Transport Protocols Revisited

Maggie Breedy[1], Peter Budulas[2], Alessandro Morelli[3], Niranjan Suri[1,2]
[1]Florida Institute for Human and Machine Cognition, Pensacola, FL USA
[2]US Army Research Laboratory, Adelphi, MD USA
[3]University of Ferrara, Italy

Abstract—**The purpose of the paper is to evaluate and compare the performance of various transport protocols over some current tactical radios using different topologies The four selected transport protocols were TCP, SCTP, UDT, and Mockets. The comparison was done with three different tactically relevant radios – the Harris PRC-117G, the TrellisWare TW-400, and the Persistent Systems WaveRelay MPU4. The results show a surprising amount of variability in performance, and indicate that the Mockets transport protocol provided the best overall performance on two of the three radios used for evaluation.**

I. Introduction

Transport protocols sit between the network layer and the middleware / application layers and are typically responsible for end-to-end delivery of data with various types and qualities of service. The most popular transport protocols are TCP and UDP, with TCP being used by the majority of applications on the Internet. TCP provides a simple stream-based protocol with reliable and in-order delivery of data. However, TCP was designed for the commercial Internet and fairly reliable wired and wireless networks. As a result, TCP typically provides poor performance when operating over tactical networks that are typically wireless, bandwidth constrained, unreliable, variable latency, and prone to temporary disconnection.

The primary purpose of this paper is to evaluate and compare the performance of various transport protocols over some current tactical radios in a suburban outdoor environment. This is a significant, ongoing task and in order to constrain the complexity of the problem, several simplifying assumptions have been made in terms of the choice of protocols, the selection of radio platforms, the size of the network, and the deployment topologies. Even with these assumptions, the results are sufficiently interesting and worth reporting – hence this paper. The evaluation is by no means complete, as new aspects are still being incorporated. However, those will be reported in future publications and reports.

It is important to note that this paper is not comparing the performance of different radios, and we discourage the reader from using the results presented as a means to compare radio performance for two reasons. First, the radios were not configured (in terms of frequencies and bandwidth) to provide the same performance. Second, we do not feel that the testing was sufficiently rigorous for that purpose. Instead, our objective is to compare the performance of different transport protocols on each of the selected radio platforms.

For the purpose of this paper, four transport protocols were selected – TCP (the de facto baseline), SCTP (Stream Control Transmission Protocol), UDT (UDP-based Data Transfer Protocol), and Mockets. An overview of these four protocols is provided in section 2. Several other protocols were considered for evaluation, including SCPS-TP (Space Communications Protocol Specifications – Transport Protocol) [ref], DCCP (Datagram Congestion Control Protocol) [ref], and QUIC (Quick UDP Internet Connections) [ref], but were ultimately not included for a variety of reasons – primary to constrain the scope of the paper, but for other reasons as well. For example, the home page for SCPS (www.scps.org) is not even online anymore. An internet archive was used to retrieve a version of the code for an implementation from 2009, and could possibly be incorporated into a future evaluation. Also QUIC was not included because it was difficult to extract an easily reusable implementation out of the overall Google™ Chromium codebase. Again, this could possibly be incorporated into a future evaluation.

Likewise, for the purpose of this paper, three tactical radios were selected – the Harris PRC-117G with the ANW2 waveform, the TrellisWare TW-400 CUB, and the Persistent Systems WaveRelay MPU4. These radios were selected primarily because they are commonly utilized for tactical edge networks. Other radio platforms that were not selected for this initial evaluation but would probably be considered in the future include the Harris PRC-117G with SRW (Soldier Radio Waveform), the Harris PRC-152A platform, and the General Dynamic PRC-154A Rifleman radio with SRW. Details regarding the configuration of the radios are described in section 3.

In order to establish a baseline performance measure prior to more complex scenarios and topologies, only three radios were deployed for all of the experiments conducted. One radio was connected to a server node, one radio to a client node, and one intermediate radio acted as a communications relay. Furthermore, all of the tests were done in a static scenario – without any mobility during the test. Mobility will be incorporated into future evaluations. The radios were deployed outdoors in a suburban office park like environment. Frequencies were selected so as to not interfere with each other but also to not allow common cellular and WiFi signals that might be present to interfere with the radios being evaluated. Being outdoors, there was random movement of automobiles – sometimes between the radio node locations. Likewise, there were buildings and trees and minor changes in

This work was conducted at and supported by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-11-2-0095.

elevation. While one of the configurations involved a number of trees in the path between two of the nodes, the tests were conducted at the end of March – when there was no foliage on the trees. The environment and the topologies are further described in section 4.

The actual results are presented in section 5, followed by a discussion of the results. Section 6 contains some conclusions and a description of future work.

II. Overview of Selected Transport Protocols

A. TCP

The Transmission Control Protocol (TCP) is a transport layer protocol of the Internet Protocol Suite, and the de-facto standard for the majority of Internet applications and services, such as HTTP(S), e-mail, SSH, etc. TCP establishes an end-to-end connection between two applications running on nodes of an IP-based network and provides the abstraction of a reliable, ordered stream of bytes flowing between them. Reliability is guaranteed thanks to a mechanism based on acknowledgments (ACKs) of correctly delivered packets and on the timeout-driven retransmission of unacknowledged segments. To ensure the ordered delivery of data, TCP uses a sequence number to identify the first byte of each segment; this allows the receiver to also restore the correct transmission sequence in case of packet loss. TCP also provides flow control and congestion control, to share the bandwidth between multiple connections equally and avoid network collapse. TCP was designed for wired infrastructure environments, hence it exhibits several weaknesses in wireless, low bandwidth, and intermittently connected networks. When applications do not require in-order and reliable data transmission, their only choice consists of switching to other transport protocols, like UDP.

Researchers have invested much effort to improve TCP over the course of its existence. This has led to the development of a number of alternative congestion-avoidance algorithms, which add features like slow-start, fast recovery, fast retransmit, and different ways to manage the size of the congestion window in response to packet losses or successful ACKs. TCP CUBIC [1] is the default algorithm in Linux kernels from version 2.6.19 to 3.1 and the one we used in our experiments. It builds upon TCP BIC and changes the congestion window growth function from a combination of linear, logarithmic, and exponential curves to a cubic function, hence its name. TCP CUBIC shows more efficient use of network resources in high bandwidth-delay product networks under a wide range of round-trip times and achieves better fairness with competing TCP flows [1].

B. SCTP

The Stream Control Transmission Protocol (SCTP) [2] is a transport layer protocol that relies on IP to provide message-oriented and connection-oriented end-to-end communications. Similarl to TCP, SCTP ensures the reliable, in-sequence delivery of messages and comparable algorithms provide flow and congestion control. However, SCTP optionally provides order-of-arrival message delivery semantics. Unlike TCP, SCTP also supports multi-homing and multi-streaming. The former allows the protocol to take advantage of multiple IP addresses or network interfaces on the same endpoint, a feature that can improve connection survivability in case of node mobility or link disruption. SCTP currently exploits multi-homing for redundancy purposes only, and it does not permit increasing the maximum throughput. A "primary" address is chosen to receive data, and heartbeats are used to monitor the availability of alternate transmission paths and to test previously discovered paths. Multi-streaming, instead, ensures the concurrent transmission of multiple streams of data between connected hosts. This feature is important to avoid head-of-line blocking between independent streams.

Although the Linux Kernel natively supports SCTP since version 2.6, we installed the Linux Kernel Stream Control Transmission Protocol Tools (LKSCTP Tools), version 1.0.16, on the systems we used in our experiments. LKSCTP provides a Linux user-space library that we used in the test utility we developed to access the SCTP-specific API that is not part of the standard sockets interface.

C. UDT

UDP-based Data Transfer (UDT) [3] [4] is an application-level data transport protocol that builds on top of UDP to support distributed data intensive applications over wide area high-speed networks (WAN). The main design goal of UDT is to reach high data transfer throughput over the network, while also achieving fairness between multiple UDT flows and without starving TCP connections. UDT is connection-oriented, and provides both stream-oriented and message-oriented delivery semantics. Additionally, it is possible to configure UDT to perform reliable, partially reliable, or unreliable message transmissions, as well as sequenced or unsequenced delivery. The UDT API is very similar to the Berkeley socket API, a choice made to simplify the process of switching from TCP or UDP to UDT. UDT supports user-defined congestion control algorithms and the multiplexing of multiple UDT connections over a single UDP flow (that is, all messages specify the same destination port number). Additionally, applications using UDT can access an internal performance monitor to retrieve statistics about open connections. Finally, as an application-level library based on UDP, it is easy to port UDT-based applications to different machines and operating systems.

To develop the test utility we used during our experiments, we installed the libudt-dev version 4.11. libudt-dev is a package for Ubuntu Linux systems that provides an implementation of version 4 of UDT that can be used for application development.

D. Mockets

The Mockets (for Mobile Sockets) [5][6] framework is an application-level communications library that is part of the IHMC Agile Computing Middleware, and is specifically designed to support adaptive applications in MANET environments. Figure 1 shows the framework's architecture and how it interacts with applications and the network.

Mockets provides a number of unique features, including complete orthogonality between reliability and sequencing in the delivery semantics, as well as prioritization, message tagging and replacement of enqueued but outdated messages, detailed communication statistics, numerous timeout options, policy-based bandwidth control, statistics collection, and endpoint migration. With Mockets, applications can open multiple independent data flows and assign different QoS and priority levels to each of them. Also, applications can access statistics on the current channel conditions and the state of connections. Finally, endpoint migration is extremely useful to avoid breaking connectivity in case of vertical or horizontal handover, frequently cause by node mobility in mobile ad-hoc and heterogeneous networks.

Like UDT, the Mockets framework is designed as an application-level library and so it is not part of the operating system kernel or network protocol stack. This allows easy porting of Mockets to many environments, and it is currently available for Win32, Linux, Android, and MacOSX platforms. This design choice was essential to support easy deployment, platform independence, and phased utilization. It is possible to only have a subset of the applications to use Mockets, while the remaining can continue to use TCP and UDP. This facilitates adoption and deployment, as applications can gradually migrate to Mockets from the sockets API.

Mockets provides a rich model of interaction between applications and the framework, which fosters a feedback-loop-based programming model. In accordance with it, applications can tune the amount of data handed over to Mockets and the delivery semantics requested for that data, making tradeoffs on delivery latency and bandwidth utilization. In turn, Mockets keeps applications up-to-date about the current state of the network and the connections, so that applications can adapt to it and make more informed choices in the future.

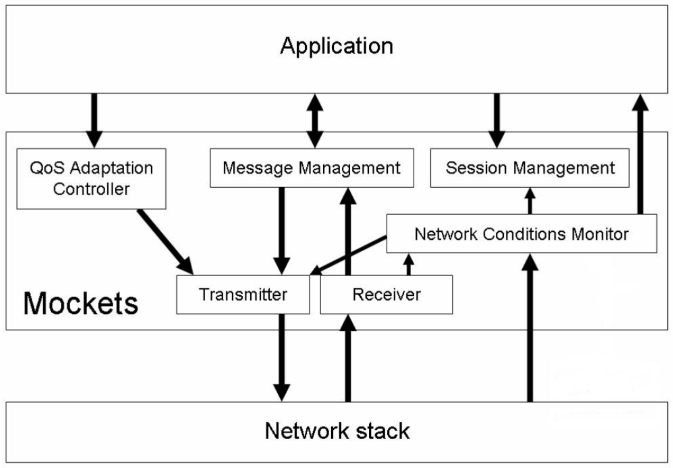

Figure 1: Mockets Architecture

III. Tactical Radio Platforms and Configuration

A. *Harris PRC-117G with ANW2*

The Harris AN/PRC-117G radio is a fielded software defined tactical radio. This radio is currently being used in tactical environments in various locations around the world. Through the software definable waveforms, this radio allows for backward compatibility with existing fielded radios as well as newer wave forms that are further optimized for the tactical environment. The 117G has a wide frequency of operation starting from 30MHz and extending into the 2GHz bands. It can operate in narrow and wide band modes, and is capable of type one encryption. Transmit power for this radio in the configuration used can vary from .2 to 5 Watts. The specific waveform used for this test is Adaptive Networking Wideband Waveform (ANW2C). When configured properly, multiple radios will form a Mobile Ad hoc Network (MANET). The three radios used were configured using CPA, a tool provided by Harris for radio network configuration. For the tests described in this paper the network was constrained to three radios allowing for a maximum of 2Mbps throughput between any two radios, occupying 5MHz of channel bandwidth. The details of the specific radio configuration used for this test are provided in Table 1.

B. *TrellisWare TW-400 CUB*

The TrellisWare TW-400 radio is a small footprint radio used in many commercial, law enforcement and tactical applications. It is based on a software defined radio architecture and uses TrellisWare's Tactical Scalable MANET Enhanced (TSM-E) waveform, with AES-256 encryption. It can operate in two specific bands 1800/2200 MHz and offers different interface adapters for analog video, audio, USB, Ethernet and Wi-Fi. As used for this test, each radio was configured with an Ethernet interface adapter. The TW-400 is capable of up to 8Mbps data rate, occupying 20MHz of channel bandwidth. These radios operate as peer to peer devices and will autonomously form a MANET. Configuration for the radios used in this test was performed through a custom configuration application. Specific details of the configuration for this test are listed in Table 1.

C. *Persistent Systems WaveRelay MPU4*

Persistent systems WaveRelay Man Portable Unit fourth generation (MPU4) is a peer to peer MANET radio being used in many commercial, law enforcement and tactical applications. This radio provides data rates up to 37Mbps using UDP and 27 Mbps using TCP, while occupying 40MHz of channel bandwidth. The radio provides AES-CTR-256 encryption. These radios are manufactured to operate in several frequency bands. The radios used for this test were operating in the 2.3 to 2.5 GHz bands. Interface dongles are available to allow audio, USB and Ethernet. The Ethernet dongles were used for the tests covered by this paper. Configuration for the radios used in this test was performed through a browser interface. Specific details of the configuration for this test are listed in Table 1.

Table 1: Radio Configuration Parameters

Radio Name	Manufacturer	Model Number	Firmware Version	Waveform	Frequency in MHz	Channel Bandwidth	Transmit Power
Harris 117G	Harris	RT-1949(P)(C)	4.4.0	ANW2C	402.50	5MHz	2W
TrellisWare CUB TW400	TrellisWare	ASY0540250	5.5.6	TSM-E	2230.00	20MHz	2W
WaveRelay MPU4	Persistent Systems	WR-MPU4-12	18.5.1	64QAM?	2507.00	40MHz	2W

IV. Environment, Topologies, and Experimental Scenario

A. *Environment*

The Army Research Laboratory (ARL) Adelphi Laboratory Center (ALC) campus was chosen for this experiment. The ARL-ALC campus consists of multiple buildings of varying construction, interconnected by paved roadways and pathways. This campus is considered to be representative of a suburban environment, and also provided a somewhat controlled area for this experiment. Though the elements involved in this test were stationary, there were multiple vehicles and pedestrians moving throughout the campus and the testing elements at any given time. In most of the testing topologies, trees and some foliage existed between the relay node and one of the end nodes. The tests conducted for this paper took place toward the end of March with most of the foliage not yet in bloom. . Terrain variation also existed between test sites, and elevation maps are provided in section 4.2. Since the intent of this test was not to test radio effectiveness or RF propagation and path loss, further measurements were not performed. Scans of the spectrum before the tests did not reveal any interferers on the frequencies being used.

The antennas used during the tests were the standard antennas provided with the man portable units. In the case of the TrellisWare radios, the standard dual band antenna was used. For the TrellisWare and the WaveRelay radios, the antennas were attached to the radio and the radio was mounted to the roof of the trailer and vehicles used in the test. For the Harris 117G, the antenna was mounted to the roof of the trailer and vehicles, but the radio was remoted to the interior, using a low loss coax cable. More details on radio configuration are given in section 3.

B. *Topologies*

Preliminary tests were conducted using the Harris and WaveRelay radios. Given the environment mentioned in 4.1, these tests determined that a relay node would be required in order to get the desired throughput at the desired distances. This intermediate node also tested the protocol's response to a relay, which would be a common occurrence in a wireless ad-hoc network. End points for the topology were selected to test performance at various distances. Figure 2 shows a map (courtesy of Google™ Earth) of the overall topology for the test. Node locations are labeled and marked with a circle. Distance is measured between each marked location in feet.

For all tests, the server node remained stationary at the trailer location. The relay node also remained stationary in the South Lot. The client node remained stationary during the tests, but then moved to each of the other locations to conduct each test. The VIP lot topology involved a server node located in the trailer, a relay node in the South Lot, and a client node in the VIP Lot. For the Flag Pole topology, the server and client remained in their previous locations while the client moved to the Flag Pole location for the test. During the Zahl Road topology, server and relay remained as before, while the client was located on Zahl road. In the K lot topology, server and relay remained in their respective locations, while the client node moved to the K Lot location for the test. The results for these tests are presented and discussed in section 5.

Figure 3 is an elevation graph of the terrain for the K Lot topology. This is provided to show the variability of the terrain.

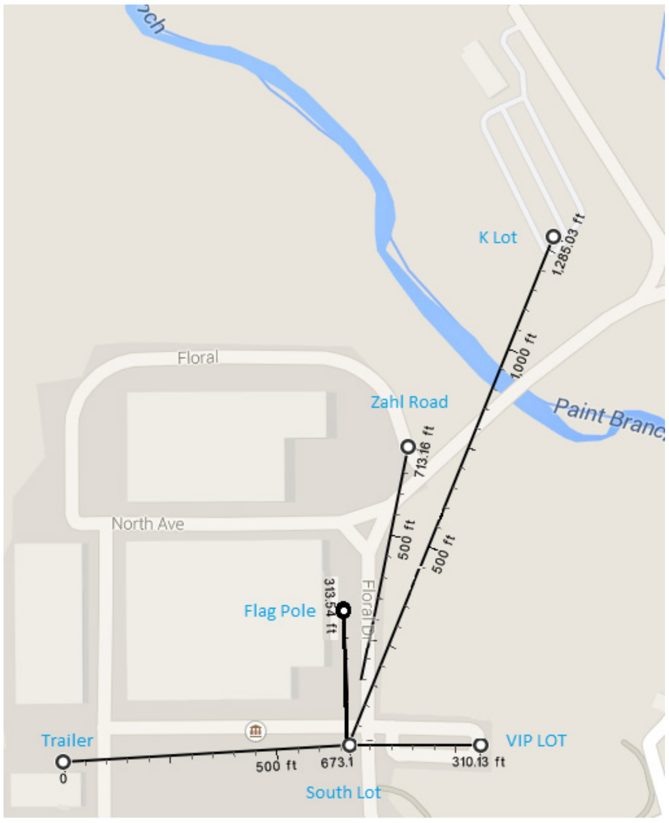

Figure 2: Topology for Protocol Performance Tests

C. *Hardware, Software, and Scenario*

As described in the topologies section, each of the tests consisted of three radio nodes and two computer nodes. The server node was always fixed at the location marked Trailer and consisted of a Dell Precision 6600 Laptop with an Intel Core i7 2820QM Processor with 8 GB RAM and a 256 GB SSD running Ubuntu Linux 14.04 Desktop 64-bit. Likewise, the client node was an identically configured laptop located at one of the four possible locations marked VIP Lot, Flag Pole, Zahl Road, and K Lot. The relay radio node, located at the position marked South Lot, was not attached to a computer node.

The experiment scenario was chosen to be very simple – a bulk data transfer over a single connection from the client node to the server node. This also ensured that there was no other traffic being generated at the same time to interfere with the single data transfer. As already mentioned, there was no mobility involved during the invocation of the test.

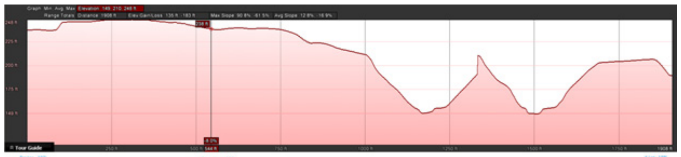
Figure 3: Terrain Elevation for the K Lot Test Topology

V. Experiment Results and Analysis

Two custom applications were implemented to exercise the selected transport protocols – one for the server side and one for the client side. The server side instantiated listeners (e.g., server sockets) for each of the protocols and simply waited for incoming connections. The client node connected to the server node using one of the four selected transport protocols, waited for a response, sent the size of the data to be uploaded to the server followed by the data. The client would then wait until the server acknowledged receipt of all the data (done by sending a single byte, the character "."). The client determined the throughput by measuring the elapsed time starting after the size of the data was sent (but before the transmission of the actual data started) and until the acknowledgement was received. The size of the data was 1024 KB for the K Lot and the Zahl Road topologies and 2048 KB for the Flag Pole and VIP Lot topologies.

Each test was conducted with one of the three radios connected to the client and server nodes. The other radios were left on and idle (which should not have caused any interference given that the frequencies were deconflicted as shown in Table 1). In order to reduce temporal effects, the client cycled through each of the four transport protocols (Mockets first, followed by TCP, then SCTP, and finally UDT). This comprised one iteration of the test and each test consisted of 10 such iterations.

The results of the experiment are shown in Table 2. Each column represents one particular topology and radio combination. Each row presents the results for a specific transport protocol for the topology and radio combination. For each result, we show the average throughput, the maximum throughput, and the minimum throughput, all in KB/sec, over 10 iterations, followed by the standard deviation. One exception is that for the K Lot topology, we only show the results for the Harris 117G, as the other two radios did not work sufficiently reliably to collect data for multiple iterations. For example, when the nodes were connected to the TrellisWare radios, they were able to ping each other, but the client would time out connecting to the server, the data transfers would abort, or the data transfers would essentially remain stuck and had to be aborted after several minutes.

One of the most interesting results of these tests was the observed variability in performance across multiple iterations. We show the maximum and minimum performance and the standard deviation to highlight this observation. Figure 3, Figure 4, Figure 5, and Figure 6 show the results in graphical form, for each of the four topologies. The thin vertical bar shows the minimum and maximum observed performance. The larger rectangle shows one standard deviation above and below the mean performance. Note that Figure 6 only shows the performance for the Harris radio. As discussed earlier, the other two radios were not able to complete the test with this topology.

Table 2: Results showing Average, Maximum, and Minimum Throughput and Standard Deviation

		VIP Lot Topology			Flag Pole Lot Topology			Zahl Road Topology			K Lot Topology		
		Harris	TrellisWare	WaveRelay	Harris	TrellisWare	WaveRelay	Harris	TrellisWare	WaveRelay	Harris	TrellisWare	WaveRelay
Mockets	Average	260.42	347.02	238.06	257.98	311.85	366.41	182.57	196.77	97.15	57.54		
	Max	262.09	371.82	403.87	261.99	324.98	581.32	190.23	220.74	117.16	64.89		
	Min	257.51	309.04	55.61	233.74	263.27	165.19	157.81	158.74	43.97	26.34		
	St. Dev.	2.02	24.04	101.46	8.68	19.94	159.40	9.93	25.37	24.83	11.70		
TCP	Average	194.15	175.56	219.49	191.03	147.99	385.46	97.29	85.82	103.86	57.43		
	Max	199.09	227.53	293.96	201.63	226.45	489.60	126.61	100.83	154.22	100.90		
	Min	180.00	99.91	158.77	177.87	55.28	267.22	43.01	70.58	64.27	25.52		
	St. Dev.	7.56	44.38	39.61	10.86	53.30	68.66	28.95	10.75	27.70	21.11		
SCTP	Average	157.01	116.98	255.01	156.53	131.80	365.72	96.38	70.86	97.77	38.90		
	Max	161.67	165.29	374.27	161.60	164.39	568.26	118.79	103.81	136.12	67.78		
	Min	117.65	63.27	129.40	121.52	76.56	165.52	36.12	38.55	36.08	17.01		
	St. Dev.	13.84	40.50	77.37	12.43	33.82	125.85	28.76	21.07	30.23	16.87		
UDT	Average	154.43	264.69	139.37	127.58	303.91	260.06	128.02	132.70	75.52	58.48		
	Max	196.15	369.94	265.04	180.28	450.61	436.49	198.03	194.16	104.38	94.74		
	Min	92.88	147.24	50.51	34.93	86.25	116.33	42.16	49.71	10.73	16.32		
	St. Dev.	28.11	70.00	67.39	54.45	120.29	98.90	49.70	49.26	27.62	24.91		

The variability observed was particularly surprising given four simplifications imposed on the experiment. The first simplification was that there was no mobility involved whatsoever. The second simplification was that all the frequencies were deconflicted prior to the test, and none of the radios were being interfered with at the RF level. The third simplification was that there were only three radio nodes in the topology. The fourth simplification was that there were only two applications using the radio, with only one active data transmission.

The observed performance was also surprising given the distances and the number of nodes in the topologies. As shown in Figure 2, the distance between the server node and the relay node was always fixed at 673.1 feet. The distance between the relay node and the client node varied from 310.13 feet and 1285.03 feet.

The next observation is to compare the performance of the different transport protocols. As can be observed in Table 2 and Figure 4, Figure 5, Figure 6, Figure 7 and Figure 8, the Mockets protocol performed better than all the other protocols on the Harris. In particular, Mockets performed 45% better than TCP and 71% better than SCTP and UDT. This result was surprising for the Harris radio given that Harris includes a TCP accelerator built into the radio when using the ANW2 waveform. TCP did perform as the second best protocol on the Harris radio, slightly beating out SCTP and UDT.

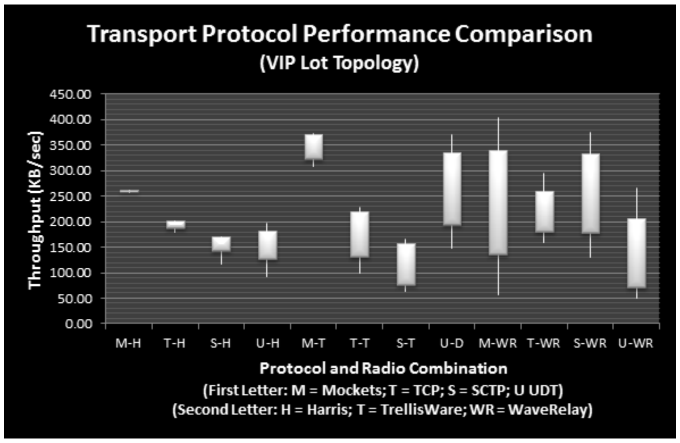

Figure 6: Performance Results for VIP Lot Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/sec)

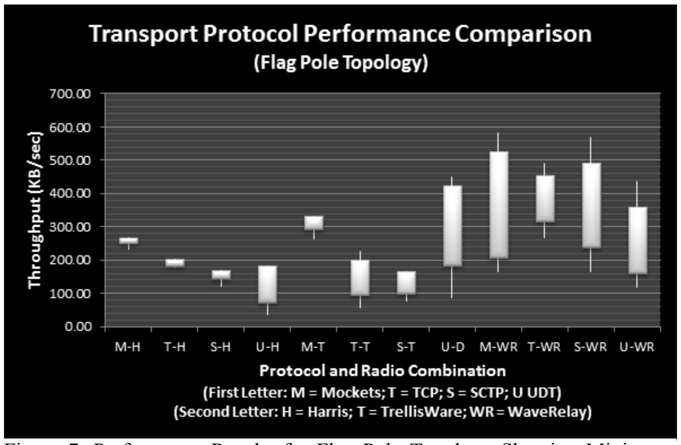

Figure 7: Performance Results for Flag Pole Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/sec)

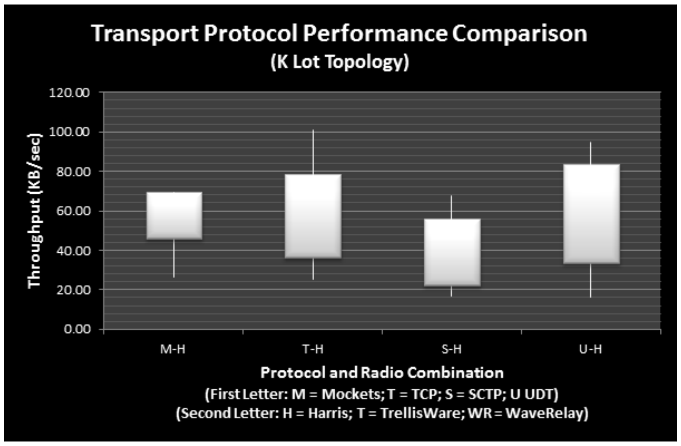

Figure 4: Performance Results for K Lot Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/sec)

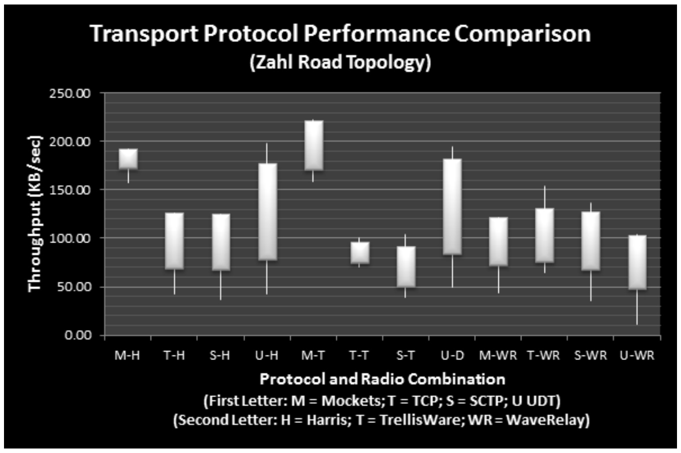

Figure 5: Performance Results for Zahl Road Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/sec)

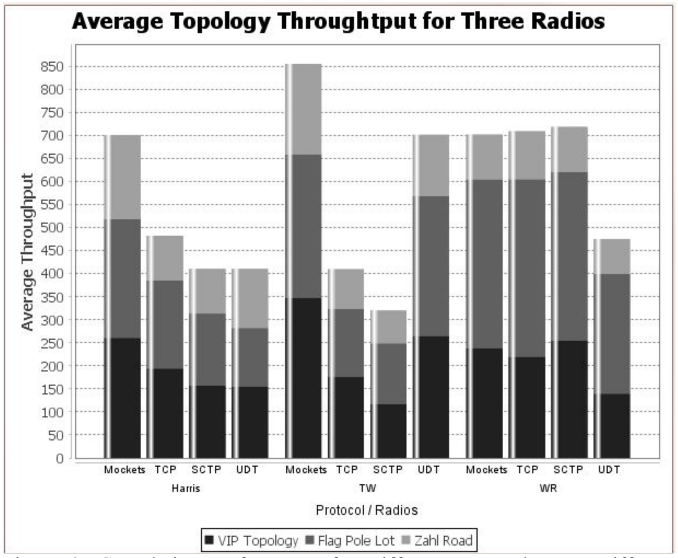

Figure 8: Cumulative Performance for Different Protocols over Different Radios

On the TrellisWare radio, Mockets significantly outperformed TCP by 109% and SCTP by 167%. It also outperformed UDT, but by a smaller margin of 22%. The second best performing protocol was UDT, followed by TCP and finally SCTP.

On the WaveRelay radio, Mockets, TCP, and SCTP performed just about the same, within 1% to 2% of each other. UDT was the worst performing protocol on the WaveRelay radio, with Mockets beating UDT by 48%, TCP beating UDT by 49%, and SCTP beating UDT by 51%.

Another surprising observation was that UDT did very well on the TrellisWare radio compared to TCP and SCTP, but did much worse on the WaveRelay radio than TCP and SCTP.

As mentioned earlier, we urge the reader to not use these results to compare the performance of one radio with another, as the characteristics of the radio as well as the configuration parameters do not provide a common baseline for such a comparison.

VI. Conclusions and Future Work

This paper compared the performance of four different transport protocols – TCP, SCTP, UDT, and Mockets over three different radios – the Harris 117G, the TrellisWare TW-400, and the WaveRelay MPU4, using four different topologies. The tests were conducted in a suburban outdoor uncontested RF environment using very simple, static, three node topologies, with one server node, one relay node, and one client node. The results showed significant variability in the results, which was surprising given the conditions of the test. Mockets outperformed all the other protocols over the Harris and TrellisWare radios, with TCP coming in second on the Harris and UDT coming in second on the TrellisWare radio. Mockets, TCP, and SCTP performed very similarly on the WaveRelay, with UDT not doing as well.

The observed variability lends support to the argument for adaptive middleware to help applications address such fluctuations in the network performance. In this particular evaluation, all the protocols were compared using the TCP model – namely a reliable and sequenced stream of bytes. The TCP model is limiting because it does not allow the transport protocol to distinguish between message boundaries, and the only type of service provided is reliable and sequenced delivery of bytes. One consequence is head-of-line blocking, which prevents subsequent messages from being delivered even if they were received in their entirety. Another consequence is that the application cannot specify different requirements for different messages. The other three protocols evaluated do provide support for message-based abstractions, which address some of these concerns. Mockets, which was designed specifically for tactical networking environments, provides extensive support to enable application adaptation [5]. However, many legacy applications still use TCP and rewriting them to use an alternate protocol would be difficult, expensive, and/or impossible. For those cases, a proxy-based approach is a good alternative. The Harris PRC-117G radio provides a TCP accelerator (which is a proxy) with ANW2 (although Mockets provided better performance than TCP, even with the accelerator). An independent proxy implemented as middleware provides additional flexibility and can work with a variety of radios. One such proxy is described in [7].

The tests conducted to date will continue in the future with additional nodes, more complex topologies, and node mobility. We also intend to incorporate additional tactical radios/waveforms (for example, SRW) and additional transport protocols.

VII. References

[1] S. Ha, I. Rhee, L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant", ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel*, Vol.* 42, No. 5, pp. 64-74, July 2008.

[2] R. Stewart, Stream Control Transmission Protocol, RFC 4960, September 2007.

[3] Y. Gu and R.L. Grossman, UDT: UDP-based Data Transfer for High-Speed Wide Area Networks, Computer Networks (Elsevier), Vol. 51, No. 7, May 2007.

[4] Y. Gu, X. Hong, and R.L. Grossman, Experiences in Design and Implementation of a High Performance Transport Protocol, in Proceedings of the 2004 Super Computing Conference (SC 2004), Nov 6-12.

[5] N. Suri, E. Benvegnù, M. Tortonesi, C. Stefanelli, J. Kovach, and J. Hanna, Communications Middleware for Tactical Environments: Observations, Experiences, and Lessons Learned, in IEEE Communications Magazine, Vol. 47, No. 10 (October 2009), pp. 56-63.

[6] E. Benvegnù, N. Suri, J. Hanna, V. Combs, R. Winkler, and J. Kovach, Improving Timeliness and Reliability of Data Delivery in Tactical Wireless Environments with Mockets Communications Library, in Proceedings of the 2009 IEEE Military Communications Conference (MilCom 2009), October 2009, Boston, MA.

[7] M. Tortonesi, R. Kohler, A. Morelli, C. Stefanelli, N. Suri, and S. Watson, Enableing the Deployment of COTS Applications in Tactical Edge Networks, in IEEE Communications Magazine, Vol. 51, No. 10 (October 2013), pp. 66-73.