

Performance Evaluation of NETCONF Protocol in MANET Using Emulation

Weichao Gao, James Nguyen, Daniel Ku, Hanlin Zhang and Wei Yu

Abstract The Mobile Ad-hoc Network (MANET) is an emerging infrastructure-free network constructed by self-organized mobile devices. In order to manage MANET, with its dynamic topology, several network management protocols have been proposed, and Network Configuration Protocol (NETCONF) is representative one. Nonetheless, the performance of these network management protocols on MANET remains unresolved. In this paper, we leverage the Common Open Research Emulator (CORE), a network emulation tool, to conduct the quantitative performance evaluation of NETCONF in an emulated MANET environment. We design a framework that captures the key characteristics of MANET (i.e., distance, mobility, and disruption), and develop subsequent emulation scenarios to perform the evaluation. Our experimental data illustrates how NETCONF performance is affected by each individual characteristic, and the results can serve as a guideline for deploying NETCONF in MANET.

Keywords NETCONF · YANG · MANET · Configuration management · Network management · Emulation

We would like to thank our Dr. Robert G. Cole for initially starting this effort and giving us feedback.

W. Gao (✉) · H. Zhang · W. Yu
Department of Computer and Information Systems, Towson University,
Towson, MD 21252, USA
e-mail: wgao3@students.towson.edu

H. Zhang
e-mail: hzhang4@students.towson.edu

W. Yu
e-mail: wyu@towson.edu

J. Nguyen · D. Ku
US Army CECOM Communications-Electronics Research,
Development and Engineering Center (CERDEC), Fort Sill, USA

1 Introduction

The emerging MANET has an infrastructure-free and dynamic network topology constructed by self-organized mobile devices. Compared to the traditional wired and access-point-based wireless network topologies, MANET has more flexible structures due to the frequent joining and exiting of nodes in the network. This flexibility, on one hand, enables MANET to better fit rapidly changing application scenarios, but on the other hand, increases the difficulty in managing the network.

Various network management protocols have been proposed, including NETCONF [1], Simple Network Management Protocol (SNMP) [2]. Although a large number of research efforts have focused on the performance evaluation of these protocols in wired networks, the performance of these management protocols on MANET are largely untested. In the research that does address this issue, various network simulation tools [3], such as NS-3 [4] and OMNET++ [5], were developed to evaluate the performance of protocols in MANET. However, these simulations provide the overall performance in the simulated topologies only, and the performance in real-world MANET topologies has not been thoroughly evaluated. In addition, while the simulated topologies were considered as the whole entities in the evaluation where the scale (i.e. number of nodes), random mobility pattern, and other parameters such as bandwidth and transmission range were present, the impacts of the individual characteristics of MANET on its performance are not clear.

In our investigation, we quantitatively evaluate the performance of NETCONF in the MANET environment. Unlike prior research that is primarily based on simulation, we leverage the network emulation tool, Common Open Research Emulator (CORE), to carry out the performance evaluation of NETCONF in MANET. To design the emulation scenarios, we develop a framework that captures the key characteristics of MANET (i.e., distance, mobility, and disruption). Specifically, distance encapsulates the structures that separate any two communicating nodes in MANET, mobility represents the leaving and joining of nodes, and disruption characterizes the environmental hindrance to the delivering packets over the network. Based on the designed framework, we develop a set of scenarios for the performance evaluation. The impact of distance is tested via increasing the number of hops in the scenarios where the MANET topologies are fixed. The impact of mobility is examined through varying the leaving time in the scenarios where a target node temporarily leaves the transmission range and losses connection. Finally, we introduce disruptions into the aforementioned scenarios to evaluate their impact.

We conduct extensive experiments to validate the performance of NETCONF in various scenarios. Our experimental data illustrates the performance of NETCONF with respect to individual characteristics and our data can be used as a guideline to deploy NETCONF in MANET. For example, the incremental hop can increase the delay in linear and exponentially enlarges the impact of disruption, and the disruption not only increases the delay for packet transmissions, but also reduce the capability of the NETCONF requests to tolerate the out-of-range time of a leaving node in the MANET topology.

The remainder of this paper is organized as follows: We conduct a literature review of the MANET and NETCONF and introduce the emulation tool CORE in Sect. 2. In Sect. 3, we describe the decomposition of MANET topology and the framework of simulation scenarios. In Sect. 4, we describe the testbed configuration and present the evaluation methodology and results. Finally, we conclude the paper in Sect. 5.

2 Background and Related Work

In this section, we provide the background of our work and conduct the literature review of areas relevant to our study.

MANET: MANET is a network with highly dynamic topologies that are constructed by mobile nodes. In a MANET, infrastructures such as access points are not required because nodes are self-organized and are able to act as routers in the network. These mobile nodes are able to move frequently and independently, and their neighboring nodes are continuously changing. The versatility of MANET makes it ideal for a variety of applications, including tactical networks, disaster recovery, entertainment, and the emerging Internet of Things (IoT) [6, 7].

With the advance of wireless technologies, there has been a tremendous increase in number of mobile nodes, leading to larger and more complex, topologies of the MANET. Many research efforts, for that reason, have investigated MANET in the following areas [7]: (i) Routing [8], (ii) Energy conservation [9], (iii) Quality of Service [10], (iv) Security [11], and (v) Network Management.

Several investigations have been devoted to studying network management protocols [12]. These efforts include the comparison of existing management protocols, and implanting these protocols in MANET environments. The former one focused on the performance comparison of protocols in the wired or wireless networks [13, 14]. For example, Slabicki and Grochla in [14] conducted the performance evaluation of SNMP, NETCONF and CWMP on a tree topology-based wireless network. The later one focused on how the existing protocols performed in a MANET environment [15, 16]. For example, Herberg et al. [15] evaluated the performance of SNMP in an Optimized Link State Routing Protocol (OLSRv2)-routed MANET environment.

NETCONF and YANG: NETCONF [1], which is an emerging network configuration protocol, is one of the network management protocols and its performance has been studied in wired or wireless networks based on access points [13, 14]. It was developed and standardized by the Internet Engineering Task Force (IETF) (in RFC6241), which provides the mechanism to retrieve, edit, and remove the configuration of devices within the network. NETCONF is a session-based protocol that enables multiple operations in the configuration procedures, and its client-server infrastructure over the secure transport (usually SSH over TCP) ensures the reliable and secure transactions.

The NETCONF protocol includes four layers of content, operations, messages, and secure transport. The content layer consists of the configuration data and notification data that is formed in XML. The operations layer defines the base protocol operations, such as `< get >`, `< edit – config >`, and `< delete – config >`, to retrieve and edit the configuration data. These operations and data are encoded in the message layer as remote procedure calls or notifications. The messages are then transported over the secure transport layer between the client and server. YANG [17], the acronym for “Yet Another Next Generation”, is the data modeling language for NETCONF protocol developed by IETF (in RFC 6020). It represents data structures in an XML tree format and is used to model the configuration and state data manipulated by the NETCONF protocol, NETCONF operations, and NETCONF notifications.

In the following, we describe the characteristics of NETCONF and YANG [18]: (i) *Session-oriented protocol over Secure Shell (SSH)*: The NETCONF protocol considers SSH for its mandatory transport protocol. Other transports, including Transport Layer Security (TLS), Simple Object Access Protocol (SOAP), are considered to be optional. (ii) *Data-driven content with YANG*: The YANG is used to define the server API contract between the client and the server. (iii) *A stateful transaction model*: The NETCONF protocol is designed for one server and one client pair. The client and server exchange `< hello >` messages at the start of the session to determine the characteristics of the session that is always initiated by the client. Via it, the client then learns the exact data model content of YANG supported by the server. (iv) *RPC exchange messages encoded in XML*: A client encoded an RPC message in XML and sends it to a server. The server, in turn, responds with a reply, which is also encoded in XML. (v) *Database operations are standardized and extensible*: There are four operations used to manipulate the conceptual data defined in YANG: create, retrieve, update, and delete (CRUD). A datastore, which is a conceptual container with a well-defined transaction model, is defined to store and access information. The server advertises the datastores it supports in the form of capability strings.

CORE: Unlike the simulation-based approaches of other research efforts, we implement an emulation-based approach for the performance evaluation in this study. We utilize the Common Open Research Emulator (CORE) [19] to emulate the MANET environments. CORE is a python framework, providing a graphic user interface for building emulated networks. Consisting of an graphic user interface for drawing topologies of lightweight virtual machines (nodes) and Python modules for scripting network emulation, CORE is able to emulate a network environment (wired and/or wireless) and run applications and protocols on the emulated nodes. It also enables the connection between emulated networks and live networks. Using CORE, we are able to establish the scenarios for the performance evaluation of NETCONF in MANET environments.

3 Our Approach

In our data transaction model, we emulate the data traffic between one pair of nodes (source and destination), managed by NETCONF. We define “connected nodes” to be any pair of nodes, which are able to communicate in the MANET. Via deconstructing the complex MANET topology into several key characteristics, we formulate the test configurations to examine the individual and combined effects of these characteristics, and quantitatively measure their impact on the performance. In the following, we first present the key characteristics of MANET dynamic topology and then design the scenarios for emulation.

3.1 Key Characteristics

Figure 1 illustrates the coupling/decoupling model. Choosing n_1 and n_2 as a pair to observe the communication. Once n_2 moves to the range of n_4 as shown in the figure, n_1 and n_2 are “connected”. During the data transmission between the connected pair, the performance will be impacted by the following three characteristics: (i) Distance, (ii) Mobility, and (iii) Disruption, which will be detailed in the next few subsections.

Distance: The distance between a pair of connected nodes is characterized by the combination of the physical and “processing” distance between them. It can be formalized in general terms as the sum of the delay of all of the hops on the route and the delay of all of the physical distances between each adjacent or connected node on the route. Figure 2 shows an example of the distance of a pair of nodes n_1 and n_2 . The number of hops between n_1 and n_2 is 2 (n_3 and n_4) and the sum of physical distance between n_1 and n_2 is 700 m (250 + 250 + 200). The performance, as a measure of

Fig. 1 MANET coupling/decoupling model

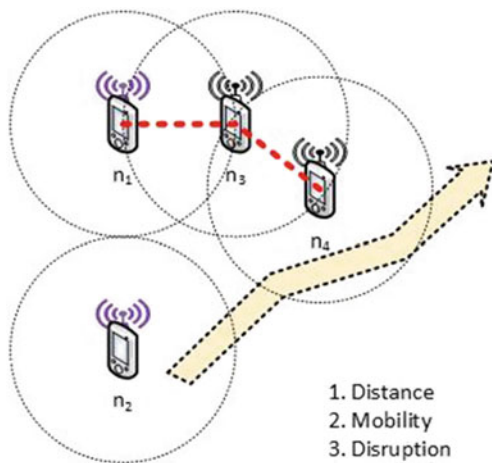
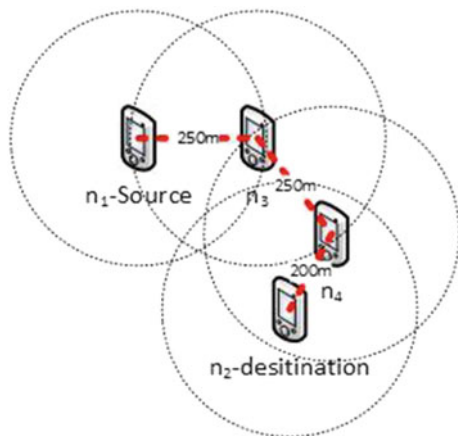


Fig. 2 Distance between two nodes (n_1 and n_2)



network speed or delay, is thus affected by the processing delay of every hop and the physical delay of data transmission over the distance. The latter can be computed by dividing the total physical distance by the speed of light. At any given point in time, both the distance and number of hops will be constants.

Mobility: This is the characteristic of the flexible morphology of the MANET. Mobility allows for the reconfiguration of the nodes, and therefore degrades performance due to the delay raised by disconnection and reconnection. The change in the connection between a pair of nodes can be modeled by three mobility patterns. We name the first pattern *Fixed Connection*, where the route does not change between the source and destination nodes and any of the nodes in the route during the data transmission. This pattern can be considered the same as a static topology. The second pattern is denoted *Same Route Returning*. During the data transmission, one or more nodes move out from the original route and cause the loss of connection between the source and destination nodes. The leaving nodes then returns and recovers the connection through the original route. The third pattern is named *Change Route Returning*. The nodes leave and cause the loss of connection as in the second pattern, but the difference is that the connection is reformed, either by the relocation of the remaining nodes, or after the leaving nodes return through a different route from the original one. This situation is considered because the routing table would be updated for the data transaction. Figure 3 illustrates the connection between two nodes n_1 and n_2 as the result of three mobility patterns. The performance affected by the mobility of nodes comes from the delay or failure of data transmission caused by the change of connection status. In the latter two patterns, the original data packet in the transmission is not delivered due to the loss of connection on the route. The data will be either delivered by the retransmission after the route is recovered or reformed, or failed to deliver due to a timeout, according to the retransmission schedule.

Disruption: Disruption is characterized as packet loss. There are many factors that can contribute to disruption, such as weak signal and shielding materials, and

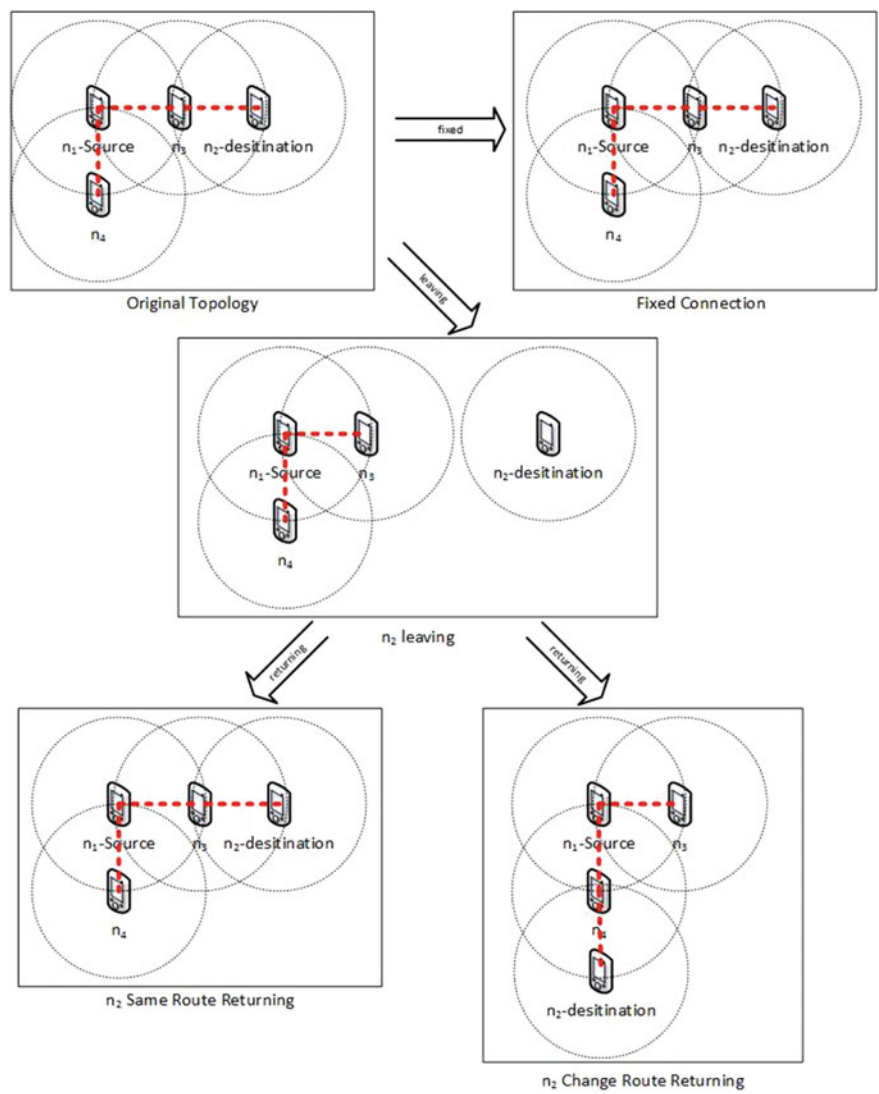


Fig. 3 Mobility patterns of nodes

in general these factors are the result of unintended or unavoidable environmental factors. The disruption results in the packet loss during the data transmission, which triggers the data retransmission and leads to additional delay, overhead, and sometimes the failure of the request or the response. The degree of disruption, with respect to the packet loss rate, can affect the performance of network management.

In a real-world MANET environment, the performance of NETCONF is affected by the complexity of the topologies. In our assumption, there are a variety of factors

in a typical MANET topology, which can affect the performance. Nonetheless, all of them can be categorized into the three key characteristics. For example, the faster the nodes move in MANET, the more frequent the loss of connection and recovery occurs. Another example is that a larger density of nodes can reduce the number of hops between a pair of nodes that is communicating, because it allows the nodes to find shorter routes to reach other nodes. Under this assumption, we are able to create a 3-dimensional framework, generating the emulation scenarios by combining different levels of performance impact in individual dimensions. It also allows us to estimate the performance of NETCONF in a complex MANET environment, and find the conditions that NETCONF can perform as well.

3.2 Emulation Scenarios

To design the emulation scenarios, we create a 3-dimensional framework based on the key characteristics to capture MANET dynamic topology. Figure 4 illustrates this framework.

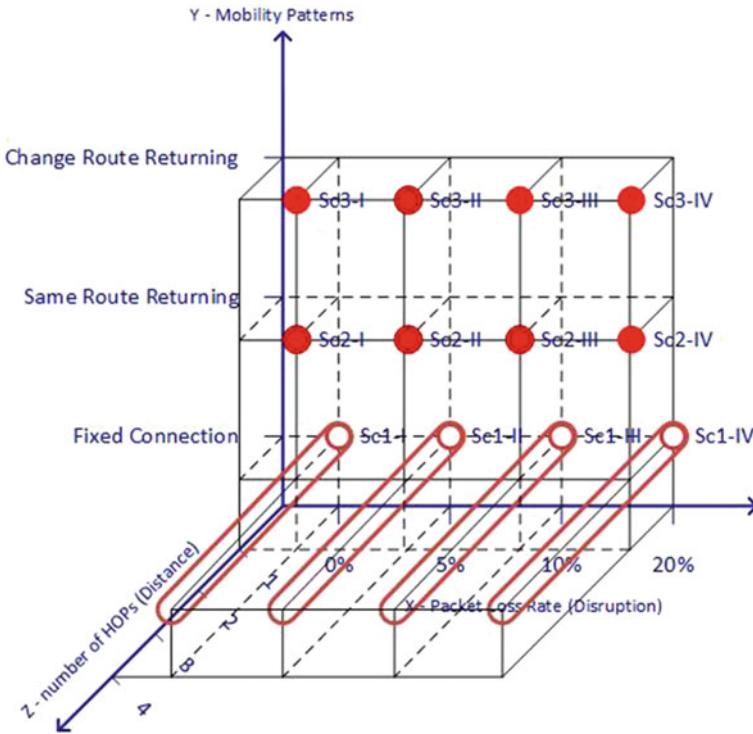


Fig. 4 Framework to design emulation scenarios

		Packet Loss Rate (Disruption)				Variables	Remark
		0%	5%	10%	20%		
Mobility Patterns	Fixed Connection	Sc1-I	Sc1-II	Sc1-III	Sc1-IV	HOPs	
	Same Route Returning	Sc2-I	Sc2-II	Sc2-III	Sc2-IV	Stay-out-time	HOP = 1
	Change Route Returning	Sc3-I	Sc3-II	Sc3-III	Sc3-IV	Stay-out-time	HOP = 1

Fig. 5 Scenario groups

The *X*-axis represents the dimension of *Disruption*, in the form of *Packet Loss Rate*. We evaluate four levels of packet loss rate, 0, 5, 10, and 20 %, to describe the degree of disruption to the MANET. The *Y*-axis represents the dimension of *Mobility*. In our experiment, we set only the destination node of the communicating pair as the candidate node to move, while all other nodes on the route are fixed. As illustrated in Fig. 3, the three mobility patterns are (i) *Fixed Connection*, where the destination node stays static during the data transmission, (ii) *Same Route Returning*, where the node leaves and returns to the previous position to recover the original route to the source node, and (iii) *Change Route Returning*, where the node leaves and returns to a new position and forms a new route. Notice that the duration that a node spends out of the network range can affect the performance, and therefore becomes a variable in our evaluation. The *Z*-axis represents the *Distance* between the pair of communicating nodes, in the form of number of hops. We set the distance between every two adjacent nodes on the route to an equal and fixed value such that the sum of physical distance is a constant multiplied by the number of hops plus one. The number of hops ranges from 0 to 4 in our emulation scenarios. By applying different level in each dimension, we generate 12 emulation scenarios grouped in 3 groups, as shown in Fig. 5.

- **Scenario Group 1** (*Sc1-I*, *Sc1-II*, *Sc1-III*, and *Sc1-IV*): Scenario Group 1 includes 4 subsets that represent the fixed MANET topologies, where the nodes are static during the data transmission. Each subset is evaluated with one of the four levels of packet loss rate as defined in the dimension of disruption, i.e. 0 % in *Sc1 – I*, 5 % in *Sc1 – II*, 10 % in *Sc1 – III*, and 20 % in *Sc1 – IV*. Additionally, each subset is evaluated at values of the variable hops 0 to 4. The goal is to observe how the distance affect the performance in a fixed topology under different levels of disruptions.
- **Scenario Group 2** (*Sc2-I*, *Sc2-II*, *Sc2-III*, and *Sc2-IV*): Scenario Group 2 includes 4 subsets that represent *Same Route Returning*, where the destination node moves out of the transmission range in the MANET, stays out of range for a variable period of time, and returns with the original route recovered. Each subset in this group is tested with the disruption levels described above. For the dimension of distance, we only consider the situation, where the number of hops is 1. This simplifies the scenario and provides the same structure in comparison with the

Change Route Returning scenario. The variable in each subset is the time that the leaving node stays out, named *stay-out-time*. We observe the delay and the success rate by increasing the *stay-out-time* in each emulation subset to evaluate the performance in *Same Route Returning* pattern across different levels of disruption. Another important assumption is that the leaving nodes “jump” out of the range instead of “move”, which omits the time that the leaving nodes moves from the original position to the edge of the signal range. This allows us to evaluate the reconnection delay purely, without the contribution of the delay from varying distance.

- **Scenario Group 3** (*Sc3-I, Sc3-II, Sc3-III, and Sc3-IV*): Scenario Group 3 includes 4 subsets that represent the *Change Route Returning*, where the destination node moves out of the MANET range, stays out of range for a variable period of time, and returns with a new route formed. Just as in Scenario Group 2, we observe the delay and the success rate by increasing the *stay-out-time* in each subset, and evaluate the comparative performance in *Change Route Returning* under different levels of disruption.

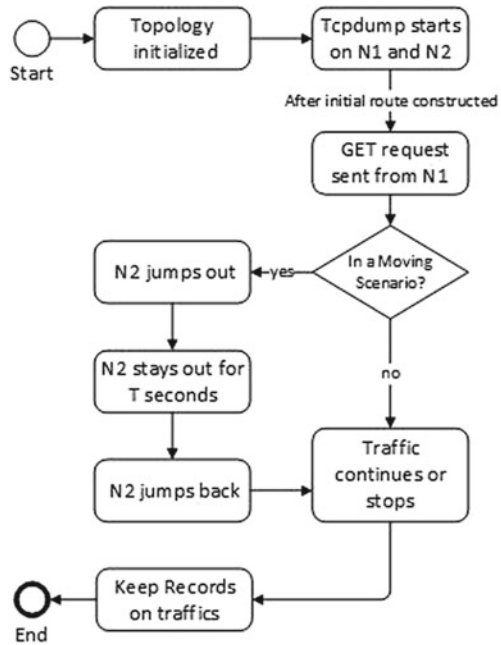
By comparing the emulation results across the scenario groups, we can evaluate the impact of the mobility patterns on the performance of NETCONF. We can also study the impact of the disruption on the performance of network management protocol, by comparing the results across the four subsets in each scenario group.

4 Performance Evaluation

We setup an emulation environment to evaluate the performance of NETCONF based on scenarios designed in Sect. 3. The testbed utilizes CORE v4.8 to emulate the MANET environment. CORE requires a Linux environment, for which we used Ubuntu (v14.04 LTS). OpenYuma [20], which is an open source NETCONF implementation, contains a *netconfd* service for the server node and a *yangcli* client for the client node. Other necessary protocols and services in the emulation environment include OLSRv2 [21] for the MANET topology and SSH for NETCONF to run. We used olsrd2 [22] from the OLSR.org Network Framework (OONF) for the OLSRv2 protocol, and OpenSSH [23] for the SSH service.

The wireless connection of the emulated MANET environment is set to the default value of 54 Mbit/s, with the maximum range of 275 m. During the emulation, data traffic and the mobility script are set to start after the initial MANET topology is constructed. *Tcpdump* is also initialized to capture all TCP traffic, and the trace file is processed by using *awk* scripts for data analysis.

Several customized settings were required to override the default CORE configuration: (i) The OLSRv2 protocol is used in each emulated node with the OONF version. (ii) The SSH is directed the configuration file to a customized file that enables the ports and service for the *netconfd* server. A global key pair is setup for non-password login between emulated nodes. (iii) The service that launches the *tcpdump*

Fig. 6 Emulation process

for both server (n_1) and client (n_2) nodes is added and the request sending command on the client node is added. This configuration can be implemented by either customizing the node configuration in CORE's graphic user interface, or adding it to the Python script.

The quantitative evaluation of the performance of NETCONF is conducted through observing the traffic of simple NETCONF requests in the scenarios that we have created. The emulation in each scenario is designed as a simple NETCONF request and feedback process between a pair of target nodes in the MANET environment. We define the target pair of nodes named n_1 and n_2 for each scenario emulated in CORE. The node n_1 is set as the NETCONF client running *yangcli*, and the node n_2 is set as the NETCONF server running *netconfd*. The data traffic is a simple, one-time NETCONF server login and "GET" request sent from n_1 to n_2 , and the expected feedback of the request from n_2 . All TCP packets sent and received by n_1 and n_2 are recorded by *Tcpdump*.

Figure 6 illustrates the process of each emulation. When the emulation process starts, the MANET topology is initialized and constructed by the pair of target nodes along with the other nodes in between. *Tcpdump* starts to trace and record the TCP packets on both client (n_1) and server (n_2) nodes. Next, a "GET" command with login request, which is delay-triggered to wait for the construction of the MANET, is sent from the client (n_1). If it is in a moving scenario, the mobility script is triggered right after the request, and the server (n_2) starts jumping out from the current topology. The leaving node (n_2) stays out of range for T seconds, and moves back in range.

Metrics	Definition	Measurement
Packet Overhead	The difference between actual packet sent and packet need to send.	Extra packets sent due to the failure caused by any influencers
Largest Delay	The largest time interval of any packet from sending out to receipt of the acknowledgement	The total delay from the leaving of the nodes to the success of next packet transmission.
Recovery Delay	The difference between the largest delay and stay-out-time	The delay for topology reconstruction and packet retransmission.
Threshold of Transaction Failure	The stay-out-time that transactions start to fail	The tolerance of the leaving time at certain disruption level

Fig. 7 Metrics

The data traffic may continue after MANET is reconstructed or stops if the node stays out longer than the threshold. If it is a fixed topology, the server node stays in range during the emulation. Finally, the emulation stops and records everything to a trace file.

By repeating the emulation, changing the value of variables in each subset, and analyzing each trace record, we are able to obtain (i) the actual packets and bytes sent and received by each node, (ii) the number of packets and bytes needed to be sent for a connection, by the sequence number, (iii) the largest delay in each full request/feedback process, which is the largest time interval of any packet from being sent out to being acknowledged, and (iv) the stay-out-time that we define as the time that the leaving node stays out of range. To measure the performance of the NETCONF, we define the metrics in detail shown in Fig. 7. Each emulation scenario with be run 5 times.

Our evaluation is separated into the categories based on the two variables: distance and mobility. The third variable, disruption, is a variability condition for the evaluation of the distance and mobility. We will discuss the disruption when we show the results of scenarios associated with distance and mobility. Additionally, in our emulation we note that the NETCONF request can only be initialized successfully while the client and server nodes are connected. Furthermore, for any node that has left, the request can only be successfully delivered and executed once the route between the pair of nodes is recovered or reformed and the routing table is corrected, before timeout.

Distance: Figure 8 shows the variation of the largest delay from the number of hops in the Fixed Connection scenarios without disruption (Sc1-I defined in Sect. 5). It can be observed from the scatter chart that the largest delay (Y-Axis) increases linearly with the number of hops (X-Axis) between the client and server nodes, indicating a constant increase in delay per hop. Therefore, a steady route can be considered similar to the wired network or wireless network based on access points. Notice that the delay due to the physical distance between two nodes, at less than 1 μ s for the transmission across 250 m, can be discarded because the total delay is in the range of 100 ms, leaving us to simply the delay for the hops.

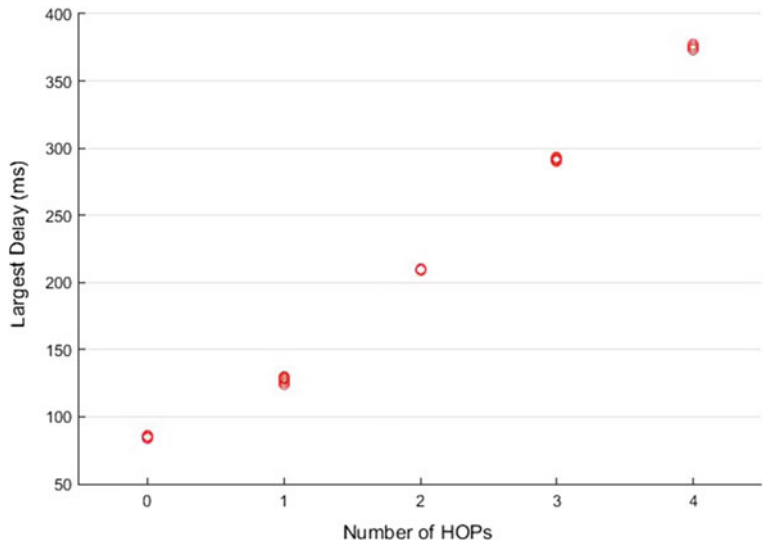


Fig. 8 Largest delay in fixed connection w/o disruption

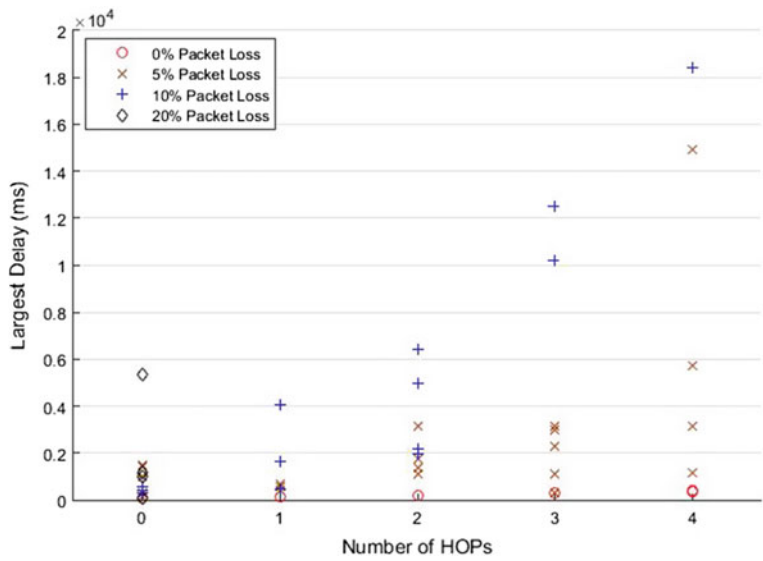


Fig. 9 Largest delay in fixed connection w/disruptions

When disruption occurs, the delay increases with increasing the number of hops. Figure 9 compares the distribution of largest delay by hops under 4 different levels of disruptions (results of the emulations in Group 1: Sc1-I, Sc1-II, Sc1-III, and Sc1-IV defined in Sect. 5). Under a higher disruption level, as observed in the scatter chart,

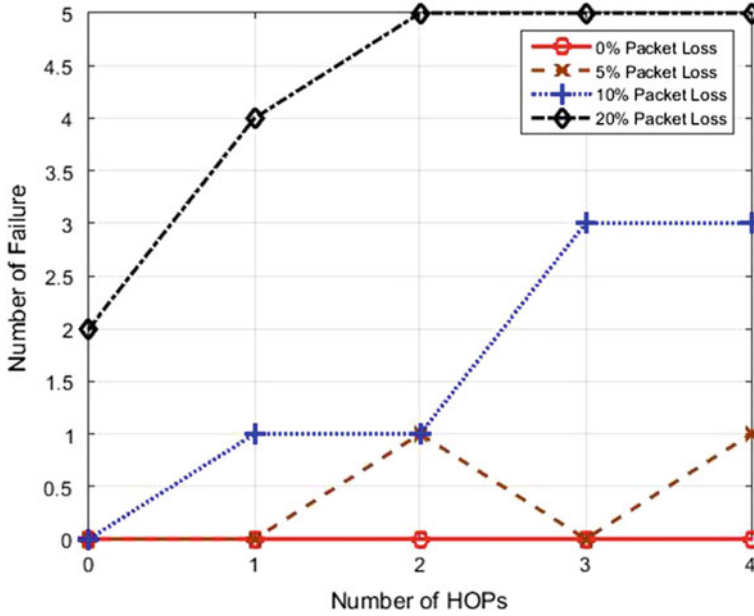


Fig. 10 Transaction failure in fixed connection w/disruption

the upper bound of the largest delay increases significantly by every incremental hop. For example, under a 10 % packet loss rate, the largest delay can increase from 1500 ms at 0 hop to 15000 ms at four hops, where the delay under the 0 % packet loss rate increases only from 86 to 375 ms.

In Fig. 10, we directly compare the number of packet failures by hops under the 4 levels of disruption. In a disruption free environment (0 % packet loss rate), the request does not fail since all packet will be delivered. Once disruption occurs, however, the probability of request failure increases by increasing the number of hops. In these scenarios (Sc1-II, Sc1-III, and Sc1-IV defined in Sect. 5), a packet is lost during the transmission between the adjacent nodes, and every incremental hop exponentially increases the probability of packet loss. For example, when the packet loss rate is 10 %, the overall probability of a packet loss for a 2-hop route is 27.1 % (or $1 - 90\%^3$) and for a 3-hop route is 34.4 % (or $1 - 90\%^4$). As the scatter chart shows, under the 10 % packet loss rate, the NETCONF request fails 1 out of 5 attempts when there is 1 hop between the client and server nodes, and increases to 3 times when number of hops raises to 3.

Once the packet loss triggers the retransmission, the delivery of the packet is delayed, and a greater packet overhead is incurred. Figure 11 compares the number of duplicate packets (overhead) by hops for each level of disruption. In each scenario where the packet loss occurs (Sc1-II, Sc1-III, and Sc1-IV defined in Sect. 5), the overhead increases with increasing hops. This distribution pattern of Overhead-Distance matches the Delay-Distance, where the delay is a measure of the time effect

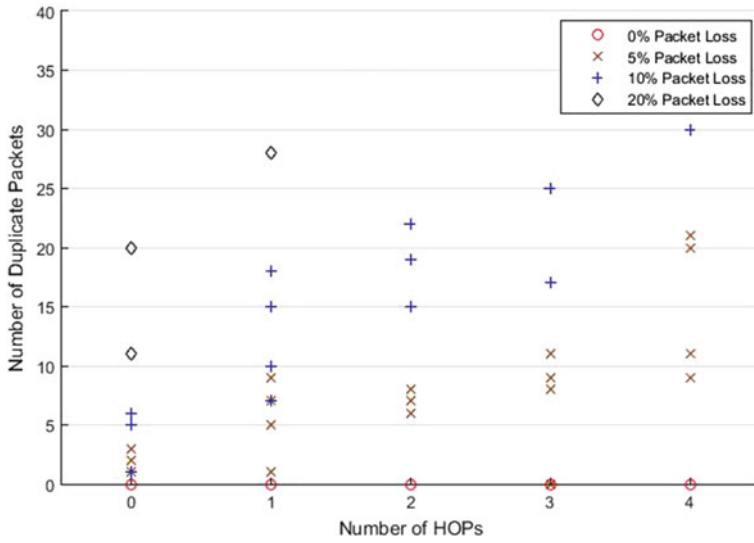


Fig. 11 Packet overhead in fixed connection

of retransmitting packets. Likewise, as the probability of disruption is exponential with distance, so is overhead, as each disruption results in a retransmission. It follows, then, that the Delay-Distance trend also increases exponentially.

The result of emulation in Scenario Group 1 indicates how the distance between two pairs of nodes will affect the performance of the NETCONF traffic in MANET. Greater distance linearly increases the delay in a disruption-free environment, and exponentially increases the delay and overhead proportionally with the probability of disruption.

Mobility: While Scenario Group 1 represents the performance of NETCONF traffic in the steady connection circumstances, Scenario Group 2 and 3 shows the influences resulting from the moving nodes. Excluding the disruption (Sc2-I defined in Sect. 5), Fig. 12 shows the variation of the *Recovery Delay* from the *stay-out-time* in the *Same-Route-Returning* scenarios without disruption (Sc2-I defined in Sect. 5). Theoretically, the time for restructuring a same topology MANET should be steady. The scatter chart, however, indicates a non-linear pattern of *Recovery Delay* by *stay-out-time*. With the increase of *stay-out-time*, the delay jumps to a high level and slowly decreases repeatedly. This indicates that the retransmission may not be triggered right after the recovery of the route, and the time interval from the recovery of route to the actual retransmission varies by *stay-out-time*. When not deducting the *stay-out-time*, Fig. 13 shows the scatter chart of the *Largest Delay* (Y-Axis) by *stay-out-time* (X-Axis). It can be observed that the *Largest Delay* increases in almost double at certain time point and remains the same level until next jump. This is because the time a packet to be re-sent is scheduled after the last failure, once the leaving node moves back and recovers the route before next scheduled transmission

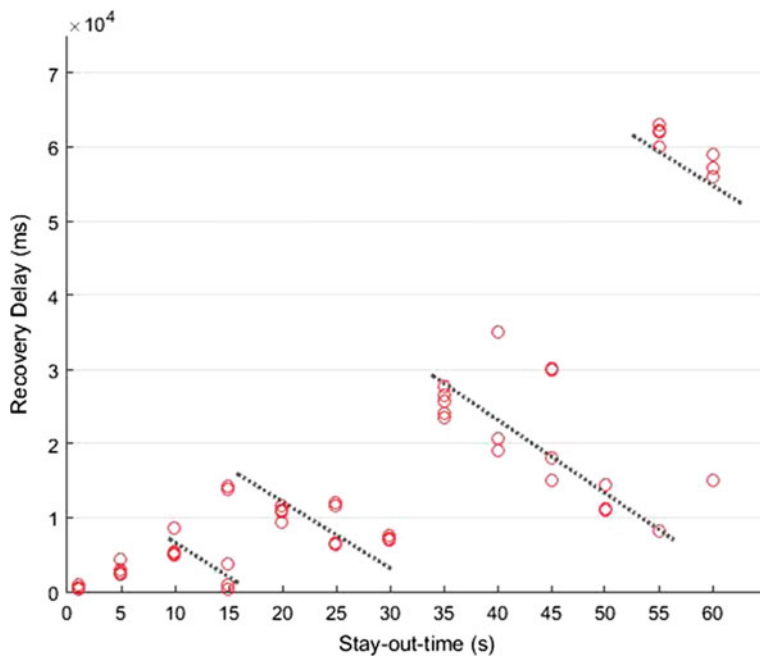


Fig. 12 Recovery delay in SameRouteReturning w/o disruption

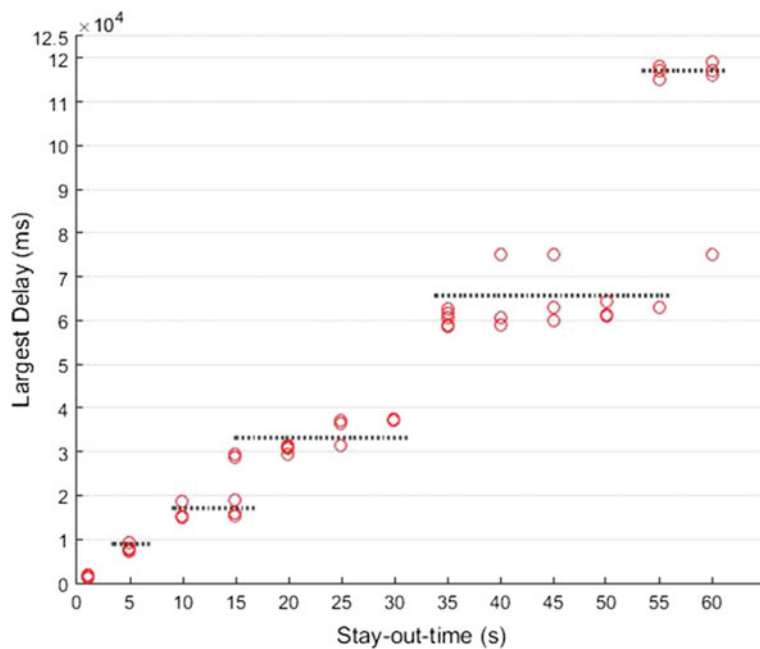


Fig. 13 Largest delay in SameRouteReturning w/o disruption

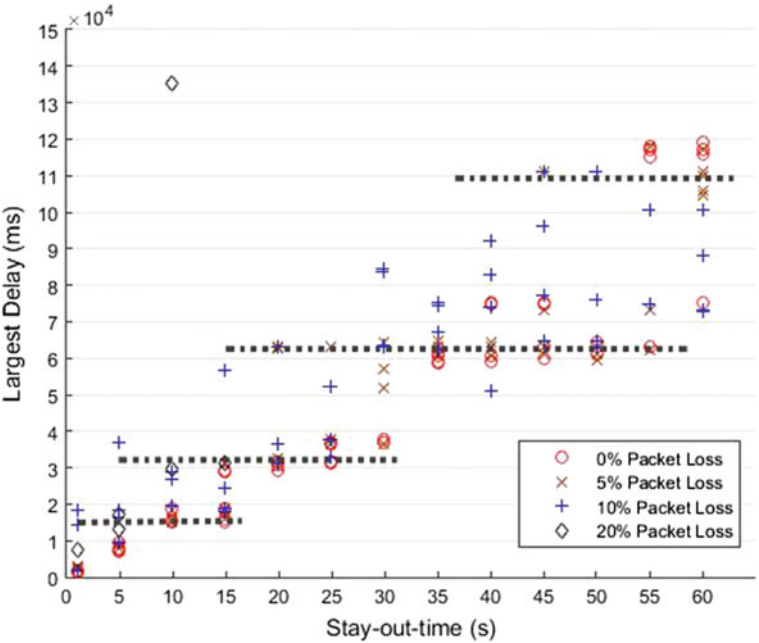


Fig. 14 Largest delay in SameRouteReturning w/disruption

time, it has to wait until the time is reached. Thus, earlier returning nodes would have the same retransmission time. For example, a node leaves for 40 and 50 s will both retransmit its packet at 68000 ms, leading to the same delay level in the figure.

Once disruption is taken into consideration (Sc-2-II, Sc-2-III, and Sc-2-IV defined in Sect. 5), the packet loss will increase the *Largest Delay* of a target *stay-out-time* to a higher level than the scenario without disruption, due to the failure of packet transmission. Figure 14 compares the *Largest Delay* by *Stay-out-time* in the *Same Route Returning* pattern across different levels of disruption. The figure shows that a higher disruption level like 10 % packet loss rate can increase the largest delay from 17000 to 32000 ms or even 65000 ms level for the same *stay-out-time* (15 s). Additionally, there are a significant number of data points that fall between two TCP retransmission schedule delay levels, due to the additional retransmission because of dropped packets. This increase of the largest delay by disruption also lowers the tolerance to the out-of-range time. Figure 15 compares the transaction failures in *Same Route Returning* among different levels of disruptions. For example, when the leaving node stays out for 10 s, there is 1 failure under 5 % packet loss rate scenario, 2 under 10 % scenario, and 3 under 20 % scenario, respectively. During the evaluation, we also find that, under the disruption-free scenario, the failure starts to occur at 66 s of stay-out-time.

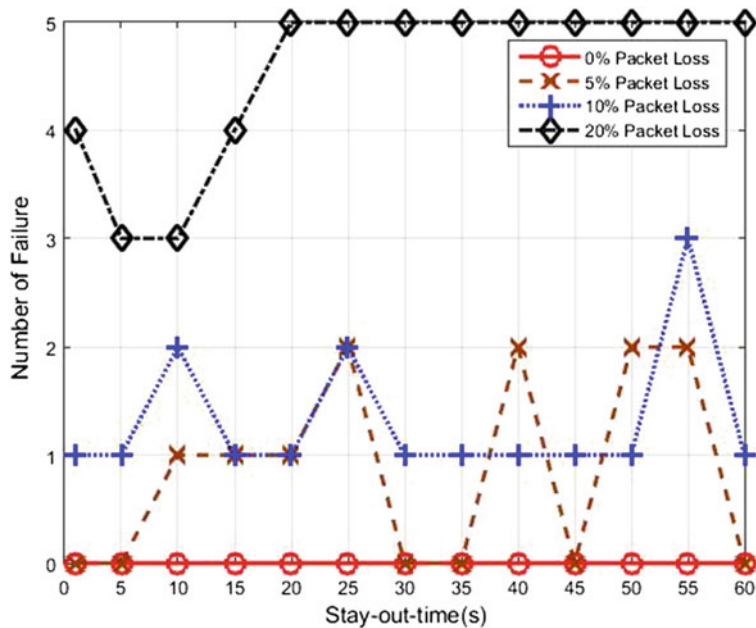


Fig. 15 Transaction failure in SameRouteReturning w/disruption

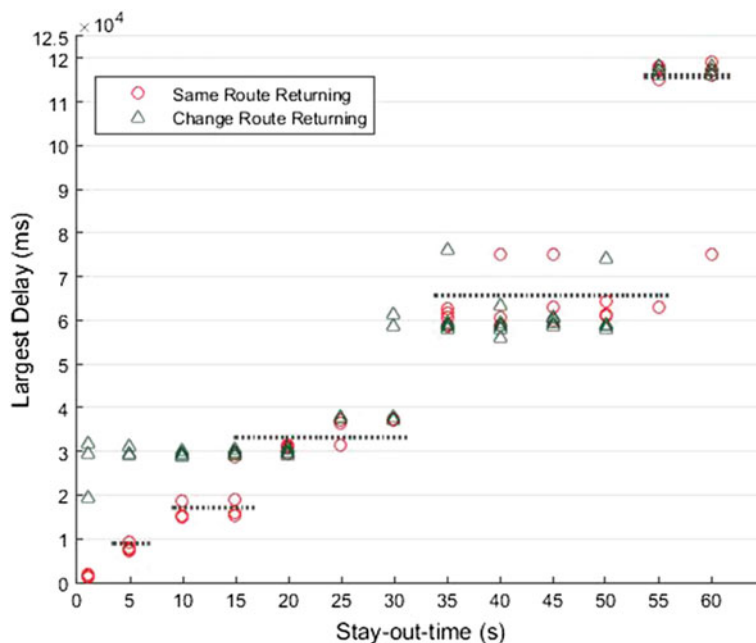


Fig. 16 Largest delay in Same versus ChangeRouteReturning w/o disruption

In the *Change Route Returning* Scenario without disruption (Sc-3-I defined in Sect. 5), the pattern is similar to the *Same Route Returning* with under the same conditions. In Fig. 16, we compare the *Largest Delay* by *stay-out-time* between *Same Route Returning* and *Change Route Returning*. It can be observed that when *stay-out-time* is above 15 s, the two scenarios follows the same pattern. Nonetheless, when the *stay-out-time* is under 15 s, the *Largest Delay* is a constant of 30 s for *ChangeRouteReturning*. This contrasts the *Same Route Returning*, where there are *Largest Delay* minima at 10 and 20 s, for *stay-out-times* of 5 and 10–15 s respectively. The distinction can be explained by the time necessary to correct the routing table for the *Change Route Returning* that does not exist for the *Same Route Returning*.

When disruption is involved (Sc-3-II, Sc-3-III, and Sc-3-IV defined in Sect. 5), *Change Route Returning* follows the same pattern as in *Same Route Returning* scenarios. Figures 17 and 18 compare the largest delay and transaction failure in different levels under different levels of disruption. From these figures, the result of emulation in Scenario Groups 2 and 3 indicates how the mobility patten of the leaving node will affect the performance of the NETCONF traffic in a MANET topology. The NETCONF requests in a MANET environment rely on the TCP retransmission strategy. The delay caused by the change of route in a MANET is determined by the TCP retransmission scheduling. Requests can be successfully sent, received, and executed as long as the route can be reconstructed before the last retransmission try. In a disruption-free environment, the transaction can tolerate 60 s for a leaving node

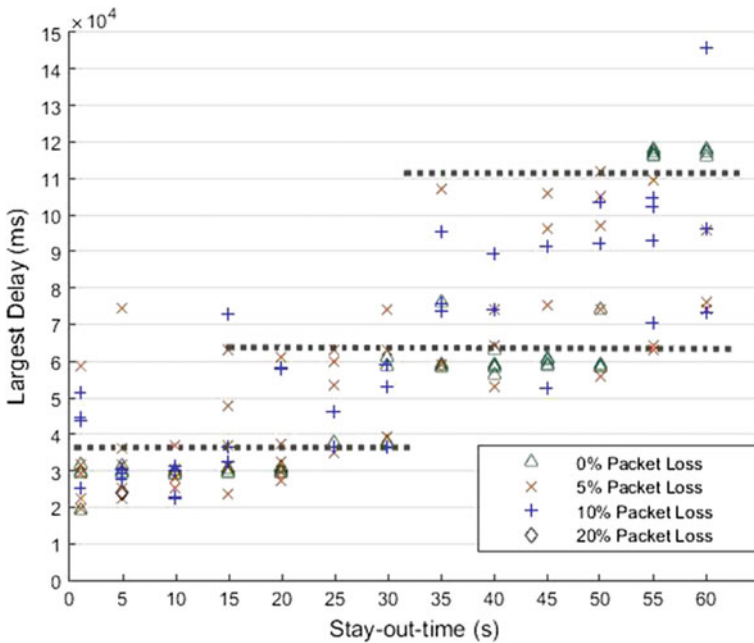


Fig. 17 Largest delay in ChangeRouteReturning w/disruption

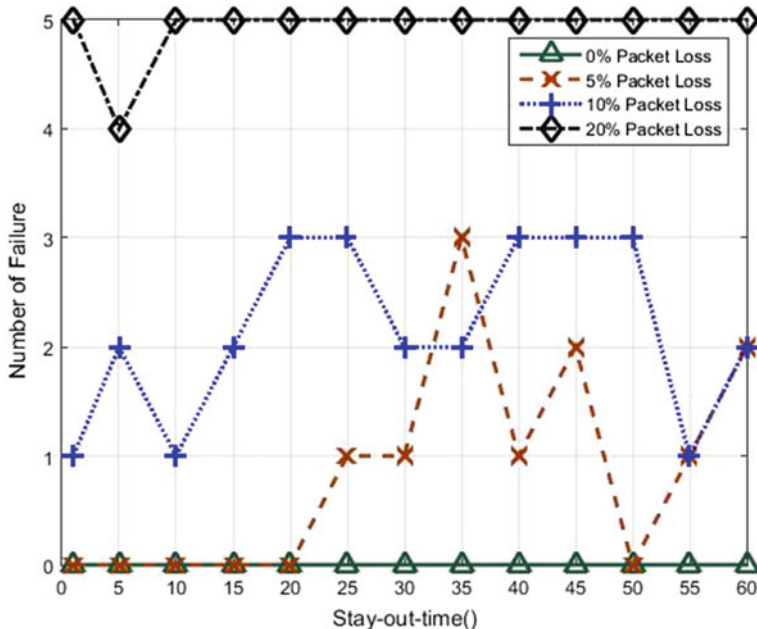


Fig. 18 Transaction failure in ChangeRouteReturning w/disruption

to stay out of range. Finally, there is no significant difference whether the original route is reconstructed or changed once the stay-out-time is greater than 15 s.

Disruption: Based on the emulation scenarios we have designed, the impact of disruption is compared among the 4 scenarios. Again, in Figs. 9, 14, and 17, the largest delay under 4° of disruption are compared in *Fixed Connection*, *Same Route Returning*, and *Change Route Returning* scenarios, and Figs. 10, 15, and 18 compare the request failure. As we can observe from each figure, a higher packet loss rate always results in a higher delay and the number of failures. This is because in a disruptive environment, the packet loss occurs during the transmission between two adjacent nodes, leading to the retransmission of packets. A higher packet loss rate can lead to more retransmissions, which further increases the delay and overhead, and reduces the capability of the NETCONF request to tolerate the *stay-out-time* of the leaving nodes.

5 Final Remarks

In our investigation, we leveraged the network emulation tool, CORE and carried out the quantitative evaluation of NETCONF in a MANET. We developed a generic framework that considers the key characteristics of MANET (distance, mobility, and

disruption) and designed scenarios to perform the emulation study. Our experimental data show how NETCONF performance was affected by individual characteristics, and the results can serve as a guideline for deploying NETCONF in MANET. The 3-dimensional framework that we designed to create MANET emulation scenarios can be applied not only to evaluate NETCONF, but to other protocols that are applicable to the MANET topology.

References

1. Enns, R., Bjorklund, M., Schoenwaelder, J., & Bierman, A. (2011). Network configuration protocol (netconf). Internet Engineering Task Force, RFC 6241.
2. Harrington, D., Presuhn, R., & Wijnen, B. (2002). An architecture for describing simple network management protocol (snmp) management frameworks. Internet Engineering Task Force, RFC 3411.
3. Imran, M., Said, A., & Hasbullah, H. (2010). An overview of mobile ad hoc networks: Applications and challenges. In *Proceedings of 2010 International Symposium Information Technology (ITSim)*.
4. Network Simulator V3 (NS-3). <http://www.nsnam.org>.
5. OMNET++. <http://www.omnetpp.org>.
6. Bellavista, P., Cardone, G., Corradi, A., & Foschini, L. (2013). Convergence of manet and wsn in iot urban scenarios. *IEEE Sensor Journal*, 13(10), 3558–3567.
7. Hoebeke, J., Moerman, I., Dhoeft, B., & Demeester, P. (2004). An overview of mobile ad hoc networks: Applications and challenges. *Journal of the Communications Network (JCN)*, 3(3), 60–66.
8. Patel, D. N., Patel, S. B., Kothadiya, H. R., Jethwa, P. D., & Jhaverii, R. H. (2014). A survey of reactive routing protocols in manet. In *Proceedings of 2014 IEEE International Conference on Information Communication and Embedded Systems (ICICES)*.
9. Chawda, K., Gorana, D. (2015). A survey of energy efficient routing protocol in manet. In *Proceedings of 2nd International Conference on Electronics and Communication Systems (ICECS)*.
10. Yu, Y., Ni, L., & Zheng, Y. (2011). Survey of qos multicast routing protocols in manets. In *Proceedings of 2011 International Conference on Computer Science and Service System (CSSS)*.
11. Alani, M. M. (2014). Manet security: A survey. In *Proceedings of 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*.
12. Goncalves, P. (2009). An evaluation of network management protocols. In *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*.
13. Hedstrom, B., Watwe, A., & Sakthidharan, S. (2011). <http://morse.colorado.edu/~tlen5710/11s>.
14. Slabicki, M., & Grochla, K. (2014). Performance evaluation of snmp, netconf and cwmp management protocols in wireless network. In *Proceedings of the 4th International Conference on Electronics, Communications and Networks*.
15. Herberg, U., Cole, R. G., & Yi, J. (2011). Performance analysis of snmp in olsrv2-routed manets. In *Proceedings of the 7th International Conference on Network and Services Management, October 24–28*.
16. Kuthethoor, G. (2008). Performance improvements to netconf for airborne tactical networks. In *Proceedings of IEEE International Conference on Military Communication (MILCOM)*.
17. Bjorklund, M. (2010). Yang—a data modeling language for the network configuration protocol (netconf). Internet Engineering Task Force, RFC 6020.
18. NETCONFCENTRAL. <https://www.netconfcentral.org>.
19. Common Open Research Emulator. <http://www.nrl.navy.mil/itd/ncs/products/core>.

20. OpenYuma. <https://github.com/OpenClovis/OpenYuma>.
21. Herberg, U., Clausen, T., Jacquet, P., & Dearlove, C. (2014). The optimized link state routing protocol version 2. Internet Engineering Task Force, RFC 7181.
22. OLSRD2. <http://www.olsr.org>.
23. OpenSSH. <http://www.openssh.com>.

Software Engineering Research, Management and
Applications

Lee, R. (Ed.)

2016, XIII, 199 p. 104 illus., 66 illus. in color., Hardcover

ISBN: 978-3-319-33902-3