2015 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) TECHNICAL SESSION
AUGUST 4-6, 2015 - NOVI, MICHIGAN

EVALUATION OF DETERMINISTIC ETHERNET FOR THE MODULAR ACTIVE PROTECTION SYSTEMS FRAMEWORK

David A. Verbree, PhD
DornerWorks Ltd.
Grand Rapids, MI

Andrey Shvartsman
U.S. Army TARDEC
Warren, MI

ABSTRACT

The U.S. Army Tank-Automotive Research, Development and Engineering Center (TARDEC) contracted DornerWorks Ltd. to evaluate Ethernet-based networking protocols for the safety-critical RDECOM Modular Active Protection Systems (MAPS) framework (MAF). The MAF requires a universal and robust high-speed communication network that can transmit heterogeneous data at near gigabit speeds in a deterministic fashion with bounded and predictable latency. The objectives were to evaluate candidate protocols through rigorous stressing scenarios to: 1) assess and estimate upper bound of performance including data throughput and reliability; and, 2) detect and identify causes and conditions of data loss or corruption. We assessed four protocols: SAE AS6802 (TTEthernet; TTE), ARINC664p7 (rate-constrained; RC), COTS UDP integrated with these two protocols (best-effort; BE), and UDP on a COTS network under three levels of network saturation and with varying payload sizes. On an unsaturated network, TTE had the greatest one-way latency (≥595.74µs) and variance of latency (s.d. ≥ 275.56 µs) from the applications' perspectives due to lack of synchronization between the network cycle time and the application transmittal period; RC and BE had low and nearly identical latencies (40.23- 452.09 µs) that increased strongly with payload size; and COTS UDP had, by far, the lowest latency (≤92.69) and was relatively unaffected by payload size. On a saturated network, TTE latency increased by 17.3-44.2% , matching the configured virtual link period; RC latency also increased slightly; BE latency increased substantially (>21 ms); UDP latency approximately doubled for larger payload sizes. Surprisingly, with 100% saturation on a full-duplex switched COTS network, COTS UDP had no lost packets except those originating from the saturating devices..

INTRODUCTION

The goal of this study was to evaluate Ethernet-based networking protocols for the safety-critical RDECOM Modular Active Protection Systems (MAPS) framework (MAF). One of the objectives of the MAPS program is to incorporate communications network protocols that enable a modular, flexible, and scalable approach to integrating active protection system (APS) technologies for ground vehicle platforms, and to lower lifecycle costs. MAPS is developing a Modular APS Controller (MAC) that supervises APS subsystems and ultimately determines the vehicle's response to an inbound threat. The MAF and MAC require a universal and robust high-speed communication network that can transmit heterogeneous data concurrently with data prioritization and/or bandwidth allocation. Due to the safety critical nature of the APS domain, the network protocol must be deterministic with bounded and predictable latency. The goal of the present study is to evaluate high-speed network protocols capable of attaining near gigabit speeds for the MAF while meeting the determinism requirements. The objectives were to evaluate candidate protocols through rigorous stressing scenarios to: 1) assess and determine bounds on performance including data throughput and reliability; and, 2) detect and identify causes and conditions of data loss or corruption.

METHODS

Protocols evaluated

We conducted a literature review to select potential candidate protocols for this study. Only ARINC 664p7 [1], SAE AS6802 [2] (Time-Triggered Ethernet; TTE), and IEEE 802.1BA [3] (Audio Video Bridging; AVB) met our criteria for ≥500 Mbps throughput, determinism, and industry support/adoption. Of these three, AVB was eliminated from consideration because it may become obsolete for this intended application once implementations of Time-Sensitive Networking [4] (TSN) are available. TSN is a collection of open standards that augment AVB by providing time-synchronization (IEEE 802.1AS [5] based on IEEE 1588 [6]), frame transmission scheduling [7] (IEEE

802.1Qbv), and other enhancements. The switches and end-points utilized in the evaluation effort simultaneously support TTE, ARINC 664p7, and standard UDP (User Datagram Packet; RFC 768 [8]) protocols. Based on this review, for our study we evaluated four protocols: TTE, ARINC 664, p7 (rate-constrained; RC), UDP over COTS network, and UDP (best-effort; BE) over the TTE network.

Test bed hardware and architecture

Our development system consists of two TTE 12-port 1 Gbps development switches and four development PCs (Dell Precision T1700 with Intel® Dual-core I3-4130 3.4 GHz processors) with TTE PCIe cards. The development PCs came pre-installed with Ubuntu® 14.04 LTS and Linux kernel version 3.13.0-36-generic. An additional PC (‘testmaster’; Dell® Precision T3610 with an Intel® Xeon® Quad-core E6-1620 v. 2, 3.70 GHz processor4), with a quad-port COTS NIC that supports high-resolution hardware time-stamping (Intel® Ethernet Server Adapter I350-T4), and a COTS 16-port Gigabit switch (Netgear® ProSAFE, model GS116E) were purchased to coordinate tests and perform IEEE 1588 (Precision Time Protocol; PTP) time-synchronization via a back-end network. An additional COTS switch (Netgear® ProSAFE, model GS-105) was purchased to evaluate COTS UDP. Since TTE does not support time-synchronization to a universal time-base, ‘Linux PTP’, an implementation of IEEE 1588, was used to synchronize clocks over the back-end COTS network. A bus tap (DualComm Technology, Inc. 10/100/1000Base-T Regeneration Tap) and a quad-port COTS NIC captured full-duplex data from the simulated MAC endpoint. Figure 1 illustrates our network configuration.

Test bed software framework

The Test Bed framework is meant to simulate the MAC, APS sensors, and APS counter-measures. All traffic is to and from the MAC with no direct communication between simulated sensors or counter-measures. We simplified the software design and test scenarios to assess communication between the MAC and any individual node (or end-system), agnostic of whether the node would represent a counter-measure or a sensor.

The Test Bed framework was designed to be modular, cross-platform, and extensible. The Test Bed consists of three applications: a server application on the testmaster loads and validates the test configuration and sends tasks to the development PCs; a client application on each development PC receives and executes tasks, logs results, and transmits logs back to the testmaster; and a log processing application on the testmaster post-processes the logs and executes scripts to statistically analyze the results via the GNU R Statistical language (version 3.0.2). The Test Bed was written in POSIX-compliant (IEEE Std 1003; ISO/IEC 9945) C++11(ISO/IEC 14882:2011). It utilizes the Libxml2 library (version 2.9.1) for reading, writing, and validating XML configuration files; the TTE API library (version 66556) for interfacing with the TTE NICs; the GNU Scientific Library (version 1.16) to aggregate data; and, the

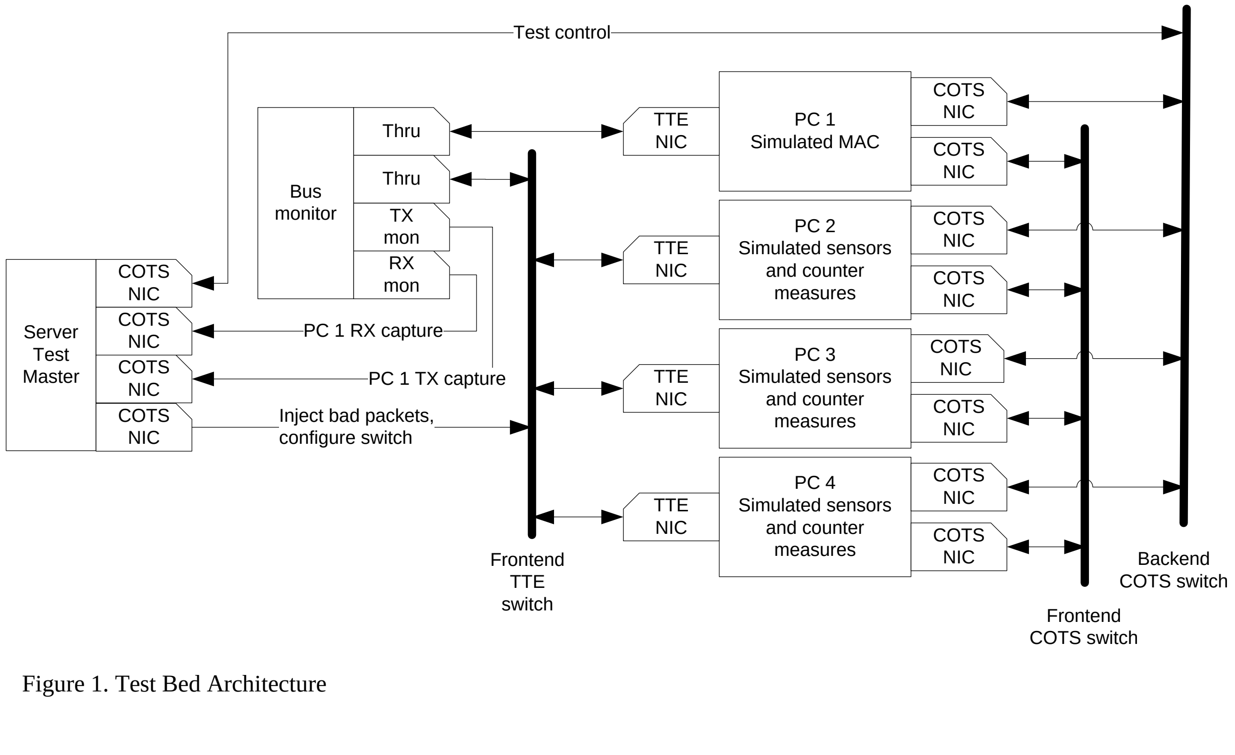

Figure 1. Test Bed Architecture

PF_RING-enabled version of libpcap (version 1.1.1) for high-speed packet capture with nanosecond-resolution hardware time-stamps and packet injection. All applications and libraries were compiled using the GNU Compiler Collection g++, version 4.8.2-19.

Some modifications to the Ubuntu default environment were necessary to improve measurement accuracy and performance. Power management was disabled and SpeedStep® was disabled in the bios to prevent clock frequency scaling. The sizes of the UDP read and write buffers were increased to 8 MB and the maximum size for message queues was set to 64 MB.

Test configuration

Tests are configured in XML (Extensible Markup Language) file format and validated at runtime using an XML Schema Definition file (XSD). Each test consists of tasks executed on each development PC. A task is configured with a direction (receive, transmit, or relay), virtual link or route ID, protocol, payload size, and payload type (sequential or randomly generated data). A message header with a message ID, message sequence number, and an additional 32-bit CRC is added to the payload of each transmission. At each transmission (send and receive), the message headers, CRC verification status, source and destination, and 64-bit 1588-synchronized CPU time-stamps are recorded on each end-system and the log files are copied to the testmaster at the end of each test. Special tasks are configured for data capture and packet injection. The log processor application on the testmaster builds a tracking history of each message as a series of links. The actual tracking history of each message is compared against the test configuration to identify lost or misdirected messages. Statistics for one-way latency, round-trip latency, and transmittal period between subsequent messages are calculated and reported.

Test scenarios and evaluation criteria

Several test scenarios were developed to evaluate each protocol for latency, throughput, determinism, jitter, and reliability. For each protocol, we evaluated the influence of payload size, bus saturation level, transmission period, and bandwidth allocation gap on one-way and round-trip communication between two end-systems. Payload sizes consisted of 64, 128, 256, 512, 1024, and 1472 B (the maximum UDP payload size supported by TTE and RC protocols). Bus saturation was achieved by injecting UDP packets on the bus at 500 and 1000 Mbps or 50% and 100% of bus saturation, respectively. All scenarios were executed with a 2-ms and then 510-ms application transmission period. This corresponds to two times the standard minimum and maximum period and bandwidth allocation gap (BAG) that can be configured for the TTE switch and end-systems. The application transmittal rate must sufficiently exceed the configured period or BAG because the TTE NIC and switch start of frame cannot be synchronized with the application's start of period. If the application transmits multiple messages within a single TTE NIC or switch cycle, data loss will occur. Each transmission is replicated approximately 1000 times to ensure adequate data points should it be necessary to filter potential context switch events. Due to the use of multiple threads and processes for servicing different classes of traffic, for data logging, and for time synchronization, it was necessary to allow context switches during idle time to allow lower priority processes limited access to the two processor cores available (4 hyperthreads).

Our measurement of transmission latency includes latencies from the API, driver, protocol stack, hardware (NIC, wire, switch), and protocol packet overhead. It was determined by subtracting each message's reception time from its transmittal time using the IDs and sequence numbers embedded in the payload. The data throughput (or effective bandwidth) is calculated as the transmitted payload size over the transmission latency and is influenced by the period or BAG, bandwidth saturation, and the payload size relative to the protocol overhead. The determinism of a protocol is assessed by multiple sources of jitter. Application transmit period jitter is calculated as the total variance of time between subsequent data transmissions which includes jitter caused by the scheduler, any context switches, and user-space major-frame time-synchronization functions (e.g. nanosleep(), clock_gettime(), etc). Receive period jitter is calculated as the total variance of time between subsequent data receptions and represents the total transmission jitter for periodic messages from the application's perspective.. Hardware jitter is calculated (for RC and TTE traffic classes) as the variance of the hardware timestamps of packets captured using libpcap via the bus tap. Network reliability and data loss is assessed in all tests by comparing logs of sent and received messages with test configuration data. Data reliability and corruption is assessed by validating 32-bit CRCs embedded in the payload of every message. Although this is of minimal value because corrupted packets are usually discarded automatically by the hardware due to failed checksums in the protocol headers. To the application, it may appear as a lost packet.

Test bed queues and processes

The test bed consists of several prioritized processes, buffers, and message queues (Figure 2) designed to avoid pre-emption and to shift as much processing as possible away from the time-critical servicing threads that ultimately transmits and receiver data and timestamps each transmission. Since the TTE API is not thread-safe, there is a single TTE servicing thread for TTE and RC traffic and a separate servicing thread for COTS UDP traffic. When a test

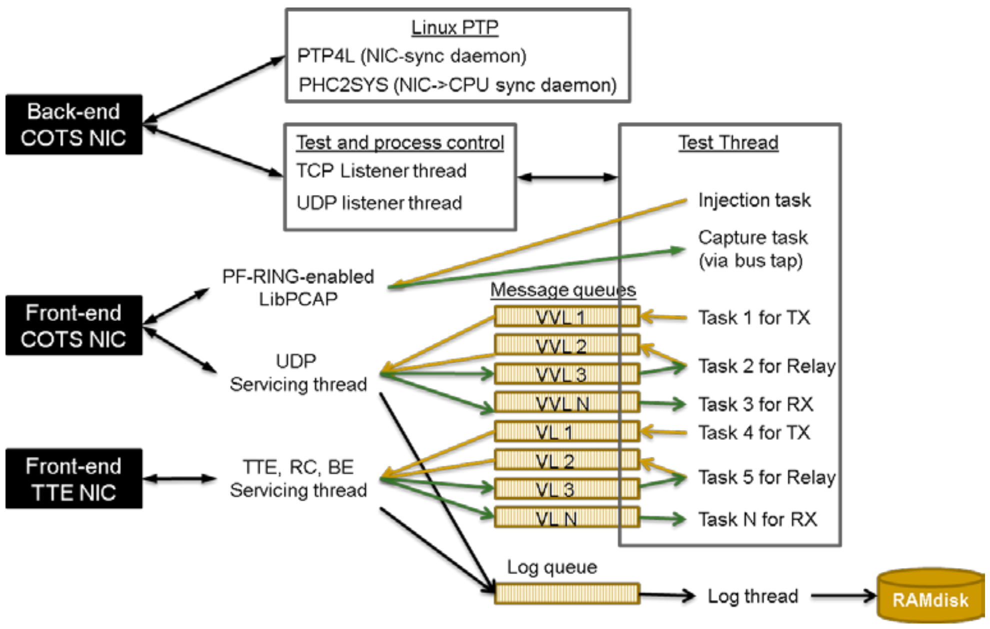

Figure 2. End-system software architecture.

is executed, a test thread is started which, in turn, executes and waits on a task thread for each configured task. A task thread can be a receiver, a sender, a relay (round-trip), a packet injector, or a packet capture. A sender task builds a message and sends it to a servicing thread to be transmitted via a dedicated message queue. The servicing threads receive incoming packets and send them to receiving tasks waiting on dedicated message queues. The servicing threads send a copy of every transmitted and received packet to a dedicated message queue for logging. The logging thread then receives the message from the log message queue, validates the payload CRC, and writes only essential information (headers, CRC status, timestamps, etc.) to a ramdisk log file which is sent to the testmaster at the end of the test.

RESULTS

Time-synchronization

Transmission latency calculations can only be as good as the timestamp resolution and time-synchronization accuracy across multiple end-systems. Although TTE provides limited support for PTP, the TTE NICs do not provide hardware timestamp support. The TTE API provides a TTE timestamp in the form of a drift-corrected time offset from NIC initialization that can be acquired via a status message returned from a receive API call or a dedicated API call. Unfortunately, these timestamps are not synchronized to a universal time-base across the TTE network. Therefore, we considered using Linux PTP to synchronize hardware time-stamps over the backend COTS network with the integrated COTS NICs (I217-LM rev. 4) which support hardware time-stamping. However, we discovered a hardware bug whereby reading the time-counter register at frequencies greater than 1 MHz causes register corruption and the driver (e1000e-3.1.0.2) performs back-to-back reads to detect rollover conditions while avoiding the locking of interrupts, thereby exasperating the problem. Therefore, we purchased and installed four additional I210 (rev 3) NICs which overcame this limitation. We also found that reading directly from the PTP hardware clock (PHC) device (e.g. “/dev/ptpX”) was vulnerable to large deviations in latency which warrants further investigation. So, we assessed the overhead and vulnerability of other userspace time sources including the processor time-stamp counter (RDTSC), the system wall-clock time (CLOCK_REALTIME), and the system monotonic time (CLOCK_MONOTONIC) and compared them with the 1588-synchronized NIC time via the PHC interface (Table 1).

Table 1. Latencies of various clock sources read 1M times from user-space on testmaster PC.

clock source	Latency				
	mean†	median	stdev	min	max
			ns		
CPU MONOTONI	602	628	307	488	188989
CPU REALTIME	600	628	247	488	117542
PTP/PHC	4817	4799	544	3632	194216
RDTSC	52	40	594	24	411848

† Latency measurements were conducted by a high-priority thread with affinity set to a single processor core.

The RDTSC clock source had by far the lowest average latency of 52 ns after 1M executions on testmaster with the processor relatively unloaded. However, it also had the greatest variance and maximum latency compared to all other clock sources. The RDT registers are processor core specific and all processes that need a common time-source must set their affinity to execute on the same core. This may be achievable on a real-time kernel; however, the non-real-time kernel used in this study does not support CPU shielding necessary to block other processes from using the same core. Scheduling processes to run on specific cores without CPU shielding caused excessive latencies due to context switches. Further, there are no mechanisms to synchronize the RDT timestamps across multiple end-systems. The system clock timers (CPU_MONOTONIC and CPU_REALTIME) had acceptable latencies and variance. Linux PTP provides an additional utility (phc2sys) which synchronizes the system wall-clock time to the PHC/PTP NIC time. Therefore, we chose to use the low-latency CLOCK_REALTIME with the added accuracy and time synchronization provided by PTP.

The start-up sequence is critically important for time synchronization. Before synchronizing the system clock to NIC time on each end-system, all NIC hardware clocks needed to be initialized and synchronized to a common wall-clock time. This ensures that the timestamps on all log files are accurate and consistent across end-systems. To achieve this, the testmaster system clock is synchronized to a time server via the NTP protocol. The NIC is initialized to system clock time using the Linux PTP phc_ctl command. The PTP server process (ptp4l) and the system clock synchronization process (phc2sys) are forked in the background. When a test is launched, the testmaster commands each end-system over the back-end COTS network to synchronize their system clocks to a time server using NTP. This brings the system clocks within range for phc2sys to take over using frequency scaling to correct for drift between the NIC and system clocks. Each end-system then forks a PTP client process (ptp4l) and the system clock synchronization process (phc2sys) to run in the background and waits for the system and NIC clocks to synchronize and stabilize. Each end-system also runs a calibration routine to determine minimum latency of reading the system clock. This latency is subtracted from all subsequent readings. The testmaster transmits a common test start time to each end-system. This value is subtracted from all timestamps to avoid 64-bit rollover conditions when processing and analyzing log data.

On each end-system, phc2sys was executed with a proportionality constant (*kp*) of 0.35 and an integral constant (*ki*) of 0.15 to reduce over-adjustment of the clock frequency that occurred occasionally using the defaults of 0.7 and 0.3, respectively. The CLOCK_REALTIME adjustment rate was set to the defaults of once per second using the fastest of five master clock readings per update. The resultant clock synchronization accuracy for a 1000-s test had a root mean square error of 8257 and 2765 and a maximum drift of 36772 and 11946 µs per second on two different end-systems, respectively.

Payloads sizes and bus saturation rates

The maximum transmission unit (MTU) supported by TTE and ARINC664, p7 is 1500 B. This consists of a 1472-B payload, an 8-B UDP header, and a 20-B IPv4 header. There is an additional 38-B overhead from the Ethernet frame which consists of a 7-B preamble, 1-B start delimiter, 14-B header, 4-B CRC, and a 12-B inter-packet gap. Therefore payload sizes evaluated in this study ranged from 64-B to 1472-B by powers of two.

Two end-systems were configured to transmit a flood of packets to two other end-systems while they communicate with one another to assess each protocol under full-duplex network saturation levels. Due to high overhead of the POSIX sendto() and lbpcap pcap_inject() functions, the bus was saturated using three concurrent threads on each end-system sending bursts of packets followed by a delay using the nanosleep() function. Bursts of 10 packets minimized the 60-ms average overhead of the nanosleep() function while keeping saturation levels as constant as possible over time. Sending bursts of packets was a reasonable approach given that the transmission of packets at such high frequencies would likely require hardware or software buffering and batch delivery at some point along the path. The packet transmittal interval to achieve different bus saturation levels are described in Table 2. For example, to achieve 100% bus saturation while end-system 1 communicates with end-system 2 requires that three threads on each end-system 3 and 4 transmit bursts of 10 packets followed by a 369-ms delay (12.3 µs x 3 threads x 10 packet bursts) to end-system 1 and 2, respectively.

Table 2. Bus saturation levels and packet injection rates

	Data rate	Packet interval‡	10-packet burst interval
Saturation level†	Mbps	µs	µs
25%	250	49.2	492
50%	500	24.6	246
75%	750	16.4	164
100%	1000	12.3	123

† Level of full duplex saturation on a 1 Gbps link
‡ Each packet consists of a 1472-B UDP payload + 8-B UDP header + 20-B IPv4 header + 38-B Ethernet frame overhead.

Application transmittal period and jitter

The application transmittal period characteristics are relatively unaffected by the actual test executed. As illustrated in Figure 3, the periods have relatively low variance without using an interrupts-based solution for data transmission. The only difference among tests was center of the distribution which corresponded with the configured application period for the test. Nonetheless, there were some outliers likely due to scheduler pre-emption. There was also a trend of greater period variance for payload sizes greater or equal to 1024 B.

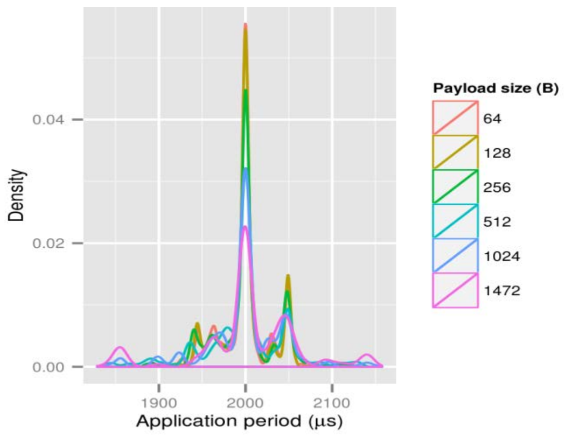

Figure 3. Representative density plot of transmittal periods for tasks configured with a 2-ms major frame

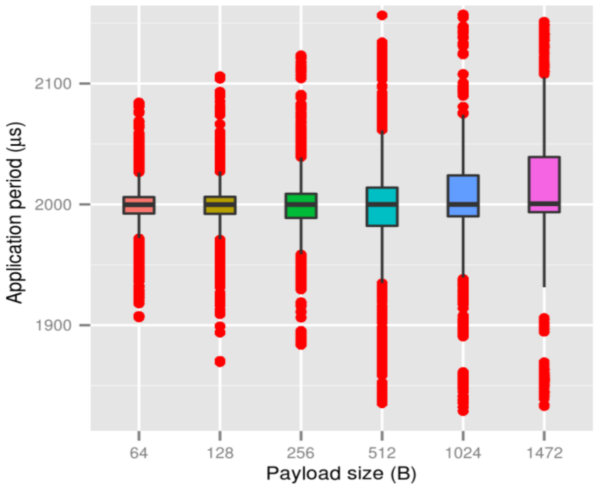

Figure 4. Representative box plot of transmittal periods for tasks configured with a 2-ms major frame.

Performance on an unsaturated network

From the application's perspective, the protocols evaluated differed greatly in latency, variance (jitter), and determinism (Table 3 and Figure 5). On an unsaturated network, with an application transmit period of 2 ms, a TTE period of 1 ms, and a RC BAG of 1ms, COTS UDP had the lowest average one-way latency and variance. It was also least affected by payload size with a range from 79.48 - 92.69 µs for payloads of 64 -- 1472 B, respectively. RC and BE latencies were lower than COTS UDP for payloads smaller than 256 B but increased strongly up to 452 µs for larger payloads. RC latencies and BE latencies were nearly identical for a given payload size on an uncongested network. The TTE switch is configured by default with a delta raster parameter of one which indicates that subsequent switch clock cycles alternate between transmitting TTE and transmitting RC or BE messages. RC and BE throughput should therefore be comparable on an uncongested network and indeed they are. Both RC and BE protocols were the most affected by payload size possibly as a result of the raster which may effectively reduce the transmit window in half. The one-way latency of TTE was higher than the other protocols. It ranged from 595.74 – 985.45 µs and increased with increasing payload sizes but with a substantial variance across all payload sizes.

Surprisingly, the protocol that was expected to be the most-deterministic (TTE) was the least deterministic and had the highest variance from the application's perspective, even though it was deterministic on the wire. This was due to the lack of synchronization between the TTE hardware cycle time and the application's transmit time. This level of synchronization would make meeting the MAPS requirement for modularity as well as supporting legacy sensors much more difficult. But without it, even a slight drift or frequency difference between the switch and application timing mechanisms can result in multiple packets sent to the TTE hardware within one period or BAG of a virtual link. In which case, only the last packet would be transmitted. Therefore, it is necessary for the application transmit cycle to be sufficiently larger than the virtual link period or BAG as we have implemented, which, unfortunately, reduces the effective bandwidth by up to 50%. However, this results in the application sending messages at various times within a virtual link's period and causes excessive variance in the time lag between application transmittal time and the TTE hardware transmittal time. This effect is evident in the TTE density histogram (Figure 5) whereby the latency is caused primarily by the difference between these two major frames.

Performance on a saturated networks

Network saturation affected all protocols to varying degrees (Table 3 and Figure 5) but there did not appear to be any substantial differences between 50 and 100% saturation. The only dropped packets were found with BE traffic with payload sizes greater than 256 B at both 50 and 100% saturation. The lack of dropped packets with COTS UDP on a 100%-saturated network was surprising. It is possible that the incoming buffers of the COTS switch are serviced and forwarded to the destinations in a round-robin fashion. This would cause packets to be dropped that originate from the saturating device but not from devices that do not exceed an equitable proportion of total bandwidth. This rudimentary scheduling provided by a COTS switch may provide

Table 3. One-way latencies of four protocols evaluated on a network with three levels of saturation with a 2-ms application transmittal period.

	Unsaturated network				50% saturated network				100% saturated network			
Payload size	**N**	**lost**	**mean**	**s.d.**	**N**	**lost**	**mean**	**s.d.**	**N**	**lost**	**mean**	**s.d.**
bytes	no.	no.	us	us	no.	no.	us	us	no.	no.	us	us
					TTE Protocol							
64	1000	0	595.74	279.86	1000	0	858.94	342.50	1000	0	771.43	313.37
128	1000	0	597.83	275.56	1000	0	782.82	314.52	1000	0	762.87	327.81
256	1000	0	613.68	276.94	1000	0	843.03	299.28	1000	0	846.94	323.41
512	1000	0	672.96	285.98	1000	0	911.58	304.40	1000	0	936.25	315.72
1024	1000	0	831.27	301.22	1000	0	1031.92	313.11	1000	0	1066.41	307.77
1472	999	0	985.45	306.92	999	0	1180.04	319.60	999	0	1155.74	298.09
					RC Protocol							
64	1000	0	40.23	2.92	1000	0	249.60	119.59	1000	0	248.47	119.61
128	999	0	57.50	2.88	999	0	257.42	116.04	999	0	258.98	115.42
256	999	0	94.13	3.15	999	0	298.34	115.97	999	0	295.20	115.90
512	999	0	168.82	2.88	999	0	379.13	119.68	999	0	370.11	115.77
1024	999	0	320.96	2.88	999	0	520.35	118.39	999	0	520.43	119.15
1472	998	0	452.06	6.01	998	0	653.49	119.61	998	0	628.17	115.72
					BE Protocol							
64	1000	0	47.66	2.67	1000	0	21205.31†	306.57	1000	0	21199.17	317.18
128	1000	0	67.27	2.62	1000	0	20781.54	283.55	1000	0	21372.63	289.55
256	1000	0	104.73	3.04	1000	0	21765.33	283.54	1000	0	21569.34	386.68
512	1000	0	174.14	3.47	1000	32	22102.62	409.31	1000	28	22020.19	346.71
1024	1000	0	322.26	5.78	1000	959	24522.79	384.90	1000	962	24968.11	66.77
1472	999	0	452.09	9.38	999	943	24856.16	360.36	999	941	24885.00	319.85
					UDP Protocol							
64	1000	0	81.90	3.31	1000	0	72.08	17.58	1000	0	55.99	11.97
128	1000	0	79.48	1.99	1000	0	180.09	86.73	1000	0	53.01	13.03
256	1000	0	84.86	1.86	1000	0	170.44	85.47	1000	0	55.65	12.58
512	1000	0	85.58	1.88	1000	0	172.26	85.14	1000	0	160.18	58.74
1024	1000	0	89.52	1.70	1000	0	181.82	74.97	1000	0	170.97	59.74
1472	999	0	92.69	2.08	1000	0	186.36	66.84	1000	0	186.62	60.03

† Further investigation is needed to determine the exact cause of the high latency and variance for BE under saturated conditions.

sufficient Quality of Service (QOS) and determinism for many applications.

The 50 and 100% saturation levels increased TTE latency to approximately 1 ms, slightly exceeding the maximum theoretical latency which is the virtual link period configured on the switch and end-points. Both saturation levels also increased RC latency by up to 210.31 µs, averaging approximately half of the configured BAG of 1 ms. The 50 and 100% saturation levels resulted in extremely high latencies for BE traffic that exceeded 20 ms. The exact cause of this is still under investigation. Saturation affected COTS UDP differently than the other protocols. Fifty percent saturation increased latency for payloads greater than 64 B by over 100% whereas 100% saturation increased latency for payloads greater than 256 B by 87-101%. These differences may have been caused by the timing offset between the start of applications' major frames and the bursts of saturating traffic. Nonetheless, COTS UDP had, by far, the lowest latency of 55.99 -186.62 µs on a fully saturated network.

CONCLUSION

In this study, TTE was shown to have higher latency and variance from the application's perspective as compared to other protocols, even if it is deterministic on the wire. This study demonstrates that a deterministic network and deterministic applications can be nondeterministic if they are not synchronized with each other. The worse-case latencies are still bounded, which is an advantage, but the worse-case latency is amplified by the number of unsynchronized components in the communication chain. Without synchronization, TTE messages must be configured to have a much smaller period than the communicating applications, resulting in a substantial loss in effective bandwidth. Further, the transmission time is reserved for TTE even if there is no message to transmit. Therefore, TTE is most appropriate for critical periodic and low-bandwidth messages. Suitable uses may include status reporting, health-monitoring, and position or state-change announcements. If time-synchronization (e.g. 1588) of the network and the application was included as an integral part of the TTE protocol, we might envision more uses for TTE in certain markets.

RC also performed as expected with the exception of the increased latency for larger payload sizes. It is comparable if not faster than UDP for very small payloads. Bandwidth partitioning schemes such as this are a good compromise between throughput and determinism. We look forward to seeing how the TSN protocol performs using AVB-based bandwidth partitioning with time scheduling and synchronization. The fast, robust, and low-overhead COTS UDP protocol remains the standard to which all other protocols are compared. Although it includes no QOS services, it can be very well-behaved on a saturated full-duplex switched network depending on the fairness method that COTS switches use to service incoming buffers.

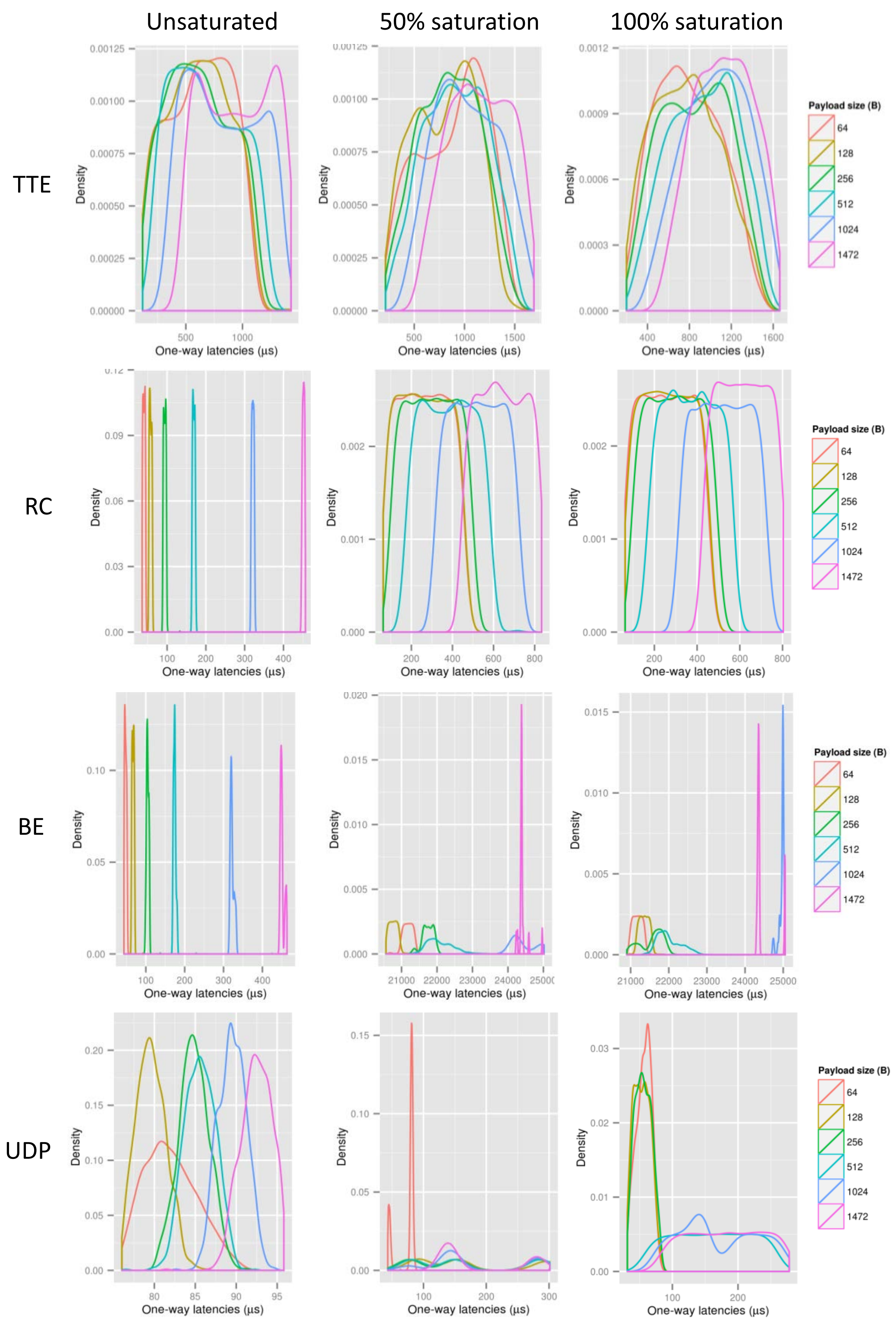

Figure 5. Density plots of one-way latencies by protocol and bus saturation level. Further investigation is needed to determine the cause of high latency of BE on a saturated network.

ACKNOWLEDGEMENTS

This research was conducted by DornerWorks Ltd. funded and contracted by the U.S. Army Tank-Automotive Research, Development and Engineering Center (TARDEC) via Alion Science and Technology Corporation.

REFERENCES

[1] Aeronautical Radio, Inc., "Aircraft Data Network Part 7. Avionics Full Duplex Switched Ethernet (AFDX) Network.," *ARINC Specification 664p7*, 27 June 2005.

[2] SAE Aerospace, "SAE AS 6802. Time-Triggered Ethernet," November 2011.

[3] IEEE Computer Society, "IEEE Standard for Local and Metropolitan Area Networks---Audio Video Bridging (AVB) Systems," 19 May 2008.

[4] Time-Sensitive Networking Task Group, "Time-sensitive networking," *IEEE 802.1*, Not yet ratified.

[5] Time-Sensitive Networking Task Group, "Timing and Synchronization for Time-Sensitive Applications," *802.1AS-Rev*.

[6] IEEE Instrumentation and Measurement Society, "A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE 1588-2008*, 2008.

[7] Time-Sensitive Networking Task Group, "Enhancements for Scheduled Traffic," *IEEE 802.1Qbv*, 15 May 2012.

[8] J. Postel, "User Datagram Protocol," USC/Information Sciences Institute, 1980.