Towards the Adaptation of an Active Measurement Protocol for Delay/Disruption-Tolerant Networking

Carlos H. Lamb[1], Bruno L. Dalmazo[2(✉)], and Jeferson C. Nobre[2]

[1] University of Vale do Rio dos Sinos, São Leopoldo, Brazil
carloslamb@gmail.com
[2] Federal University of Rio Grande do Sul, Porto Alegre, Brazil
{bldalmazo,jcnobre}@inf.ufrgs.br

Abstract. Delay/Disruption-Tolerant Networking (DTN) can be characterized by high delays and disconnections between the participating nodes. Despite these characteristics, DTN needs network management features similar to those found in conventional networks such as, for example, latency monitoring in message exchanges. One of the approaches commonly used for such monitoring is the use of standardized protocols for active measurements, such as, for example, the Two-Way Active Measurement Protocol (TWAMP). However, protocols like TWAMP are not prepared for DTN because they consider conventional network environments, such as the Internet. This paper describes the Delay/Disruption-Tolerant Two-Way Active Measurement Protocol (DTWAMP), a proposed extension for TWAMP which enables the execution of active measurements in DTN. In addition, experiments were performed to evaluate the proposed solution. The results demonstrate that DTWAMP has desirable properties to perform monitoring tasks in DTN.

Keywords: Delay/Disruption-Tolerant Networking · Active measurements · Two-Way Active Measurement Protocol

1 Introduction

Computer networks employ different architectures to perform communication between the participating nodes. These architectures consider some assumptions about the exchange of messages. For example, the TCP/IP architecture, used in the Internet, was developed for environments with low latency and small packet loss [4]. However, this architecture does not perform well for all types of communication networks. There are network environments, considered challenging, that have the difficulty of maintaining end-to-end communication with low latency and small packet loss (e.g., TCP needs a convergence layer to operate in such environments [11]). One of the most commonly used architectures for these environments is called Delay/Disruption Tolerant Networking (DTN) [4], which

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 167–178, 2019.
https://doi.org/10.1007/978-3-030-24289-3_14

is standardized by the Internet Engineering Task Force (IETF). The protocol implied in such standard is the Bundle Protocol (BP) [2,10].

The DTN communication particularities impact on the tasks execution, among them management ones. In any case, a DTN has network management needs as well as traditional computer networks [1]. In this context, active measurement mechanisms are an important tool to monitor and the health of a network as a whole. Such mechanisms inject synthetic traffic into specific network paths to measure the network performance in terms of, for example, delay, jitter, and packet/frame loss. Due to DTN characteristics like high latency and frequent disconnections, the TCP/IP network management tools tend not to perform in an adequate way. Such tools usually need closed loops that depends on the timeliness of management messages among management entities (e.g, manager and agent). Besides that, TCP/IP network management produces "chatty" interactions, which are feasible only in low delay environments.

The use of standardized two-way measurement protocols, such as the Two-Way Active Measurement Protocol (TWAMP) [6], can provide detailed information about round-trip metrics in an interoperable fashion. The evaluation of these metrics is performed using the results obtained during the exchange of test messages between two nodes. However, these protocols were designed to operate on a TCP/IP network and their direct use in a DTN would hardly present the expected results. In this context, there are some measurement *ad hoc* tools, such as *DTNperf* [3], that can perform two-way measurements. Unfortunately, such tools does not follow measurement standards either in the employed architecture or in the format of test messages. This can hamper the interoperability of measurement mechanisms among heterogeneous DTN nodes and even the comparison of different measurement results.

The present work presents the Delay/Disruption-Tolerant Measurement Protocol (DTWAMP), an extension for TWAMP to provide two-way measurements in DTN. Such extension allows the use of a standardized protocol in a DTN environment to deliver different round-trip metrics between two DTN nodes. The main contribution is to support interoperable monitoring tasks in DTN. The performed evaluation highlighted the feasibility and the performance of the proposed protocol. Furthermore, the experiments were conducted using a proof of concept implementation, developed using a well-established DTN framework.

This paper is organized as follows. In Sect. 2, the background on Active Measurements and DTN is presented. The DTWAMP is described in Sect. 3. In Sect. 4, the evaluation of the proposed solution is depicted. The Sect. 5 presents the related work. Finally, the concluding remarks are described in Sect. 6.

2 Background

This section presents the main background concepts for the proposed solution depicted in the present work. These concepts are divided into two subsections. In the first subsection, active measurement mechanisms are highlighted, specially regarding TWAMP. This section closes with a discussion about network management properties and challenges in DTN.

2.1 Active Measurement Mechanisms

Active measurement mechanisms can be employed in different contexts, such as pre-deployment service validation and live network-wide SLA monitoring. A well-defined injection of measurement traffic (i.e., test messages) in the networking infrastructure is usually called a measurement session [9]. In this context, active measurements are performed either one-way or two-way (i.e., round-trip). Two-way measurements, which are common in TCP/IP networks, employ time stamps applied at the echo destination to achieve better accuracy. Thus, they do not require synchronization between local and remote clocks. However, it is difficult to isolate the direction in which performance issues are experienced using two-way measurements.

The IETF's TWAMP [6] supports round-trip measurement through the exchange of timestamps between two hosts. In general, TWAMP architecture is usually composed of two hosts having specific functions, the client node (the controller) and the server node (responder). TWAMP consists of two protocols, the TWAMP-Control and the TWAMP-Test. In TWAMP-Control, there are at least eight exchanges of messages between the controller and responder to establish a test session. After such establishment, the test session is started and TWAMP-Test is responsible for transmitting test messages of the same size between the nodes in both directions. The results obtained during this exchange of messages is used to evaluate the network performance between the controller and the responder [6].

TWAMP also has a reduced version, called TWAMP Light, in which it is not necessary to exchange control messages to establish the measurement session [6]. TWAMP Light does not specify how the establishment of such session between the controller and the responder is performed, thus it assumes that a previous relation exists, for example, through a configuration file. In a TWAMP Light measurement session, there is only the exchange of test message (i.e., there is no TWAMP-Control). Since TWAMP-Control requires several messages to configure the measurement session, the use of TWAMP Light provides quicker measurement results and less measurement overhead.

2.2 Delay/Disruption-Tolerant Management

The IETF DTN architecture is described in RFC 4838 [4]. Such architecture combines the use of a message switching technique and the persistent data storage, defining an overlay layer called the Bundle Layer, located between the transport layer and the application layer. Devices that use the bundle layer are considered as DTN nodes, and the messages carried by such nodes are called bundles [4]. The Bundle layer has an aggregation protocol, the Bundle Protocol (BP) [10], which defines the format of bundles as well as their use. The format of the bundles is defined by a mandatory primary block, an optional payload block, and a set of optional extension blocks [4].

DTN management is required to assure an adequate operation in DTN. Traditional management solutions, such as Simple Network Management Protocol

(SNMP) and Network Configuration Protocol (NETCONF), are examples of management protocols that can not be used in DTN environments due to the high delay/disruption-tolerant end-to-end communications. Although there are works which present solutions for the management of DTNs based on these protocols (e.g., SNMP [7]), they were either developed for specific DTN scenarios or still need implementation.

DTN management cannot employ protocols that need to establish control loops across the network infrastructure. Thus, it is not possible for the management application to be fully aware of the local network conditions at the remote device to be managed. This is a critical constraint in defining how management tools should define, transmit, and receive their data. In this context, the Asynchronous Management Architecture (AMA) and the Asynchronous Management Protocol (AMP) [1] relax this constraint for management tasks. Despite the development of AMP, there are no standards for active measurement protocols to be deployed in DTN. In this context, even with less interactions, TWAMP Light still is not tailored for DTN.

3 Delay/Disruption-Tolerant Measurement Protocol (DTWAMP)

The present paper proposes DTWAMP, an extension of TWAMP to provide two-way measurements in DTN. In these networks, the delivery of a message to its destination can take a significant amount of time (e.g., hours). This ends up influencing how DTN nodes communicate with each other, as well as services such as network management. The DTWAMP can be useful for calculating performance metrics, such as delay, jitter, and bundle loss within the DTN infrastructure. For example, with the measurement results of the bi-directional delay between two DTN nodes, DTN applications can configure the required life time that an application message must have. In this context, DTWAMP is an adaptation of TWAMP Light for DTN environments. DTWAMP has a simple logical (depicted in Fig. 1, composed by the *controller*, which initiates the emission of test messages, and the responder, which replies test messages.

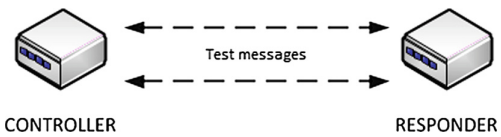


Fig. 1. DTWAMP logical model

The following assumptions are considered for the proposed solution. The DTWAMP works with TWAMP non-authentication mode test messages, thus it is not applied any security controls to test messages exchanged between nodes.

This is because TWAMP Light does not have TWAMP-Control, which is necessary to provide authentication and encryption [6]. In addition, DTWAMP controller assumes that the responder is reachable and start replying test messages as soon as it receives them. Also, to perform the exchange of TWAMP-Test messages in a DTN, such messages must be adapted to the BP format. In this context, we assume that is possible to transport TWAMP-test messages using the BP in most DTN infrastructures. Finally, we modeled "collaborative" intermediate DTN nodes in the sense that they may carry all test messages that are exchanged between the controller and the responder.

This section is organized as follows. First, the messages format employed by DTWAMP is presented. After that, the proof of concept implementation of DTWAMP is described.

3.1 Messages Format

The TWAMP establishes that test messages exchanged between the controller and the responder during the TWAMP-Test use fields with fixed sizes [6]. This is done to ensure that the test messages transmitted during the test session have equal size in both directions. Therefore, DTWAMP employs fixed sizes in the controller and responder data that will be transmitted by the block text data of the payload block of the bundle. In this context, the BP does not specify the size of the *block body date* field of the payload block of the bundle since it is defined as a variable size [10]. Thus, since the responder has a larger amount of information to provide, we defined a specific *packet padding* field in the controller message to match the responder message.

DTWAMP messages are transported through bundles in the same sense as test messages used by TWAMP Light are transmitted using TCP/IP. The test messages specified in Fig. 2 are the payload block of the bundles forwarded by the controller and the responder. *Block Type*, *Proc. Flags* and *Block Lenght* are the original fields of the *payload* block of a bundle (represented in green background in Fig. 2) [10]. In terms of the controller, the DTWAMP transmits the *Sequence Number*, *Timestamp* and *Packet Padding* fields (represented in gray background in Fig. 2(a)) as payload data of the bundle. The *Sequence Number* field is an integer beginning with 1 that is incremented by the number of bundles sent. The field *Timestamp* has the time registered before sending the bundle to the responder node. The packet padding field has the fixed value of 20 bytes to be able to match the *payload* size of controller bundles to the responder ones.

The responder employs the *Sequence Number*, *Timestamp*, *Receive Timestamp*, *Sender Sequence Number*, and *Sender Timestamp* fields (represented in gray background in Fig. 2(b)). The *Sequence Number* field uses the same *Sequence Number* provided by the controller. The *timestamp* field employs the time recorded before sending the bundle to the controller node. The *Receive Timestamp* field uses the time recorded when the *controller* bundle is received. The *Sender Sequence Number* and *Sender Timestamp* fields have the values reported by the controller.

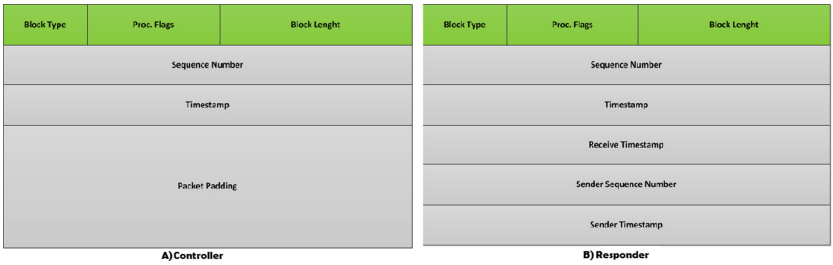


Fig. 2. DTWAMP test messages format

The IETF DTN architecture uses *timestamps* since it depends on the synchronization between DTN nodes to perform activities such as routing between nodes and the deletion of bundles according to their expiration time [4]. Although TWAMP defines *timestamps* as the number of seconds from the beginning of the year 1900 [6], DTWAMP, as a DTN application, uses the *timestamp* defined by RFC 5050 [10] which employs the number of seconds from the beginning of year 2000.

The fields used by the DTWAMP controller and responder as bundle payload data are based on the TWAMP non-authentication test messages. In this context, not all fields of TWAMP-test messages are relevant for the DTN environments. The *error estimate*, *TTL* (Time To Live), and *MBZ* fields are not used by the DTWAMP. The *error estimate* field is not used because this field is a calculation that shows the error and synchronization estimate between the test messages. As in a DTN environment there may be constant oscillations during communication between nodes, such as high latency and disconnections, DTWAMP does not use this field.

The *TTL* field, which is used by the responder in TWAMP, is another field that was not used in the DTWAMP. This field is defined according to the TTL from the IP header of the packet received by the TWAMP node [6]. Since the primary block of each bundle has a *lifetime* field that determines the payload lifetime of the bundle [10], the *TTL* field was not adapted for the DTWAMP.

3.2 Implementation

We implemented DTWAMP as a proof of concept using the C programming language. The *dtnperf* application, which is an open source application made available through the DTN2 reference implementation, was used as the starting code. The DTWAMP controller is based on the *dtnperf-server* and the DTWAMP responder is based on the *dtnperf-client*.

The DTWAMP implementation defines the controller as responsible to initiate message exchange, to calculate the *round-trip* metrics, and to display the measurement results. The responder just acts as a reflector. The controller computes the raw time and number of transmitted bundles and the round-trip results as well as summary information (e.g., average round-trip delay for each direction).

The DTWAMP controller application provides two modes of operations for reporting the round-trip delay between two DTN nodes. The first option, *time mode*, defines the duration of the measurement session (in seconds). In this mode, if specified duration ends during the transmission of a bundle, the controller still wait for this bundle. The second mode of operation, *bundle mode*, defines the number of bundles to be exchanged during a measurement session between two DTN nodes.

4 Evaluation

The objective of the evaluation presented in this section is to demonstrate the feasibility of the DTWAMP in different scenarios when performing the round-trip delay between two DTN nodes. First, the environment used for the evaluation is described. Then, the performed experiments are depicted. Finally, the experimental results are discussed.

4.1 Experimental Environment

The DTN environment used in the evaluation is the DTN2 reference implementation[1]. The DTN2 implements the BP as defined on the RFC 5050 [10]. The BP agent and its support code is implemented as a unix daemon (dtnd). The DTN applications interact with *dtnd* through Remote Procedure Calls (RPCs). The employed operation system is Ubuntu Linux.

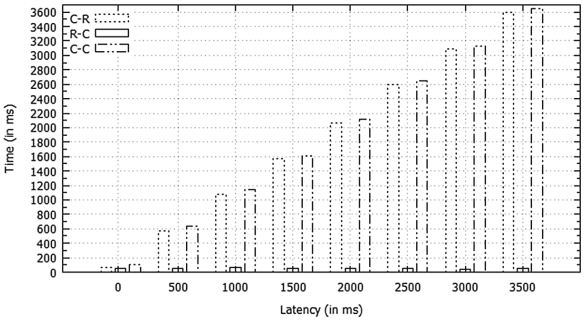
The network delay was configured individually on the DTN nodes using the *netem*[2]. The delay was configured as specified in the experiments. The DTN nodes use Network Time Protocol (NTP) to ensure synchronization. The experiments were repeated ten times and the variance of the obtained results was minimal.

4.2 Experiments

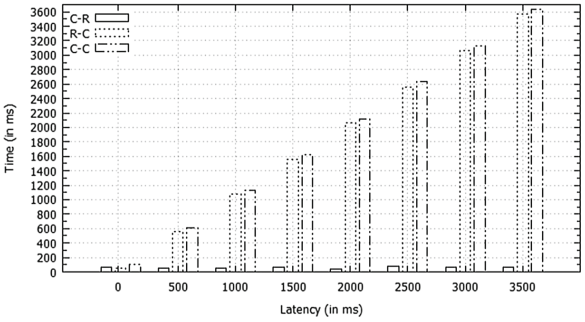
The first experiment was performed using DTWAMP *time mode* (configuration - 600 s) and its results can be visualized in Fig. 3. Initially, the experiment was executed without any additional delay in the environment to provide a baseline (in the bars indicated as "0"). Then, the delay was initially set with 500 milliseconds (ms). After that, the delay was incremented in each subsequent experiment in 500 ms steps, until the last experiment (3500 ms). During this experiment, there was no other network traffic between the DTN hosts.

Figure 3 shows the results obtained from the first experiment. In this figure, "C-R" represents the delay in the controller node - responder node direction; R-C represents the delay in the responder node - controller node direction; and C-C represents the round-trip delay. In this experiment, it is possible to verify that the delays collected by DTWAMP are similar to the delay values configured on the netem.

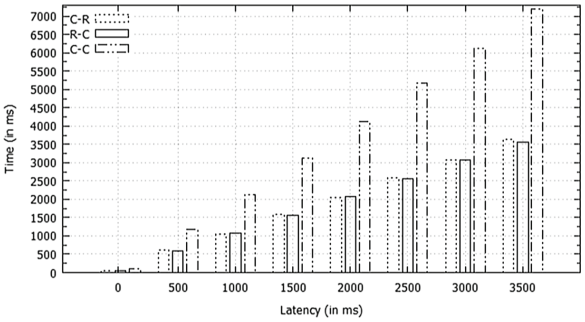
[1] DTN2 source code - https://sourceforge.net/projects/dtn/.
[2] netem - https://wiki.linuxfoundation.org/networking/netem.



(a) VM1



(b) VM2



(c) VM1 and VM2

Fig. 3. DTWAMP measurement results obtained with delays configured in the VMs

The largest difference between the configured and the measured delay considering VM1-VM2 is found on the execution with 2500 ms. However, even in this experiment, the times obtained in VM1 (2594 ms) have a small difference, being only 1.21 % greater than those found in VM2 (2563 ms). Regarding the VM2-VM1, the biggest difference is found in the experiment with 500 ms latency.

In this experiment, the dealays collected in VM1 (630 ms) are only 3.45 % higher than those found in VM2 (609 ms).

If we compare the bundle output times of the controller to the response of Fig. 4 with the experiment from VM1, the difference is minimal between them. The biggest difference is found in the experiment with 3,500 ms latency. In this experiment, the times obtained in Fig. 4 (3628 ms) are only 1.09 % greater than those found in the VM1 latency experiment only (3589 ms). With respect to the output times of the texting respond to the controller of the experiment from VM2, the major difference compared to Fig. 4 is found in the experiment with 500 ms latency. In this experiment, the times obtained in Fig. 4 (577 ms) are 4.15 % greater than those found in the VM2-only latency experiment (554 ms). In these comparisons, the biggest difference found is with respect to the experiment with 500 ms latency of Fig. 4 (1182 ms) and 1,000 ms latency of the experiment with latency only in VM2 (1135 ms). In this comparison, the *round-trip* time of the controller in Fig. 4 is 4.14 % greater than that found in the VM2 experiment.

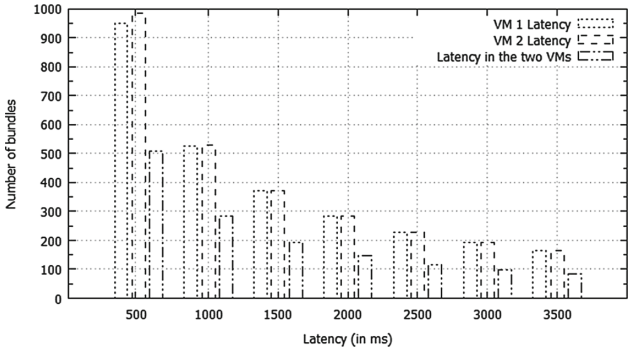


Fig. 4. DTWAMP results obtained with delays specified on both VMs and the number of bundles sent in latency experiments

The number of bundles sent during the delayed experiments decreased due to increased delay. Figure 4 shows the number of bundles sent during the DTWAMP operation in each experiment with latency performed. In Fig. 4 it can be seen that the number of bundles sent during the experiments of Fig. 3 are equivalent. Only in the experiment with 500 ms latency, VM2 sent 32 bundles more (3.36%) than VM1. However, if we compare with the experiments in Fig. 4 (delays in the two VMs), the number of sent bundles decreased by 48.37% (476 bundles less). This number is expected since delay is configured in both directions.

The initial experiments were performed under no background traffic.Then, additional experiments were executed to demonstrate the operation of the DTWAMP when there is traffic in the DTN infrastructure. The network traffic considered for this experiments was the transfer of an 1 GB file between two DTN nodes. Before the experiments were carried out, the average transfer time

of such file was calculated in two situations: transfer of the file through the *dukto*[3] application over TCP/IP and the transfer of the file using bundles, through the *dtncp* application (from the DTN2 implementation). The average file transfer time by *dukto* was 40 s, while *dtncp* was 70 s.

The first two experiments were performed with the DTWAMP in the *time mode* configured with 40 and 70 s respectively, under normal conditions, and their results can be visualized in Table 1. The following experiments were performed as follows: In VM1, the file was transferred from the same to VM2 through *dukto* and the DTWAMP was executed with 40 s of time. Then the file was transferred through *dtncp* and executed the DTWAMP with *time mode* of 70 s. In both experiments, the DTWAMP was executed in parallel with the transfer of the file between the nodes. Subsequently, the same experiments were performed from VM2. The results obtained can be visualized in Table 1. In the experiments performed both from VM1 and from VM2, it can be observed that the average times presented have differences according to the transfer mode used in each experiment. In addition, it can be noted that the bundle transfer from VM2, as compared to VM1, had a total 86.93% greater delay. In addition, the amount of bundles sent was 40.22% lower.

Table 1. Comparation between the file transfer experiments from VM1 and VM2

	C-R	R-C	C-C	Sent bundles
Time mode 40 ms	46 ms	35 ms	83 ms	487
Time mode 70 ms	47 ms	33 ms	80 ms	867
Dukto transfer VM1-VM2	93 ms	258 ms	352 ms	115
Dukto transfer VM2-VM1	253 ms	71 ms	325 ms	124
Bundle transfer VM1-VM2	511 ms	261 ms	773 ms	92
Bundle transfer VM2-VM1	472 ms	972 ms	1445 ms	55

4.3 Discussion

The results demonstrated in Fig. 4 and Table 1, in summary, prove that the DTWAMP can be executed in latency environments, as long as they are supported by the DTN2 standard configuration. For larger latencies, you need to check for additional DTN2 settings. In addition, with the knowledge of the estimated time of sending and receiving a message between two DTN nodes, the other DTN applications can define the lifetime of this message. If you can not change the lifetime of the message, DTN applications can also analyze whether the message can be sent and/or received within the estimated time, thus avoiding unnecessary traffic.

Experiments performed during file transfer as a bundle demonstrate that the node that is sending the file has its most affected time than the one receiving the

[3] Dukto R6 - http://www.msec.it/dukto.

file as a bundle. While in the transfer by the textit dukto, through the TCP/IP network, the opposite happens, the node that is receiving the file has its time more affected than the one that is sending the file. This experiment can also be considered as an experiment in which there was an increase of processing by the nodes, since the file was not transferred by the DTN network.

5 Related Work

This section presents briefly some measurement tools which use the DTN2 implementation. These tools can be characterized by the execution of active measurements to deliver their metrics [8].

Davis and Doria [5] proposed *dtnping*, which is a DTN version of TCP/IP ping. Comparing with DTAMP, dtnping is also a active measurement mechanism that exchanges bundles between DTN nodes. However, dtnping does not follow a standard to define neither its architecture or its messages. This hampers fair performance evaluations on different DTN infrastructures. Besides that, it also impacts the use of comparable measurement tools.

Caini *et al.* [3] proposed *dtnperf*, which is a DTN application similar to TCP/IP iperf. The main dtnperf feature is to provide the throughput between two DTN nodes. The architecture of this application is similar to the DTWAMP one, composed by two entities: the *dtnperf-client*, which computes the performance metrics and the configuration options, and the *dtnperf-server*, which only reflects test messages. However, *dtnperf* uses the bundle receiving flag, while DTWAMP employs specific *timestamps* during the communication of two DTN nodes. In addition, *dtnperf* does not respect any measurement standard (in the same sense as dtnping).

6 Final Remarks

One of the major challenges in DTN management is the lack of an end-to-end connection between DTN nodes. This is one of the main reasons why traditional management tools are not directly employed in DTN. In the context of measurement mechanisms, there is neither a standardized measurement protocol or one based on Internet standards. This hampers the execution of measurement tasks in different DTN infrastructures and measurement tools.

The present paper describe DTWAMP, an extension for TWAMP to provide two-way measurements in DTN. Such extension adapts a special TWAMP mode, TWAMP Light, to delay/disruption-tolerant infrastructures. In order to demonstrate the feasibility of the proposed solution, experiments were performed to show the DTWAMP operation considering configured delay as ground truth. In addition, synthetic bundle traffic was added to verify how DTWAMP would react when there were other types of bundles being carried by DTN nodes. DTWAMP can be used by DTN applications to produce the estimated delay for an application bundle, which can be used to predict the Quality of Service perceived for this application.

Although the DTWAMP presented positive results in the performed evaluation, new features can improve the protocol. For example, the DTWAMP messages could be adapted to use TWAMP-Test authenticated and encrypted mode in order to improve their security features. Besides that, regarding the transport of test messages, the DTWAMP sends them through a exclusive bundle which incurs in a significant overhead. Thus, in future work, we aim at adapted the DTWAMP messages to be transmitted along with other bundles either as an opportunistic send or piggybacking. Finally, DTWAMP was not evaluated in environments that have frequent disconnects. Therefore, it is necessary to include such environments in future evaluations to highlight the properties of the protocol in DTN scenarios with pose disconnections.

References

1. Birrane, E.: Asynchronous management architecture. Work in progress as an internet-draft, draft-birrane-dtn-ama-06 (2017)
2. Burleigh, S., Fall, K., Birrane, E.: Bundle protocol version 7. Work in progress as an internet-draft, draft-ietf-dtn-bpbis-10 (2017)
3. Caini, C., d'Amico, A., Rodolfi, M.: Dtnperf_3: a further enhanced tool for delay-/disruption-tolerant networking performance evaluation. In: 2013 IEEE Global Communications Conference (GLOBECOM), pp. 3009–3015. IEEE (2013)
4. Cerf, V., et al.: Delay-tolerant networking architecture. RFC 4838 (informational) (2007)
5. Davies, E., Doria, A.: Functional specification for DTN infrastructure software comprising RFC 5050 bundle agent and associated components. version 1.2 (2010)
6. Hedayat, K., Krzanowski, R., Morton, A., Yum, K., Babiarz, J.A.: Two-way active measurement protocol (TWAMP). RFC 5357 (request for comments) (2008)
7. Isento, J.N., Dias, J.A., Canelo, F., Rodrigues, J.J., Proenca, M.: Moni4VDTN: a monitoring system for vehicular delay-tolerant networks. In: 2012 IEEE International Conference on Communications (ICC), pp. 1188–1192. IEEE (2012)
8. Nobre, J.C., Mozzaquatro, B.A., Granville, L.Z.: Network-wide initiatives to control measurement mechanisms: a survey. IEEE Commun. Surv. Tutor. **20**(2), 1475–1491 (2018). https://doi.org/10.1109/COMST.2018.2797170
9. Roy, A., Acharya, T., DasBit, S.: Quality of service in delay tolerant networks: a survey. Comput. Netw. **130**, 121–133 (2018). https://doi.org/10.1016/j.comnet.2017.11.010. http://www.sciencedirect.com/science/article/pii/S1389128617304188
10. Scott, K., Burleigh, S.: Bundle protocol specification. RFC 5050 (request for comments) (2007)
11. Sipos, B., Demmer, M., Ott, J., Perreault, S.: Delay-tolerant networking TCP convergence layer protocol version 4. Work in progress as an internet-draft, draft-ietf-dtn-tcpclv4-06 (2018)