Supporting Military Communications with Named Data Networking: An Emulation Analysis

Basil Etefia
The Aerospace Corporation
P.O. Box 92957
Los Angeles, CA 90009
basil.u.etefia@aero.org

Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
gerla@cs.ucla.edu

Lixia Zhang
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
lixia@cs.ucla.edu

***Abstract*—Named Data Networking is a new concept that takes a fundamentally different approach to what is found in Internet Protocol (IP) network systems. IP-type networks name locations (in the form of IP addresses) to route messages to destinations when sending data. Named Data Networks (NDNs), on the other hand, are primarily concerned about the data itself and not necessarily about the locations where the data is found. The forwarding decisions in an NDN are based on the actual data being sent/requested and not the presumed locations of the data content. Instead of naming network locations to facilitate message delivery (IP), NDNs name the actual data. This new approach allows the nodes of NDNs to more easily take advantage of the broadcast capabilities found in many of today's networking devices while also avoiding many of the problems and inefficiencies that exist within IP networks.**

Because many modern military communication systems rely on IP mechanisms to provide networking functionality, it is worthwhile to investigate the benefits NDN might be able to provide these systems. Specific design features of the NDN architecture provide an efficient alternative to IP-based systems, particularly those systems operating within challenged environments. Thus, this paper presents an emulation analysis of NDN-aided military communication networks. We also briefly describe the NDN architecture and how it can be incorporated into various military communication networks, such as the Navy's Automated Digital Networking System (ADNS) and the Army's Warfighter Information Network-Tactical (WIN-T). Preliminary results show tremendous performance gains when the NDN architecture is applied to these military communication networks.

I. Introduction

This paper presents a performance analysis of Named Data Networks (NDNs) when used in the challenged and dynamic environments associated with military communication systems. These networks take a fundamentally different approach to what is found in most modern network systems. The Internet Protocol (IP) suite, perhaps the most commonly used networking architecture today, uses names (in the form of IP addresses) to route messages to destinations when requesting and sending data. NDNs, on the other hand, are primarily concerned about the data itself and not necessarily about the locations where the data is found. The forwarding decisions in an NDN are based on the actual data being sent/requested and not the presumed locations of the data content. Instead of naming network locations to facilitate message delivery (like IP), NDNs name the actual data. Therefore, NDNs take a fundamentally distinct approach in that they eliminate the host model found in IP networks. Because NDNs do not use a host model, their architectural design makes them very well suited to deal with link intermittence and mobility, which are characteristics most often found in the challenged environments of military communication networks. It is the goal of this paper to investigate the benefits NDNs can provide to tactical, military communication systems.

II. Background

In an NDN, data is transmitted only in response to an interest message [1]. These interest messages specify the name of the content the source node is interested in and not necessarily the location of the node storing/producing this data. This is the fundamental difference from traditional IP networks—IP networks use source and destination addresses to forward and route data. NDNs use strictly the content names for data forwarding. An NDN node consists of three core components, a content store, pending interest table (PIT), and the forwarding information base (FIB). A diagram illustrating how these components are used is depicted in Figure 1.

When an interest packet arrives at an NDN node, a longest-match lookup is done on the name of the content. The index structure used for lookup in an NDN is ordered so that a content store match will be preferred over a PIT match, which will be preferred over a FIB match. Thus, if there is already a data packet in the content store that matches the received interest packet, this data packet will be sent out on the interface the interest packet arrived on and the received interest packet will be discarded (since it was satisfied). Otherwise, if there is an exact-match PIT entry, the interest packet's arrival interface will be added to the PIT entry's "Requesting Interfaces" list and the interest packet will be discarded. An interest packet for this data has already been sent upstream. Thus, when the data packet it solicits arrives, all that needs to be done is to send a copy of that packet out on the interface that the new interest packet arrived on. If there is a matching FIB entry, then the interest packet needs to be forwarded upstream towards the data. A new PIT entry is created from this interest packet and its arrival interface. If there is no match for the interest packet it is discarded (this node does not have any matching data and does not know how to find any).

978-1-4673-3/12/$31.00 ©2013 IEEE

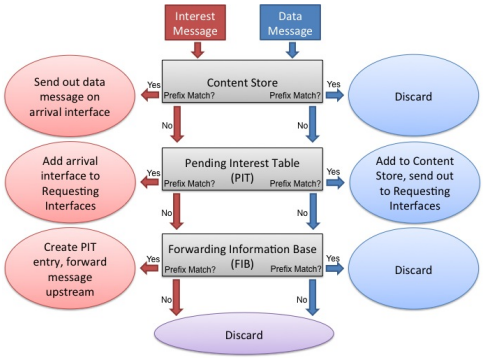

Fig. 1. NDN Node Processing Diagram

Data packet processing is very simple in an NDN. Data packets are not routed but simply follow the chain of PIT entries back to the original requester(s). A longest-match lookup of a data packet's content name is done upon packet arrival. A content store match means the data is a duplicate so it is discarded. A FIB match means there are no matching PIT entries so the data is unsolicited and it is discarded. A PIT match (there may be more than one) means the data was solicited by an interest(s) sent by this node. The data is added to the content store and a copy is sent out to each of the "Requesting Interfaces." Using this simple messaging scheme, content permeates these networks both quickly and efficiently.

Because NDNs do not use a host model, their architectural design makes them very well suited to deal with dynamics and mobility. The multipoint nature of data retrieval in these networks, combined with the broadcast capabilities of today's network devices, provides flexibility to maintain communication in highly dynamic environments. Also, using a cache, a mobile NDN node may serve as the network medium between disconnected areas, or provide delayed connectivity over intermittent links. These points illustrate why NDNs are well suited to handle mobility. Basically, when a user interested in specific data moves to a new location, it can simply rebroadcast its interest. Using the NDN architecture this interest message will propagate upwards and eventually find the desired content. This content will then follow the same path where the interest message came up.

Regarding security, all NDN content is authenticated with digital signatures, and private content is protected with encryption. NDN authenticates the binding between names and content. This provides both authentication and message integrity. The signature in each NDN data packet is over the name, the content, and a small amount of supporting data useful in signature verification. This allows publishers to securely bind arbitrary names to content. NDNs' signed bindings between names and content act in essence to certify that content. When that name refers to an individual or organization, and that content is a public key, the result is essentially a digital certificate. This allows NDN to easily support traditional mechanisms for establishing trust in keys. Similar to IP-based network hosts, application-level NDN consumers must solve traditional key management problems, such as associating public keys with individuals and organizations. Also, NDN content privacy is provided by encryption. NDNs, however, do not require trusted servers or directories to enforce access control policies. No matter who stumbles across private content, only authorized users are able to decrypt it. Encryption of content is completely transparent to the network in an NDN; it is all just named binary data (though efficient routing and data sequencing may require that some name components remain in the clear).

Fig. 2. ADNS Notional Topology

To facilitate interoperability with existing networks, NDN messages can be encapsulated by IP packets, as described in the emulation experiments of the next section. This allows networks to take advantage of the NDN framework with minimal modifications to existing network components. NDNs are sufficiently compatible with IP and they can be deployed incrementally, using current IP infrastructures.

III. Methodology

Previously in [2], we reviewed the design of the NDN architecture and discussed how these networks might fare in certain military communication scenarios. We studied two different military communication networks—the Army's Warfighter Information Network-Tactical (WIN-T) and the Navy's Automated Digital Networking System (ADNS)—and we concluded that NDNs can provide an efficient alternative to these current IP-based architectures. This paper presents a follow-on emulation analysis to that work. For these experiments, we used the University of Utah's Emulab test bed [3]. Emulab is a collection of servers interconnected via virtual local area networks (VLANs) to create an ad hoc emulation test bed. All nodes used in these experiments were 64-bit Intel Quad Core Xeon machines running Ubuntu 11.04. Palo Alto Research Center's (PARC) open-source NDN package CCNx [4] was used to implement the NDN functionality. All CCNx instances were run over the user datagram protocol (UDP). Furthermore, we used Perl scripts to automate experiments and configurations.

A. ADNS Emulation Configuration

In ADNS, shipboard users connect to land-based users through shore sites. These connections are generally made up of two separate legs—the ship-to-shore leg (generally through SATCOM links) and/or the shore-to-host leg (generally across terrestrial wire-line links). A notional topology for ADNS is displayed in Figure 2. Because the majority of this traffic travels through the shore site for processing, routing, and/or

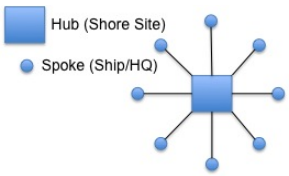

Fig. 3. Hub-Spoke Topology

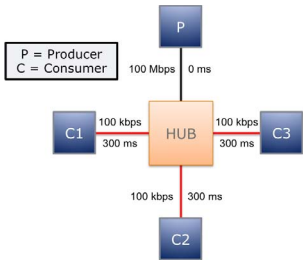

Fig. 4. ADNS Emulation Topology

forwarding, this model can be alternately viewed as a hub-spoke network, which is depicted in Figure 3. NDNs can easily take advantage of this model by simply caching data produced by the ships or land-based users at the shore (hub), allowing these sites to satisfy these interests instead of traveling all the way to the land-based or shipboard producers. Doing so conserves bandwidth by eliminating the overhead of end-to-end connections and also clipping traffic flows at the shore instead of at nominal endpoints.[1]

In order to evaluate NDN when used within ADNS, we used the hub-spoke network topology shown in Figure 3 to build the ADNS emulation topology shown in Figure 4. To emulate the satellite links between the consumers and the hub (shore site), Dummynet [5] was used to throttle the link data rates to 100 kbps and also to incorporate a one-way packet delay of 300 ms. The rate control and packet delay implementation was automated via Dummynet by the Emulab network test bed software. The link between the producer and the hub (shore site) was configured to have a data rate of 100 Mbps and a packet delay of 0 ms. This was meant to emulate the landline between the shore site and the base headquarters. Also, to emulate dynamic, mobile environments, uniformly random packet losses were added on the links between the consumers and the hub (the red links of Figure 4). These packet loss percentages were varied from 0%, 1%, 2%, 3%, and 4% in an effort to evaluate how these networks perform in challenged environments. This packet loss functionality was implemented using netEM [6].

B. WIN-T Emulation Configuration

In WIN-T, users in the field connect to other users through a hierarchy of vehicles and network equipment. A notional topology of this is depicted in Figure 5. NDNs can also take advantage of this hierarchy, and a generalized model of which is depicted in Figure 6. NDNs allow each of these network gateways to act as caches, temporarily storing the data produced by the content providers of this network. Doing so allows these network gateways to satisfy requests instead of requiring traffic flows to travel to each end host (or HQ), which again conserves network bandwidth by eliminating the overhead of end-to-end connections and also clipping traffic flows at the gateways instead of at nominal endpoints.

In order to evaluate NDN when used within WIN-T, we used the hierarchical network topology shown in Figure 6 to build the WIN-T emulation topology shown in Figure 7. The satellite links and the packet losses were emulated in the same way as described previously in the ADNS emulation description (red links have packet losses, satellite links have data rates of 100 kbps and one-way delays of 300 ms). The links between the consumers and their consumer routers, the producer and its producer router, and also CR2 to CR3 were configured to have a data rate of 100 Mbps, a packet delay of 0 ms, and random packet losses in an effort to emulate line-of-sight links.

C. WIN-T Mobility Emulation Configuration

In addition to the WIN-T emulation above, we completed a similar emulation experiment that includes node mobility. We used an emulation topology similar to that shown in Figure 7, except we removed the third consumer and added an extra link from the second consumer to the third consumer router. This was done to be able to emulate a mobile consumer. With this configuration, the second consumer can now be connected to the second consumer router or the third consumer router depending on its location. This modified emulation topology is shown in Figure 8.

The satellite links and the packet losses were emulated in the same way as the WIN-T emulation. To emulate mobility, we used Perl scripts, cron jobs and iptables (Linux firewall suite) to control the time and duration of node connectivity. At 5-minute intervals, iptables drops all outgoing traffic on one of C2's interfaces. C2's outage lasts 5 seconds. Walking through this process, before the experiment begins (time 0^-) both of C2's interfaces are up. At the beginning of the experiment (time 0), C2's eth1 interface is up while C2's eth2 interface is down. At time 300 seconds (5 minutes), both of C2's interfaces are down. Five seconds later (time 305 seconds), C2's eth2

Fig. 5. WIN-T Notional Topology

[1]Also note that, if all of the ships requesting this data are within the same beam or utilizing the same receive-only system (GBS, TV-DTS), the shore site can simultaneously send (i.e., broadcast) the data content to all of these users at the same time, conserving even more bandwidth.

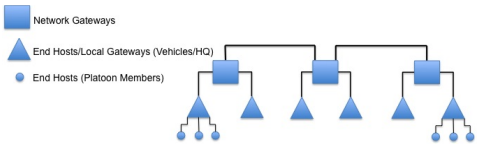

Fig. 6. Hierarchical Network Topology

interface is up while C2's eth1 interface is down. This process continues throughout the duration of the experiment, with link up/down events alternating on C2's interfaces.

D. Emulation Experiment

To assess NDN performance, two separate experiments were conducted on each of these topologies:

(1) NDN Experiment: all nodes of each experiment (producer, hub, producer router, etc.) were running CCNx. Each consumer requested 3 MB of data (600 files sized at 5 kilobytes each) from the producer using NDN interest and data messages.[2] Specifically for the WIN-T mobility emulation, to emulate the broadcast nature of these NDN consumers on a wired test bed, the interest message was sent on both of C2's interfaces when it requested data. This message was received by CR2 or CR3 accordingly, depending on which link of C2's was up during that time.[3]

(2) FTP Experiment: each consumer requests 3 MB of data (600 files sized at 5 kilobytes each) from the producer using FTP. FTP was chosen because it operates over TCP, which provides the message reliability required by these military, tactical networks. Specifically for the WIN-T mobility emulation, because of the mobility introduced by C2 in this network, we needed to implement some message routing functionality—the network needed a way to identify the new/current location of C2. Thus, we used Quagga [7] to implement OSPF in this network. For each OSPF instance, we used a hello timer of 10 seconds and a retransmit interval timer of 30 seconds.

IV. Results

The results from the ADNS, WIN-T, and WIN-T mobility emulation experiments are shown Figure 9. For each emulation topology, three different metrics were used to analyze performance:

(1) Average Delivery Delay: experimental delivery delay averaged across the 600 requested files.

(2) Maximum Delivery Delay: maximum delivery delay across the 600 requested files.

(3) Total Bandwidth Consumption: summation of the total bytes transmitted across each link of each experiment. This is used to determine how much total bandwidth the network is consuming.

[2]If a data message is not received 5 seconds after its corresponding interest message, the interest message is rebroadcasted.

[3]Note that no extra routing functionality needed to be added to this network. When a consumer moves to a new location in an NDN, it can simply just rebroadcast its interest message and still find the content of interest.

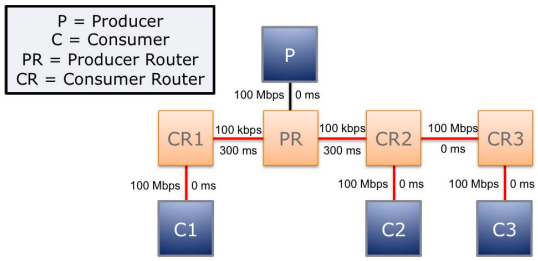

Fig. 7. WIN-T Emulation Topology

In each of these plots, the packet loss percentage is plotted along the x-axis and the corresponding performance metric is plotted along the y-axis.

A. ADNS Emulation

Figure 9 displays the average delivery delay recorded during the ADNS emulation experiments. As expected, as the packet loss percentage increases, the average delivery delay also increases. This is because when there is a packet loss, the respective interest (NDN) or request (FTP) message must be retransmitted to alert the producer to resend the corresponding data. This process results in an increase in delivery delay. Furthermore, as expected, the average delivery delay of all consumers of the NDN experiment is lower than the average delivery delay of all consumers of the FTP experiment. This is true for several reasons. For one, the NDN experiment caches messages at the hub node. This allows the interest messages of most of the consumers of the NDN experiments to have to travel only to the hub in order to find the requested data. The hub forwards the interest to the producer for the first interest requesting content that doesn't already exist within its content store. After the producer responds to the hub with the requested content, the hub then forwards this content to the consumer who originally requested that content. For each subsequent interest message that is requesting this same data, the hub simply satisfies these interests, using its cache to forward the requested content down to the respective consumer. Secondly, in the FTP experiment, each consumer forms a separate end-to-end TCP connection with the producer to obtain the requested content. Not only does this FTP approach not take advantage of caching at the hub, it also suffers because of its additional handshake and ACK mechanisms. FTP has its

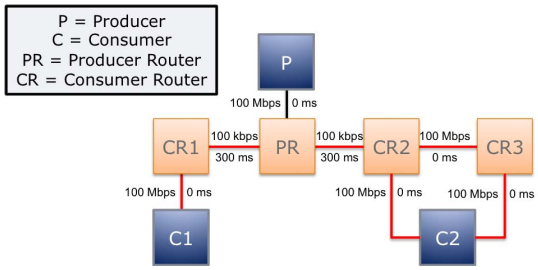

Fig. 8. WIN-T Mobility Emulation Topology

own handshaking mechanism on top of TCP.[4]

Figure 9 also displays the maximum delivery delay recorded during the ADNS emulation experiments. As expected, as the packet loss percentage increases, the maximum delivery delay also increases. This is again due to the fact that when there is a packet loss, the respective interest (NDN) or request (FTP) message must be retransmitted to alert the producer to resend the corresponding data. This process results in an increase in delivery delay. Furthermore, the maximum delivery delays of the FTP experiment consumers show some variation, which is due to the uniformly random packet losses. Despite this variability, all consumers of the NDN experiment remain well below the maximum delivery times of the consumers of the FTP experiment due primarily to NDN's caching and handshake-free functionality.

Lastly, Figure 9 also displays the total bandwidth consumption recorded during the ADNS emulation experiments. Again as expected, as the packet loss percentage increases for both the NDN and FTP experiments, the total bandwidth consumption increases. This happens because a packet loss usually results in the retransmission of data, which consumes additional bandwidth. Furthermore, in terms of total bandwidth consumption, the FTP experiment consumes far more bandwidth than the NDN experiment as the packet loss percentage increases. This is mainly due to all the control messaging required for FTP operation (ACKs and FTP/TCP handshaking) and the minimal control messaging required for NDN operation.

B. WIN-T Emulation

Figure 9 displays the average delivery delay recorded during the WIN-T emulation experiments. As the packet loss percentage increases, the average delivery delay also increases. The reasons for this are the same as described in the ADNS emulation experiments. Furthermore, as expected, the average delivery delay of all consumers of the NDN experiment is lower than the average delivery delay of all consumers of the FTP experiment. This is due to the fact that the NDN experiment caches messages at the consumer routers, which allows the interest messages of most of the consumers of the NDN experiments to not have to travel all the way to the producer in order to find the requested data. In the FTP experiments, each consumer forms a separate end-to-end TCP connection with the producer to obtain the requested content. This does not take advantage of any network caching and also incorporates additional handshake and ACK mechanisms. Also, C3 of the FTP experiment performs worse than its C1 and C2 counterparts in terms of average delivery delay, especially as the packet loss percentages are increased. This is a result of C3 having to compete with C2's traffic flow en route to the producer. Looking at Figure 7, C3 must traverse CR2 to get to the producer. CR2 also happens to be C2's shortest route to the producer. Because C3 has to compete with C2's traffic flow and also traverse an extra lossy link (CR2-CR3), its average delivery delay performance suffers. For this same topology, the NDN experiment does not suffer from the same drawback. C3 of the NDN experiment displays similar performance levels to that of C1 and C2. This is because CR2 can simply just cache content and satisfy the requests originating from C3.

Figure 9 also displays the maximum delivery delay recorded during the WIN-T emulation experiments. As the packet loss percentage increases, the maximum delivery delay also increases. As described earlier, this is due to the fact that when there is a packet loss, the respective interest (NDN) or request (FTP) message must be retransmitted to alert the producer to resend the corresponding data. This process results in an increase in delivery delay. Furthermore, the maximum delivery delays of the FTP experiment consumers show some variation, which is due to the uniformly random packet losses. Despite this variability, all consumers of the NDN experiment remain well below the maximum delivery times of the consumers of the FTP experiment due primarily to NDN's caching and handshake-free functionality. Also, in terms of maximum delivery delay, C3 of the FTP experiment performs worse than its C1 and C2 counterparts as a result of having a longer path to the producer and also sharing a link with C2's traffic. C3 in the NDN experiment, however, shows similar performance to its own C1 and C2 counterparts. This is mainly because of NDN's caching capabilities.

Lastly, Figure 9 also shows as the packet loss percentage increases for both the NDN and FTP WIN-T emulation experiments, the total bandwidth consumption increases. Furthermore, in terms of total bandwidth consumption, the FTP experiment consumes far more bandwidth than the NDN experiment does as the packet loss percentage increases. The reasons behind each of these outcomes are the same as those described previously in the ADNS emulation results section.

C. WIN-T Mobility Emulation

Figure 9 displays the average delivery delay recorded during the WIN-T mobility emulation experiments. As the packet loss percentage increases, the average delivery delay also increases. Also, the NDN approach performs better than FTP does. The reasons for these outcomes are the same as those described previously in the WIN-T emulation results section. Specifically in this WIN-T mobility emulation experiment, C2 of the FTP experiment performs worse than its C1 counterpart does in terms of average delivery delay. This is because C2 is now mobile. C2's links are dynamic, and OSPF is used here to determine message paths. C1, on the other hand, is not mobile, so it does not suffer the same outages as C2 does. For this same topology, however, the NDN experiment does not suffer

[4]There are actually three different phases happening within the FTP approach, the TCP handshake/teardown phase, the FTP handshake/teardown phase, and the actual data transmission. In all (and in the case with 0% packet loss so there are no retransmissions), there are about 20 different individual messages transmitted in each data flow in obtaining a single 5 KB file, and many of these messages are not pipelined (i.e. the client must receive a message from the server before it transmits the next handshake message, ACK, etc.). Each of these non-pipelined messages incur a round trip delay of 600 ms which begins to add into the total delivery delay, resulting in these relatively large delay values.

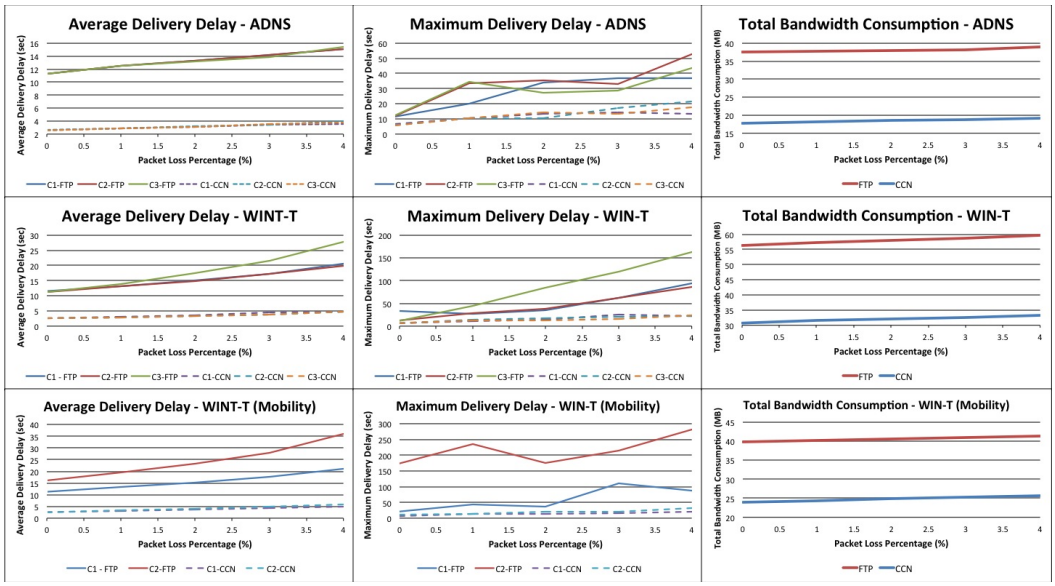

Fig. 9. Emulation Results

from this same drawback. C2 of the NDN experiment displays similar performance levels to that of C1. This is because CR2 (or PR) can simply just cache content and satisfy the requests originating from C2.

Figure 9 also displays the maximum delivery delay recorded during the WIN-T mobility emulation experiments. Similar to the WIN-T emulation results, as the packet loss percentage increases, the maximum delivery delay also increases. Furthermore, the maximum delivery delays of the FTP experiment consumers show some variation, which is due to the uniformly random packet losses. Despite this variability, all consumers of the NDN experiment remain well below the maximum delivery times of the consumers of the FTP experiment due primarily to NDN's caching and handshake-free functionality. Also, as described in the previous paragraph, C2 of the FTP experiment performs worse than its C1 counterpart does, primarily because C2 is now mobile. C2 in the NDN experiment, however, shows similar performance to that of its own C1 counterpart. This is primarily because of NDN's caching capabilities.

Lastly, Figure 9 displays the total bandwidth consumption recorded during the WIN-T mobility emulation experiments. As the packet loss percentage increases for both the NDN and FTP experiments, the total bandwidth consumption increases. This is because a packet loss usually results in the retransmission of data, which consumes additional bandwidth. Furthermore, in terms of total bandwidth consumption, the FTP experiment consumes far more bandwidth than the NDN experiment as the packet loss percentages increase. This is mainly due to all the control messaging required for FTP operation (ACKs and FTP/TCP handshaking) and the minimal control messaging required for NDN operation.

V. Conclusion

To conclude, this report presents a performance analysis of NDN when used in the challenged and dynamic environments associated with military communication systems. We provided a brief background about NDNs and also evaluated several NDN-aided military emulation topologies, comparing their performance to that of TCP-based FTP. For these topologies, emulations show that NDN provides better efficiency than TCP-Based FTP does in terms of average delivery delay, maximum delivery delay, and bandwidth consumption.

As a path forward, we would like to investigate the security aspects of these NDN systems. Smetters and Jacobson [8] have already begun to look deeper in this topic. Also, because storage is a key function of NDNs, we would like to investigate the amount of memory required by each NDN node for effective operation in these challenged environments. Given the military's communication network requirements, it is important that both of these areas are examined. However, given their ability to conserve bandwidth and minimize delivery delay, NDNs have the potential to greatly improve the performance of military/tactical communication systems.

References

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[2] B. Etefia and L. Zhang, "Named data networking for military communication systems," in *Aerospace Conference, 2012 IEEE*. IEEE, 2012, pp. 1–14.

[3] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 255–270, 2002.

[4] Project CCNx^{TM}. (2009, Sep.). [Online]. Available: http://www.ccnx.org

[5] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[6] S. Hemminger *et al.*, "Network emulation with netem," in *Linux Conf Au*, 2005, pp. 18–23.

[7] K. Ishiguro, T. Takada, Y. Ohara, A. Zinin, G. Natapov, and A. Mizutani, "Quagga routing suite," 2007.

[8] D. Smetters and V. Jacobson, "Securing network content," 2009.