# Evaluation of a Delay-Tolerant ICN Architecture

Hasnain Lakhani*, Tim McCarthy*, Minyoung Kim*, David E. Wilkins*, Samuel Wood†
*SRI International, Menlo Park, CA, USA †UCSC

***Abstract*—Simulation/emulation is key for early testing, assessment, and scalability evaluation of networking solutions for mobile ad-hoc networks (MANETs). If the solution is highly configurable – such as ENCODERS, SRI's delay-tolerant information-centric networking (ICN) solution – this type of evaluation is crucial. For effective modeling of information flows, the test framework needs to: (1) allow repeatable execution of scenarios with different patterns of network traffic, operating in different mobility and network-usage contexts, (2) provide a rich simulated environment that can model virtually any network topology and mobility, with high-fidelity device models, and (3) support flexible large-scale simulation, with the option of using virtual machines that execute the same code that would be used on an actual device. We describe our evaluation framework and the results of using it to develop and evaluate ENCODERS.**

## I. Introduction

We developed a high-fidelity, large-scale, repeatable evaluation framework that was crucial to our successful development of a delay-tolerant information-centric networking (ICN) solution for mobile ad-hoc networks (MANETs). This framework was key for early design testing, self assessment, and scalability evaluation. Because our solution is highly configurable, this framework also provided the means to conduct a systematic parameter-space exploration to find the best configurations. We begin by giving overviews of our ICN solution, ENCODERS, and our evaluation framework and methodology.

### A. ENCODERS

Typical MANETs are subject to dynamic network topologies, network partitions, energy constraints, and bandwidth limits. ENCODERS (Edge Networking with Content-Oriented Declarative Enhanced Routing and Storage) is SRI's solution for these challenges. It developed algorithms that operate at and exploit the higher-level of abstraction offered by an ICN architecture, and techniques for the storage and dissemination of content that is relevant in a given context, thereby further exploiting the richness of the available metadata and user/application interests. ENCODERS is open source [2].

We started with the Haggle open-source code base [3], which provides underlying functionality for neighbor discovery and basic protocols, among other things. We made major improvements in it, including improving performance in mobile networks and extensions for utility-based dissemination and cache management, network coding, and security.

Applications communicate in ENCODERS via data objects that contain both metadata and content. Metadata includes both a description of content and of the appplication's interests. This separation of metadata from content allows the selective distribution of content based on interest matching and is a key feature supporting the efficient use of bandwidth and low latency in ENCODERS (because content is much larger in size than the metadata that describes it).

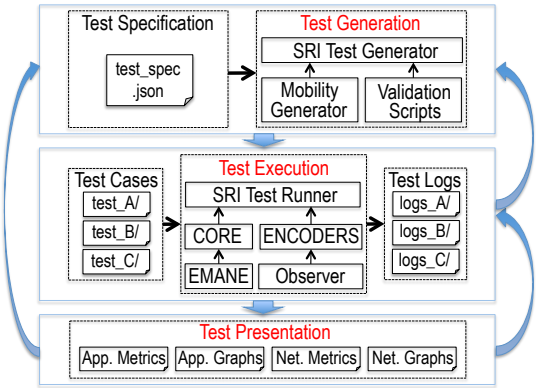


Fig. 1. Test Automation Framework: A concise specification of scenario parameters is used to automatically generate tests (using the SRI Test Generator module) with the mobility of the nodes automatically detailed. These tests, which can be rerun reproducibly using the SRI Test Runner module, are then automatically run and results generated for human analysis.

ENCODERS is a search-based data-dissemination framework. It efficiently controls dissemination, basing decisions on priorities that reflect information and mission needs, while using network resources wisely. As long as any path exists between two nodes, information will get through, even if not all segments of the path are up at the same time. Thus, it maintains delivery of critical information despite interruptions and intermittent connectivity. ENCODERS is layered on top of existing network protocols (e.g., UDP, TCP), making it network agnostic. Furthermore, its modular, open architecture facilitates extension, including supporting future interoperability with more specialized airborne protocols.

In [4], we describe (i) the ENCODERS architecture (initially named ICEMAN), which integrates multiple content-dissemination, utility-based caching, and transport mechanisms to provide a publish/subscribe API with attribute-based content naming, and (ii) content- and context-based policies to achieve efficient communication at the edge. Our design emphasizes compositionality. Without architectural changes, our system supports any combination of the caching, transport, and dissemination mechanisms.

### B. Evaluation Framework

MANETs typically include network disruption and reconnection; as well as limitations on bandwidth, range, transmission power, computing power, and memory. We present evaluation results showing that ENCODERS delivers the highest-priority (most relevant) information available in a timely manner, given constraints from the these limitations. We measure throughput, delivery fraction (in terms of relevant data objects), and latency as our performance metrics. To show that ENCODERS is a robust solution that can work continuously, we also present results for measures such as bandwidth utilization and CPU/memory usage.

| Composite Policy | Composition of Features |
|---|---|
| Phase 1 | Baseline |
| SP | Send Priority |
| SP+UR1 | SP, Utility-based Replication 1 |
| SP+UR2 | SP, Utility-based Replication 2 |
| SP+UC+UR1 | SP, Utility-based Caching, UR 1 |
| SP+UC+UR2 | SP, Utility-based Caching, UR 2 |
| SP+SC+UR1 | SP, Social-aware Caching, UR 1 |
| SP+SC+UR2 | SP, Social-aware Caching, UR 2 |

TABLE I
THE COMPOSITE POLICIES THAT WE EXPLORED.

Figure 1 shows the flow of our test-automation framework, which is composed of test (i) generation, (ii) execution, and (iii) analysis. Users can specify a json file containing multiple test specifications (mobility scenarios, application parameters, connectivity parameters, and so on). *Test Generator* creates multiple test cases from this specification, which are executed by *Test Runner*. Each test case contains scripts for mobility, network setup, logging, and ENCODERS configuration files. Each test case is fully self contained and reproducible. Test Runner executes these simulation/emulation scenarios using CORE/EMANE [5], [1] and keeps detailed logs from the test runs that are later analyzed. These logs contain delivery as well as node, network, and performance metrics.

Through the use of the test-automation framework, we were able to concisely specify large numbers of automated parameter-space exploration studies and regression tests that ran nightly on a set of Linux servers. Daily, we had detailed graphs to explore the effects of both code and configuration changes, which allowed us to rapidly improve ENCODERS.

Our first-year (Phase 1) evaluation of ENCODERS was previously reported [8] and enabled us to understand the performance characteristics of different policies. We found that dissemination, transport, and caching policies have significantly different performance characteristics (in terms of total data objects delivered and latency) and that a combination of these policies was necessary to achieve the best performance. More specifically, the best observed performance was achieved with combinations of hard- and soft-constraint utility-based caching policies that rank data according to network context.

We present evaluations for a variety of composite policies, which are summarized in Table I. Each policy is described as a set of features that are added onto the Phase 1 baseline policy, which is the best-performing combination of hard- and soft-constraint policies just mentioned. These features are described in detail elsewhere [4]. Briefly, *Send Priority* uses the priority specified in each data object. *Utility-based Replication* refers to a form of content replication: data objects are proactively pushed based on their utilities to the receiving nodes, for the sake of increasing the delivery ratio, or reducing the delivery latency. It may use resources sending data to nodes that have no interest in the data. *Utility-based Caching* is a generalized caching approach that frames the cache purging and replacement decisions as a utility-optimization problem. A caching policy defines content- and context-sensitive utility functions that vary in time and space. *Social-aware Caching* uses assumptions about the social hierarchy of the nodes.

| | A1 | A2 | A3 |
|---|---|---|---|
| Types | GPS; BioInfo1; BioInfo2 | Area Picture; Map; Audio | Frag Order; Op Order; Mission |
| Sizes (KB) | 1;5;10 | 250;500;1000 | 251;501;1001 |
| Pub. Dist. | uniform, mean =10s | exponential, mean = 60s | exponential, mean = 300s |
| Total Pub. | 1840 | 690 | 152 |
| Max Recv. | 115200 | 2070 | 1520 |
| Publishers | all | all | all |
| Subscribers | all | squad leaders | within squad |

TABLE II
THREE CLASSES OF APPLICATION-GENERATED TRAFFIC.

| A1 | A2 | A3 |
|---|---|---|
| RTTL of 10s | RTTL of 900s | RTTL of 400s |
| Replacement by creation time | No Replacement | No Replacement |
| No Replication | Utility-based Replication | No Replication |
| No Network Coding | Network Coding | Network Coding |

TABLE III
POLICIES FOR EACH TRAFFIC CLASS. RTTL (RELATIVE TIME-TO-LIVE) GIVES THE EXPIRATION DATE OF A DATA OBJECT.

## II. Scenarios

We used two scenarios for evaluation: (i) ground-based platoon-level operations and (ii) an airborne-ground-space network. In both cases, using our evaluation framework was key. Given the scenario definition, we first define mobility constraints on the nodes so that our Test Generator can automatically detail movement. We then use our automated test framework to conduct a parameter-space exploration to guide us in determining a scenario-specific set of parameter settings for further evaluation.

### A. Ground-based platoon-level operation

We modeled a 30-node tactical scenario with an explicit social hierarchy (three squads of 10 members each). To specify squads' movement, we use the Nomadic Community Mobility model in the BonnMotion scenario generator [6]. In this model, groups are performing a random walk around their reference points that follow a Random Waypoint Model. There is high network connectivity within a squad, but intra-squad connectivity is very limited. Occasionally squads pass near each other, providing brief periods of high connectivity.

Each node runs applications that generate different classes of traffic, which are defined in Table II. For each class, we generate content that supports situational awareness amd specify policies, as shown in Table III. RTTL gives the time (upon receiving from the neighbor) after which a data object can be discarded without delivering to applications, and replacement by creation time (an example use of our general replacement mechanism) allows newer versions of the same data to replace older versions.

A1 traffic models a blue-force tracking application that generates small-sized content that is frequent and continuous (e.g., GPS coordinates). Every node publishes content to everyone else. A2 traffic models an application that collects data in response to random events, such as taking a photo of a vehicle, map annotations, or audio recordings. Content is pushed to the squad leaders who share it with other leaders. A3 traffic models intra-squad communication (e.g., the squad leader pushes an operational order to the squad).

We evaluate ENCODERS for the variety of policies in Table I. Each content type is further subject to the specified policies,

| Traffic Class | Content Type | SP | UR 1 | UR 2 | UC | SC |
|---|---|---|---|---|---|---|
| A1 | GPS | Very Low | No | No | Low | Low |
| | BioInfo1 | Low | No | No | Low | Low |
| | BioInfo2 | Low | No | No | Low | Low |
| A2 | Area Pic. | Medium Low | High | Low | Medium | Medium |
| | Map | Medium | Medium | Medium | Medium | Medium |
| | Audio Rec. | Medium High | Low | High | Medium | Medium |
| A3 | Frag Or. | High | No | No | High | High |
| | Op Or. | Very High | No | No | High | High |
| | Mission | Very Very High | No | No | High | High |

TABLE IV
THE PRIORITIES AND UTILITIES OF EACH CLASS OF CONTENT. THE RIGHTMOST FIVE COLUMNS ARE THE FEATURES USED BY THE COMPOSITE POLICIES IN TABLE I.

| Content Type | Publisher | Subscriber | Size | Freq. |
|---|---|---|---|---|
| Fighter-track | All fighter pairs | Other pairs, all C2 | 7500bits | 10s |
| Full Air Pic. | All C2 | Ship, a C2, fighter pairs | 75000bits | 2s |
| Ground-track | Ground | All C2 | 7500bits | 2s |
| Radar Image | Sensing fighter pair | One C2, Ship, Ground | 1Mbits | 5s |
| Video Rec. | Sensing fighter pair | One C2, Ship, Ground | 7.8Mbits | 10m |

TABLE V
SUMMARY OF DATA EXCHANGE IN THE AIRBORNE SCENARIO.

using the priority and utility assignments in Table IV. We start with the best-performing Phase 1 policies [8], then augment them with new features, and evaluate performance.

### B. Airborne/ground network with satellite communication

In the scenario in Figure 2, two 4-ships of fighters perform a reconnaissance mission. Each 4-ship is supported by a C2 aircraft in a racetrack orbit which serves as a relay to one ship and a ground site. The transmissions from fighters are limited, because they do not want to be detected by their transmissions and may be jammed, so the fighter pairs do not communicate with each other. The links from C2 aircraft to fighters and the link between one C2 aircraft and Ship have 50% connectivity (10 minutes on and 10 minutes off). There is with no connectivity from fighters to C2 aircraft.

Space-based nodes (such as nano-sats) support communication from fighters. The initial model contains two nano-satellites, which are always connected. Our initial model optimistically assumes a continuous satellite presence in the network. This corresponds to having enough satellites to provide continuous coverage and an ability to handoff network responsibilities from one satellite to another, which ENCODERS could support. In future work, one could model more realistic constellations of nano-sats with hand-over as constellations pass overhead, and parameterize links to explore different devices and scenarios.

We modeled the data exchange for a mission where the fighters collect radar images and video recordings of the area of interest as shown in Table V. Table VI summarizes the policies we evaluated for each type of content.

## III. ENCODERS Evaluation Results

We first describe results from the ground scenario, which was tested on both Linux containers and Android devices (Nexus S). For Linux, we set a CPU limit for each virtual host to roughly match the CPU resources on the target phones. We model an IEEE 802.11a/b/g link in EMANE with an omnidirectional antenna gain of -5 dBi, a system noise factor of 4dB, and a freespace pathloss model. Using our evaluation framework, we conducted a parameter-space exploration

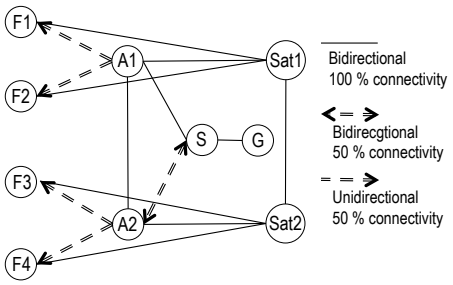


Fig. 2. Airborne/ground network scenario — F1-F4: fighter pairs, A1-A2: C2 aircraft (e.g., E-3 AWACS), S: surface ship, G: ground site, Sat: satellite.

| Content Type | Replacement | RTTL | SP | Network Coding |
|---|---|---|---|---|
| Tracks, Full Air Pic. | By creation time | 60s | High | Off |
| Radar Image | By creation time | No | Medium | On |
| Video Rec. | No | No | Low | On |

TABLE VI
POLICIES SPECIFIED FOR EACH TYPE OF CONTENT.

for varying policies and show the results here. Graphs are generated by our Test Automation Framework as shown in Figure 1. Next we show results from the airborne scenario. We use the RF-Pipe link model with a data rate of 10Mbps. The metrics across all our experiments are total data objects delivered for each traffic class, delivery latency distribution for each traffic class, total transmit/receive bytes, and total data-object delivered bytes.

### A. Robustness Results via Observables

Figure 3 shows observables for the purpose of evaluating the robustness of ENCODERS. These figures show observables used for monitoring the resource usage (e.g., bandwidth, CPU, memory) and network status (e.g., a connectivity measure based on the number of neighbors). These graphs all show that the quantity in question (bandwidth consumption, CPU, and so forth) level off as time elapses, and do not exhibit undesirable exponential properties, providing evidence of ongoing, robust execution of ENCODERS at each node. Other observables were also continuously monitored for self-assessment, including observables on information flow, such as content and interest. Disseminated data objects can be observed with distributed monitoring, and can be further analyzed to optimize use of network resources.

To understand what factors affect network performance, we observed the use of the cache. Figure 3(d) shows the ratio of cache usage over time when the cache size is limited to about 75% of the typical observed cache usage (when the cache is unlimited) in our scenario. We see that the cache is heavily utilized and close to capacity, an observation we will use later in this section in analyzing the importance of cache management to performance.

Evictions can be due to purging or replacement. Figure 3(g) shows a fairly linear growth over time with small spikes, indicating that ENCODERS is managing the cache smoothly, consistent with long-term stability. Figure 3(h) shows the number of data objects hard evicted (dropped immediately on receipt, before insertion into the cache). By comparing to Figure 3(e), we see that hard evictions spike when connectivity spikes. Figure 3(e) shows a sudden spike in connectivity before

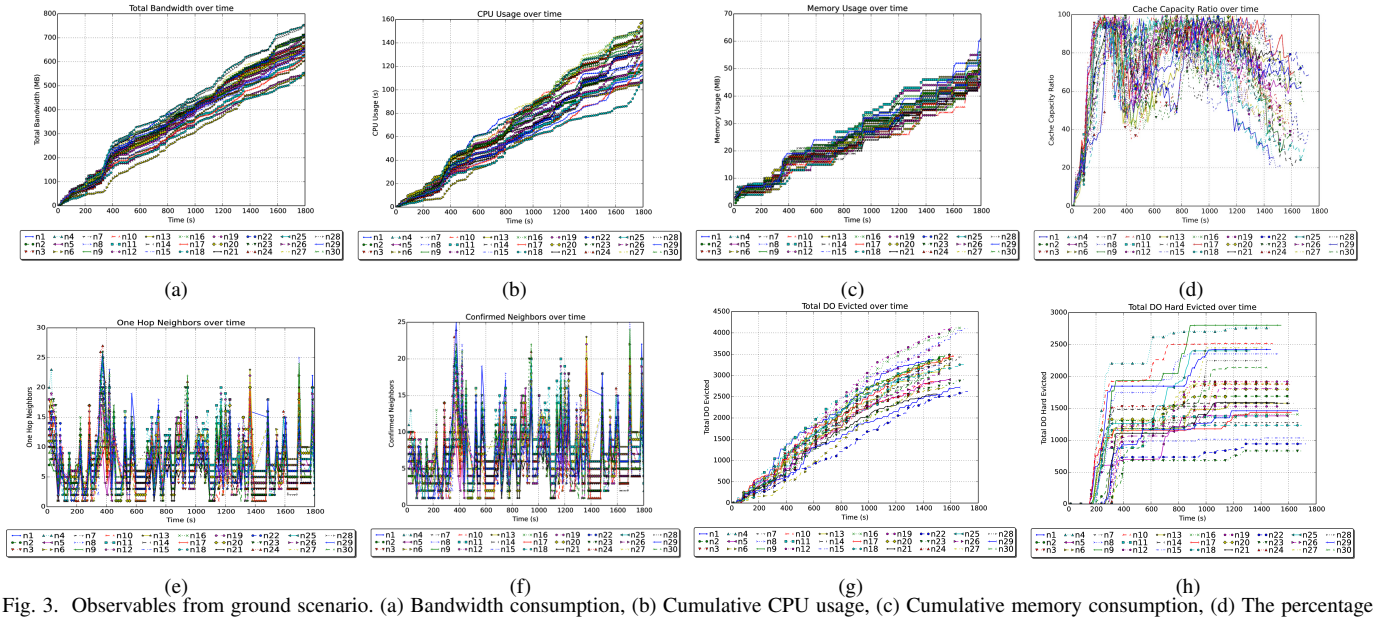


Fig. 3. Observables from ground scenario. (a) Bandwidth consumption, (b) Cumulative CPU usage, (c) Cumulative memory consumption, (d) The percentage of cache capacity used, (e) Number of physical neighbors at any given time, (f) Number of the logical neighbors at any given time, (g) Total cache evictions, (h) Hard evictions. X-axis is time in seconds. Y-axis is the value of the observable in question, and each colored line represents the value of the observable over time for one node in the network. The graphs do not exhibit undesirable exponential properties for resource usage.

t=400, when hard evictions also spiked. This happens because there are many cached items that are duplicated or outdated when a new group of neighbors is connected.

### B. Ground-based platoon-level operation

Figure 4 shows the data-object delivery in a 30-node ground-based operation. For the first three bars of each column (colored blue, green, and red), the y-axis (left side) is total bandwidth in MB. For the last two, the y-axis on the right side gives the delivery fraction. The fraction of total data objects delivered for A2 and A3 is depicted by the cyan bar in histograms, and the fraction for A1 is depicted in the rightmost bar of each group of 5 bars. Delivery fraction is the ratio of delivered data objects over the theoretical maximum number of data objects that can be received given a fully connected network without mobility and resource constraints.

The bandwidth data are as follows: total transmitted bytes (Tx) and total received bytes (Rx) includes all traffic that occurs (i.e., both content and overhead), while data-object received bytes (DRx) measures the portion of Rx that constitutes data objects in traffic classes A2 and A3 that are delivered to an application that is interested in them.

In Figure 4(a), the 2 rightmost bars in each group show improved performance over our baseline (the leftmost 5 bars labeled as Phase 1). Figure 4(b) presents the evaluation results on Android devices, which presents lower total usage of bandwidth for overhead for all policies, and the red and aqua bars in each group show improved performance for the higher priority A2 and A3 content for all policies.

We now compare the two graphs in Figure 4. In our baseline (Phase 1), we see the A1 data delivery (purple bar) is much higher for Androids than it was for Linux containers, while the DRx and A2+A3 values are much lower. This is because network coding [7] often blocks due to insufficient CPU on Android devices, resulting in all resources being devoted

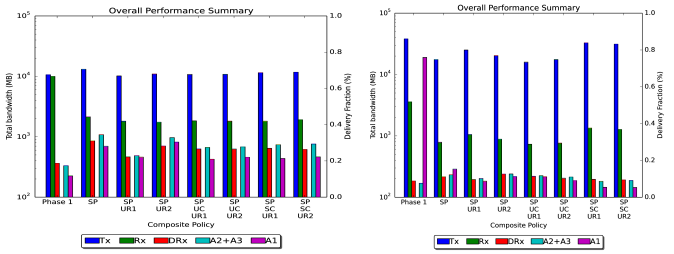


Fig. 4. Data object delivery; (a) Linux containers, (b) Android devices. The x-axis depicts results for different dissemination/caching policies (Table I).

to delivering the small A1 data objects. Our utility-based dissemination prioritizes A2 and A3 content, so most of the resources are spent on A2 and A3, with greatly improved throughput within the Android CPU limitations, as shown in the red and aqua bars of Figure 4(b). The A1 rates (purple bar) are low because outdated A1 data is not disseminated.

Figure 5 presents the delivery performance on Linux containers and Android devices. Data-object delivery latency is the amount of time that it takes the data object to be delivered to an interested subscriber. The x-axis is latency in seconds, and the y-axis is the total number of data objects that were received within that given latency. Each colored line represents a different composite policy from Table I. Figure 5(a)(c) is for latency from subscription (i.e., interests injected into ENCODERS), and Figure 5(b)(d) is for latency from publication (i.e., data objects injected into ENCODERS).

The different y-axis scale in Figure 5(a)(c) shows that 2-3 times more A2+A3 objects are delivered with Linux containers. The latency is lower with Androids (Figure 5(c)(d)) because most of the data objects never get delivered – the blue (Phase 1) plot is caused by the same phenomenon as the tall purple bar in Figure 4(b), namely insufficient CPU for network coding means nearly all delivered objects are A1 data.

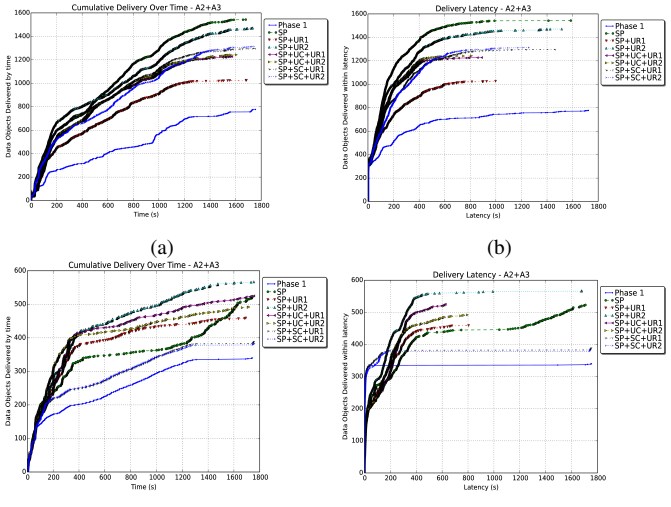


Fig. 5. Delivery performance: (a) and (c) the cumulative number of data objects delivered (over the entire scenario), (b) and (d) within a given latency. (a) and (b) are on Linux containers; (c) and (d) on Android devices. Each line represents a different policy combination (Table I).

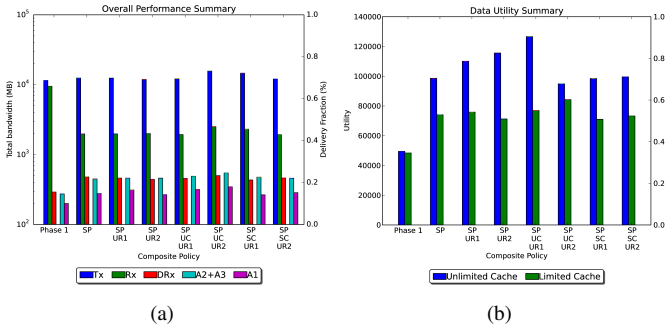


Fig. 6. Effects of limiting cache size with 30-node Linux containers (a) data object delivery with cache limited to 24MB, (b) utility-based view.

To understand what factors affect network performance, we experimented to assess the sensitivity of ENCODERS to the cache size at each node. We use Figure 4(a) as the baseline, which depicts performance with an unrestricted cache. In this case, we observed that cache usage went up to approximately 32MB at many nodes, well within the cache limitations of proposed devices. Figure 6(a) shows the effect of limiting the cache to about 75% of this typical cache usage. We see that the baseline results are not affected. Because of the poorer performance in Phase 1, the cache rarely filled up, so limiting it had no effect.

On the other hand, ENCODERS delivers more data, using 32MB of cache at many nodes. With a 24 MB cache limit, we see delivery of A2+A3 reduced by 20-50%, although still significantly improved over the baseline (Phase 1). This happens because data objects are dropped before they can be delivered. These results indicate that ENCODERS performance is quite sensitive to cache size if the cache size is not sufficient.

Figure 6(b) is another utility-based view of these results. In this analysis, we assign high/medium/low utilities to A3/A2/A1 content type, respectively. Such utilities might be assigned by the application producing them, taking into account the mission and the situation. Intuitively, a higher utility indicates more useful data. We see that the effects of limiting the cache size are significant, although performance is still better than baseline.

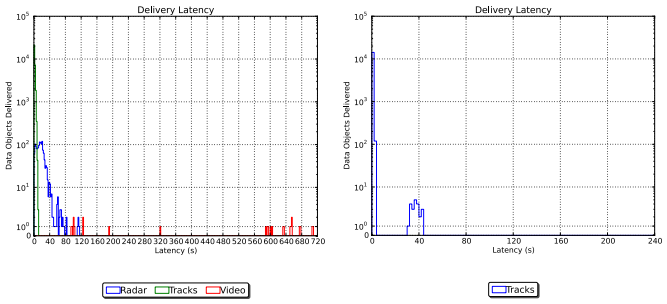


Fig. 7. Data-object delivery latency for different types of content: (a) with, and (b) without the satellite. Without the satellite, only tracks get delivered.

| | With Satellite | | | Without Satellite | | |
|---|---|---|---|---|---|---|
| Content | Min | Mean | Max | Min | Mean | Max |
| Tracks | 0.032 | 0.094 | 1.322 | 0.033 | 0.792 | 40.615 |
| Radar | 0.875 | 9.565 | 85.964 | n/a | n/a | n/a |
| Video | 39.834 | 368.543 | 686.220 | n/a | n/a | n/a |

TABLE VII
LATENCY (SECONDS) FOR AIRBORNE DATA OBJECT DELIVERY

## C. Airborne/ground network with satellite communication

We used our evaluation framework to evaluate performace in the airborne scenario, with and without the satellite nodes.

Table VII and Figure 7(a) show that, with satellites, 69% of tracks are delivered within 2 seconds and 100% within 12 seconds, while 66% of radar is delivered within 20 seconds. Videos take approximately ten minutes, a time that is likely high because of the transmission being interrupted (due to intermittent connectivity of the C2) while in progress. The graphs on the right side have a logarithmic y-axis, so the first bar represents over $10^4$ tracks. Without satellites, Table VII and Figure 7(b) show that only tracks get delivered, mostly within 2 seconds. Looking more closely, only 14,356 tracks are delivered versus 30,756 tracks with the satellite. The lower performance without satellites is due to disconnections (as nodes move) and the use of replacement policy, which will discard undelivered tracks when they are outdated. Radar and video are not delivered because fighters choose not to broadcast to non-space neighbors, to avoid detection.

Figure 8 shows data-object delivery for different types of content. In 8(a), videos are published every ten minutes by the fighters and received at a C2 node at regular intervals. Videos arrive at the ground and ship at approximately the same times because of the reliable link between them. About half the time, there is a significant delay between the video getting to the C2 node and subsequently to the ship, which is due to periods of disconnection while the C2 is out of range of the ship.

Radar data is received at regular rates at the C2 node in Figure 8(b). The maximum possible number of radar images received at both C2 nodes is twice that at the ground or ship (as there is only one of each). The radar images received at ground and the ship are significantly less than this maximum, because outdated images get replaced when lack of connectivity prevents their delivery. Delivery at the ground lags that of the ship as delivery to ground goes through the ship.

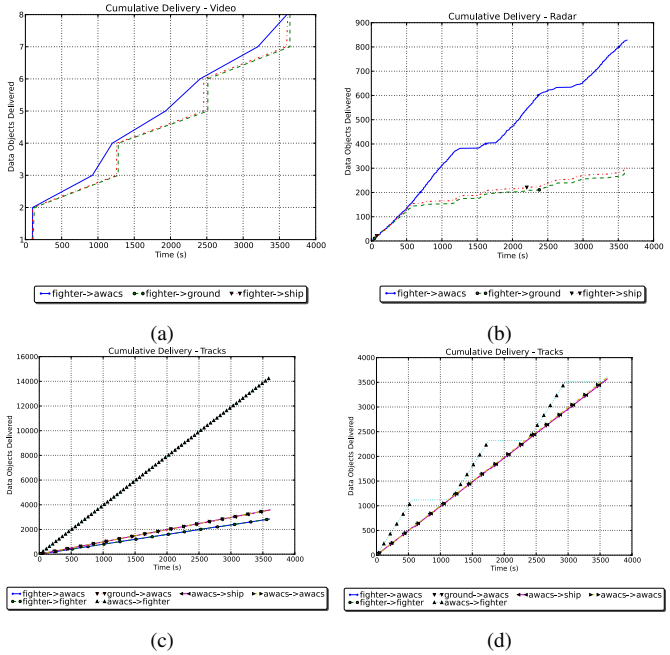


Fig. 8. The cumulative number of data objects by type received over time between each pair of publisher/subscriber. The subfigure shows (a) video, (b) radar, (c) tracks with the satellite, and (d) tracks without the satellite. Note the large differences in scale of the y-axis for each graph.

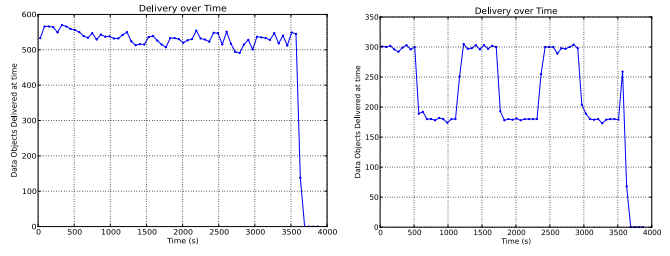


Fig. 9. The number of data objects delivered in the previous minute at time t, (a) with, and (b) without the satellite. Note the different y-axis scales: (a) shows 500+ data objects being consistently delivered, while (b) alternates between about 175 and 300, as the C2 node connection comes and goes.

The tracks are delivered at a regular rate as shown in Figure 8(c), because they are highest priority and travel along stable links. In 8(d), tracks from the ground to C2 go at a constant rate as one C2 is consistently connected to the ship and it can forward them on to the other C2 node. Tracks from the C2 nodes to the other C2 and ship go at a constant rate for the same reason (thus, lines for each of these overlay each other in the graph). Tracks from C2 nodes to the fighters go at a constant rate while they are connected but there are 10 minute periods where no tracks go through due to disconnection, producing the discontinuity seen in the green-diamond lines in the graph. There is no burst when C2 nodes come back in contact, because ENCODERS has replaced all outdated tracks with the latest information.

With satellites, the other nodes in the network gain, without the detection risk of the fighters communicating to C2 nodes, knowledge of the fighters location (mostly within 2 seconds), and of the content published by the fighters ($\frac{2}{3}$ of radar delivered within 20 seconds, videos take approximately ten minutes). Figure 9 shows the delivery rate of data objects over time with and without satellites. We see almost double the delivery, and consistent delivery, due to the satellites ensuring contact between nodes. ENCODERS is also able to deliver radar and video because of the satellite. Without the satellite, tracks from C2 nodes to fighters can only be transmitted when they are in contact (every ten minutes), producing the regular intervals of higher/lower delivery seen in Figure 9(b). There is a consistent delivery without satellites of tracks from ground to C2, as well as between C2 nodes and the ship, but no delivery of content published by the fighters.

## IV. Conclusion

We developed tools to automate a systematic parameter-space exploration for MANET scenarios. Our fully automatic test generation and execution framework can run simulations with a variety of network topology, mobility, and link models provided by CORE/EMANE. Our setup supports flexible simulation and also emulation where we use Linux containers to execute the same code that would be used in a testbed or a field demonstration. We run ENCODERS in resource-constrained Linux containers that approximate the performance of the target platform. Because of these laboratory tests, ENCODERS was robust and ready for the large number of tests that we ran on a 30-node Android phone testbed.

We modeled a new military scenario that included ground, maritime, airborne, and satellite nodes, to explore how widely applicable our system is beyond platoon-level ground operations. Our parameter-space exploration for a given scenario consists of identifying key parameters of the ENCODERS design, and running many tests to investigate the trade-offs inherent in different combinations of the values of those parameters. In this way, we generate scenario-specific guidance for setting the values for those parameters. We believe that the proposed methodology is highly usable for various scenarios and systems (both military and non-military).

**Acknowledgment:** This work was supported by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR Systems Center Pacific under Contract N66001-12-C-4051. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

## References

[1] EMANE. http://cs.itd.nrl.navy.mil/work/emane/.
[2] ENCODERS. http://encoders.csl.sri.com/.
[3] Haggle. http://www.haggleproject.org.
[4] ENCODERS software design description v.2.0. http://encoders.csl.sri.com/wp-content/uploads/2014/08/CBMEN-SRI-Design-Description-V2.0-Dist-A.pdf, 2014.
[5] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim. CORE: A real-time network emulator. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7, 2008.
[6] C. de Waal and M. Gerharz. Bonnmotion: A mobility scenario generation and analysis tool. *Communication Systems group, Institute of Computer Science IV, University of Bonn, Germany*, 2003.
[7] J. Joy, Y.-T. Yu, M. Gerla, M.-O. Stehr, S. Wood, and J. Mathewson. Network coding for content-based intermittently connected emergency networks. Mobicom Demo '13, 2013.
[8] S. Wood, J. Mathewson, J. Joy, M.-O. Stehr, M. Kim, A. Gehani, M. Gerla, H. Sadjadpour, and J. Garcia-Luna-Aceves. ICEMAN: A system for efficient, robust and secure situational awareness at the network edge. MILCOM '13, 2013.