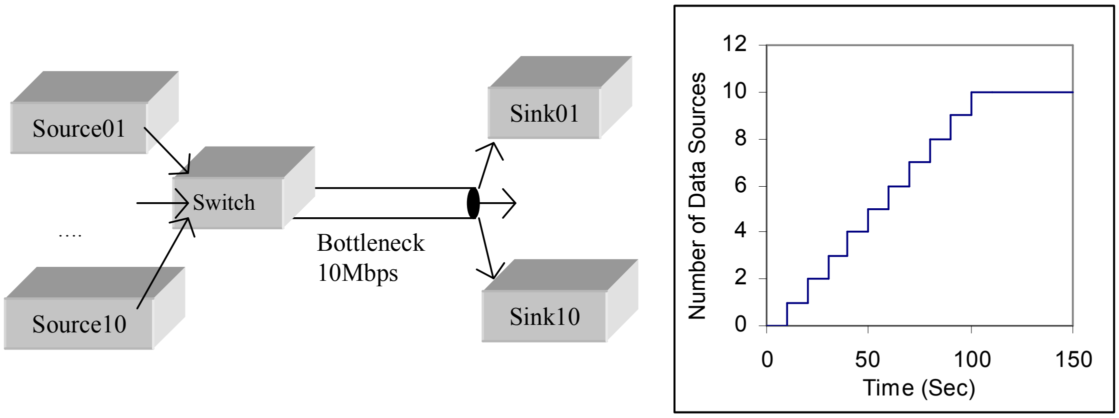
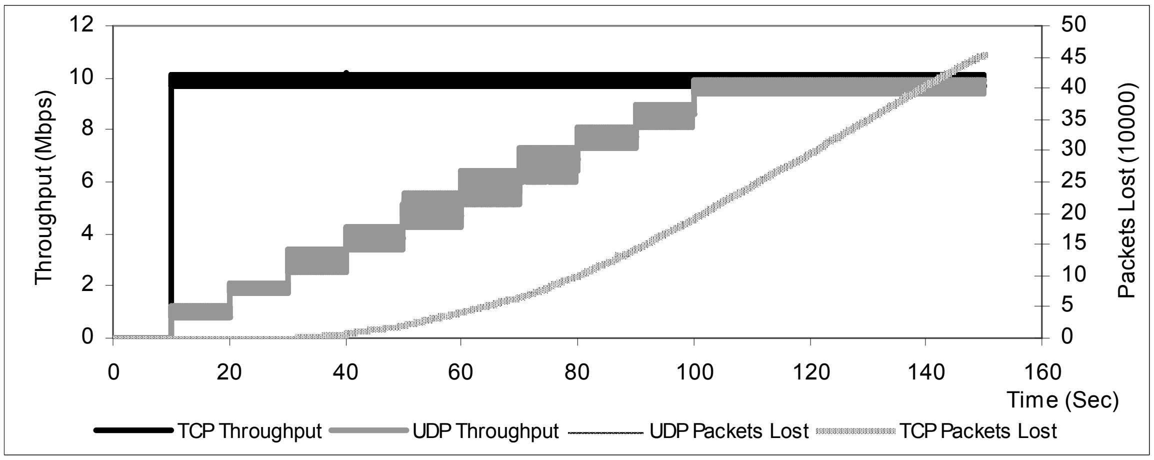
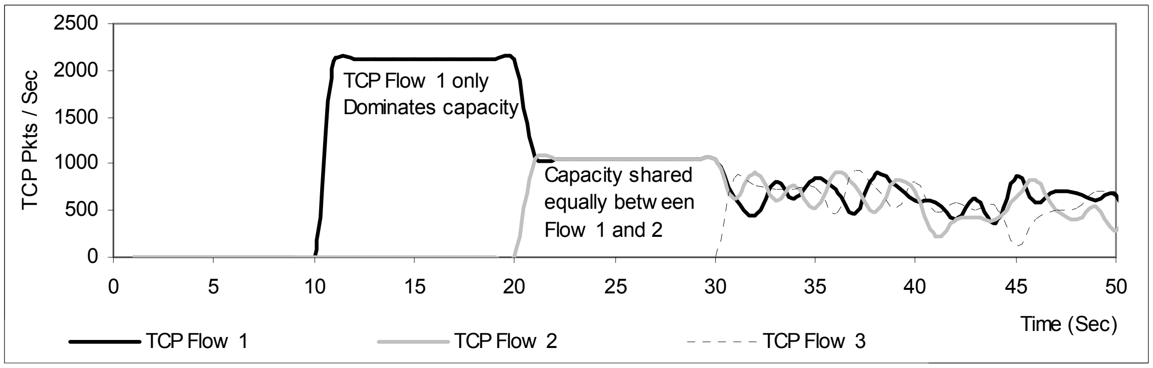


Figure 1 Network Topology

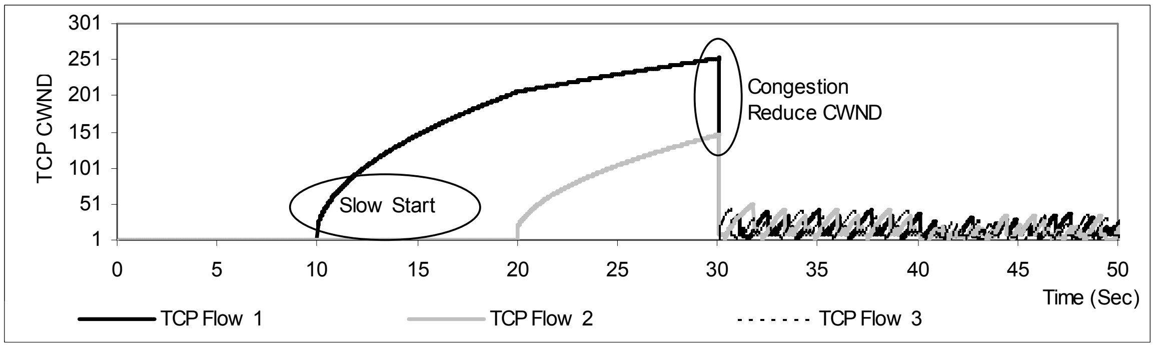
A comparison of UDP throughput versus TCP throughput for the same test configuration is shown in Figure 2(a). It can be seen that there is a fundamental difference in behaviour between UDP and TCP data flows. UDP is not concerned with the capacity of the network and will transmit its data packets at the required rate regardless of network conditions or the reliability of packet delivery. As more UDP data sources are added, the network capacity is increased. In this scenario, no UDP packets are lost since the aggregate load of the UDP data sources is equal to the capacity of the bottleneck link. In contrast, TCP probes the network by gradually increasing its Cwnd in order to maximize utilisation of available network capacity. Effectively, through probing TCP induces congestion resulting in packet loss and which requires subsequent packet retransmissions, shown in Figure 2(a). This TCP maximisation of network utility can be seen during time interval 10-20sec when there is only one active TCP flow transmitting data, shown in Figure 2(b). This TCP flow quickly maximizes the total capacity of the 10Mbps bottleneck link by increasing its Cwnd, shown in Figure 2(c). At time 20sec, a second TCP flow is started. The bottleneck has sufficient capacity to allow both TCP flows to co-exist with both flows equally share the available capacity, and neither flow experiences the onset of congestion. However, at time 30sec, a third TCP flow is started, which induces congestion to occur in the bottleneck resulting in packet loss in the three active TCP flows. Packet loss and packet retransmission causes the congestion control mechanism in the affected TCP flows to reduce their respective Cwnd, as shown at time 30sec in Figure 2(c). It can be seen that TCPs congestion control mechanism allows TCP to maximise network utilisation and allow competing TCP flows to equally share the available link capacity, however, its adaptive behaviour introduces stability issues and per flow fluctuations.



(a)



(b)



(c)

Figure 2 (a) Comparison of TCP and UDP (b) Close-up of TCP Throughput (c) Close-up of TCP Cwnd

4. Application Layer Protocols

It is evident from the comparison of TCP and UDP, that TCP is not suitable for real-time data transmission for three key reasons:

1. TCP multiplicatively decreases the Cwnd window size on encountering packet loss. A sudden decrease of the window size may cause a sharp bit rate variation, which can seriously degrade the transmission rate of a DAU.
2. Real-time data is time sensitive so the retransmission of lost packets is not very efficient since it is unlikely that the retransmitted packet will arrive at the receiver in time for processing in a congested network.
3. Per packet positive acknowledgements impose a large network overhead. Negative acknowledgements would be more efficient in this case.

However, the alternative, UDP, has significant drawbacks for continuous real-time traffic as there is no guarantee that the packets will arrive, packets may by lost, misordered or delayed, nor do UDP packets have sequence numbers which prevents packet loss (or out of order) detection. While these limitations may be overcome in particular cases (e.g. when the network is stable and the traffic is known, bounded, and less than the network bandwidth), in the general case an Application Layer Protocol is required to run over UDP/IP to compensate for these limitations.

5. Real-time Transmission Protocol (RTP)

RTP was designed for the transport of real-time data [6, 7] and is used for a wide variety of real-time networked audio and video applications. It is a mature, widely accepted and standardized protocol. RTP consists of two closely linked parts: the Real-time Transport Protocol (RTP) to carry data that has real-time properties and the RTP Control Protocol (RTCP) to monitor the QoS of the RTP data transmission. RTP was designed to be flexible enough to provide the relevant information required by a real-time application and deliberately designed to be incomplete so that it can be tailored through modifications and/or additions to the headers as needed to make it more specific to its application. In this way, a common base packet format can be used to provide interoperability between devices while the header extensions can provide more sophisticated processing capabilities. Since RTP typically runs over UDP/IP, RTP does not provide any QoS guarantees in terms of reliable delivery, delay, or sequential delivery, however, RTP enhances UDP by providing the following services/facilities:

- Adds sequence numbers and timestamps facilitating the decoding of the payload.
- Includes payload type and source identification.
- Multiplexing services.
- Checksum services.
- Delivery monitoring.
- Loss detection.
- Intra- and Inter stream synchronization.

An RTP session consists of one or more RTP streams and each RTP stream may be coupled with its own RTCP stream. RTCP is based on the quasi-periodic transmission of feedback and control information Report Packets associated with an RTP stream. This feedback allows senders and receivers to diagnose faults in the network and determine whether those faults are local or global on the network. An RTCP Report Packet begins with a fixed part similar to that of RTP data packets, followed by structured elements. The structured elements have an alignment requirement to make RTCP packets "stackable". The defined structured elements in RTCP Report Packets include Sender Reports (SR); Receiver Reports (RR); Source DEScription (SDES) i.e. a description of the data source; BYE; APPlication-specific (APP), which are designed for application specific functions. The SR and RR contain network-level statistics information including timestamps to allow one-way delay measurements, transmitted/received packet count, transmitted/received byte count, packet loss measures, delay, and inter-arrival jitter.

5.1. Interpretation of the RTP Standard Header for Telemetry

An RTP packet consists of a fixed header, a list of contributing sources and the payload data. The RTP packets have a minimum 12B header as shown in Figure 3.

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
V		P	X	CC				M	PT							SEQ															
RTP TIMESTAMP																															
SSRC																															
CSRC_1																															
CSRC_n																															

Figure 3 RTP Standard Header

- **Version (V) -** This field identifies the version of RTP.
- **Synchronization Source (SSRC): The source (DAU) of a stream of RTP packets**, identified by a 32-bit numeric SSRC identifier carried in the RTP header so as not to be dependent upon the network address. All packets from a synchronization source form part of the same timing and sequence number space, so a receiver can group packets by synchronization source for processing.
- **Sequence number:** The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.
- **RTP timestamp: The timestamp is the sampling instant of the RTP data packet.** More details of the RTP timestamp can be found in [6].
- **Marker (M):** The interpretation of the marker is defined by a profile. It is intended to allow significant events to be marked in the packet stream. **For networked telemetry the marker bit indicates the first packet generated at the start of an Acquisition cycle.**
- **Payload type (PT):** This field identifies the format of the RTP payload and determines its interpretation by the application. For example, a set of default mappings for various encoding audio and video formats is specified in RFC 3551 [7]. **A profile specifies a default static mapping of payload type codes to payload formats. The next section describes a telemetry payload profile.**
- **Padding (P) -** If the padding bit is set, the packet contains one or more additional padding octets at the end, which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored, including itself. **Padding may be needed by some encryption algorithms with fixed block sizes or for carrying several RTP packets in a lower-layer protocol data unit.**
- **Extension (X):** If the extension bit is set, the fixed header MUST be followed by exactly one header extension.
- **Contributing Source (CSRC) count (CC):** The CSRC count contains the number of CSRC identifiers that follow the fixed header. This is only required if RTP streams are multiplexed or “mixed”.

- **CSRC list:** 0 to 15 items, 32 bits each. The CSRC list identifies the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 can be identified. CSRC identifiers are inserted by mixers using the SSRC identifiers of contributing sources. **Mixer:** An intermediate system that receives RTP packets from one or more sources and processes/multiplexes the packets in some manner and then forwards a new RTP packet. During this mixing stage, the SSRC information of mixed packets is moved to the CSRC field and the Mixer itself becomes the SSRC of the mixed packet. When there is no mixer, the CSRC list is empty and the CC value is 0.

To summarize, the benefits of using RTP for real-time telemetry data include:

- Network independent identification of RTP telemetry data sources.
- RTP provides packet tracking through sequence numbers.
- Ability to synchronize telemetry data streams with other RTP payloads using RTP timestamps.
- Monitoring data delivery performance through RTCP.
- Combining real-time data streams received from multiple DAUs into a set of consolidated streams through RTP mixers.
- Converting data types, etc. through the use of RTP translators.

5.2. RTP TELEMETRY PAYLOAD PROFILE DEFINITION

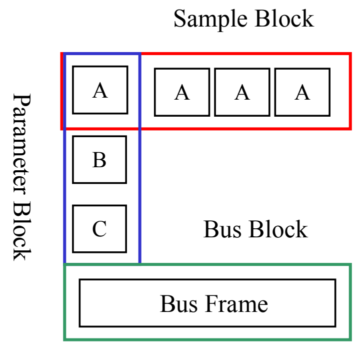
In RTP, each application and/or encoding format has a profile definition, which describes how the data is structured in the RTP packet. This section suggests how such a profile might look for a telemetry packet.

To ensure inter-operability, telemetry RTP packets should be self-describing i.e. no other sources of information are required to monitor a particular telemetry data stream. The RTP Telemetry Payload Profile must be able to handle three data packing scenarios as shown in Figure 4:

- **Sample Block**: 1 Parameter with N Samples (1:N)
- **Param Block**: N Parameters and 1 Sample (N:1)
- **Bus Block**: Parsed data i.e. a captured bus frame (1:1)

Any combination and number of these structured blocks can be contained within the RTP packet. Each block begins with a Common Block Header. The Common Block Header (CBH) consists of:

- **Block Type (BT)**: indicates the type of block i.e. Sample Block, Param Block or Bus Block.
- **Length**: describes the length of the block in Bytes thereby ensuring that the start of the next telemetry block can be found in the packet where more than one block is contained in a packet.
- **Status**: status fields for the telemetry data block. Status information shall include the Data Synchronicity, Time type and so on.
- **Time**: Time of the first sample regardless of parameter type in the packet. The time format is determined by the time type contained in the Status byte. Using NTP or PTP timing format, the timing information is stored in two 32bit parts, a second part and fractional part.



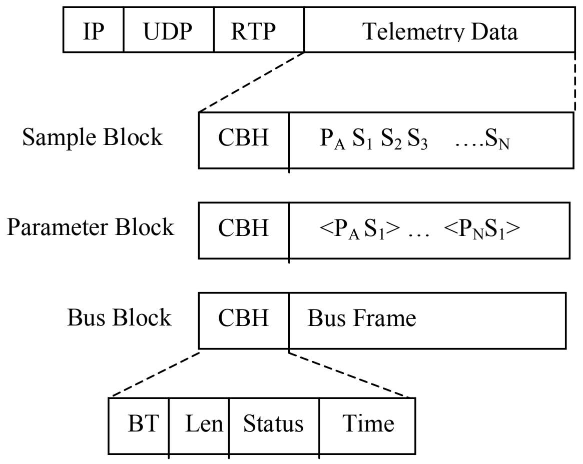


Figure 4. RTP Telemetry Packing Scenarios

Figure 5. RTP Telemetry Payload Block Structure

Following the CBH, the block instance is described. A Sample Block needs to contain only a Parameter ID, sampling rate field, followed by *N* 2B samples of the described parameter. A Parameter Block is less efficient since it must describe *N* parameter-sample pairs costing *N* times (Parameter ID and 2B Sample value). The Bus Block consists of a Bus ID followed by the captured bus frame. The different block structures and encapsulation of the telemetry data is shown in Figure 5.

5.3. RTP PAYLOAD SELF-DESCRIPTION AND EFFICIENCY TRADEOFF

Although RTP and the Telemetry Profile impose an additional header overhead, the meta-data contained therein offers numerous advantages for telemetry data transmission, processing and recording. The RTP SSRC allows for the network to contain 2^{32} unique telemetry data streams and each SSRC may contain 2^{16} unique parameter definitions within the RTP stream. To demonstrate the efficiency of this protocol, the packing efficiency is defined as the number of samples that can be packed into a packet divided by the maximum number of samples that can be packed into a UDP datagram with no parameter meta-data. For example, given a maximum sized Ethernet packet of 1500B, the maximum UDP payload is 1458B. Assuming each sample is 2B, a maximum of 729 samples can be contained in the payload of the UDP packet without any content description yielding 100% packing efficiency. Using only the RTP standard header of 12B, the maximum number of samples that can be contained in the payload is reduced to 723 samples/packet yielding 99.2% efficiency. Assuming that the RTP payload profile definition contains a Telemetry Common Block Header size of 20B and Parameter IDs are two 2B then:

- Sample Block efficiency: UDP Payload 1458B – (12B Standard RTP header + 20B Telemetry Common Block Header + 2B Parameter ID + 2B Parameter Sampling Rate) = 712 samples resulting in 97% packing efficiency.
- Parameter Block efficiency: UDP Payload 1458B – (12B Standard RTP header + 20B Telemetry Common Block Header + 2B Parameter ID + 2B Sample value) = 356 Parameter-sample pairs resulting in 49% packing efficiency.
- Bus Block efficiency: UDP Payload 1458B – (12B Standard RTP header + 20B Telemetry Common Block Header + 2B Bus ID) = Maximum encapsulated bus frame length is 1426B.

The Sample Block captures the best-case parameter-sample packetization strategy while the Parameter Block captures the worst-case packing strategy. By using combinations of Sample, Bus, and Parameter block structures within the RTP packet using the Telemetry Profile definition, a packing efficiency of up to 97% can be achieved. To further reduce the packet header overhead, the RObust Header Compression (ROHC) algorithms can be used [8] to compress the RTP/UDP/IP headers typically from 40B to 4B [9].

6. CONCLUSIONS

In this paper we have described the application of the Real-time Transport Protocol (RTP) for the packetization of time-sensitive telemetry data in a networked environment.
The key points in this paper are:

1. The Ethernet is a best-effort network transmission technology. There are many challenges for best-effort networks, namely heterogeneity, congestion, delay, jitter, and loss.
2. Network layer protocols, such as TCP or UDP, are used to improve upon "best-effort" Ethernet.
3. Through network simulation, this paper shows that TCP is not suitable for real-time data and as such UDP should be used.
4. UDP alone is insufficient to adequately describe real-time telemetry data. A telemetry-specific application layer protocol is required to enhance UDP to meet the transmission requirements of real-time telemetry data.
5. The Real-time Transport Protocol (RTP) is a standard generic protocol designed for real-time data transmission.
6. RTP consists of profiles and payload type definitions that describe how the data is packetized for different applications and environments.
7. This paper introduces an RTP payload profile definition for telemetry that allowed multiple samples, multiple parameters, and captured bus frames to be packetized in a structured, efficient and self-describing manner.
8. Despite an increased header overhead using RTP with the telemetry payload profile definition, a 97% Parameter-sampling packetization efficiency can be achieved.

REFERENCES

[1] iNET, "Telemetry Network System Architecture", v1.0, May 2004
[2] C. E. Spurgeon, "Ethernet, the Definitive Guide", O'Reilly & Associates, ISBN 1-56592-660-9
[3] IETF, RFC 0793/STD0007, "Transmission Control Protocol", Sept 1981
[4] IETF, RFC 0768/STD0006, "User Datagram Protocol", Aug 1980
[5] Network Simulator (NS-2) [Online April 2008], www.isi.edu/nsnam/ns/
[6] IETF, RFC 3550/STD0064, "RTP: A Transport Protocol for Real-time Applications", July 2003
[7] IETF, RFC 3551/STD0065, "RTP Profile for Audio and Video Conferences with Minimal Control ", July 2003
[8] IETF, RFC 5225, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", April 2008
[9] ROHC, [Online April 2008], http://www.rohc.net/links.html