ICEMAN: A System for Efficient, Robust and Secure Situational Awareness at the Network Edge

Samuel Wood*†, James Mathewson*†, Joshua Joy‡, Mark-Oliver Stehr¶,
Minyoung Kim¶, Ashish Gehani¶, Mario Gerla‡, Hamid Sadjadpour*†, J.J. Garcia-Luna-Aceves*
*UC Santa Cruz, †SUNS-tech, Inc., ‡UC Los Angeles, ¶SRI International
{sbwood, jlmathew, hamid, jj}@soe.ucsc.edu, {jjoy, gerla}@cs.ucla.edu, {stehr, mkim, gehani}@csl.sri.com

***Abstract*—Situational awareness applications in disaster response and tactical scenarios require efficient communication without a managed infrastructure. In principle, the performance, size, weight, and power of commercial off-the-shelf mobile phones and tablets are sufficient to support such applications, provided that efficient protocols and mechanisms are put in place for the efficient and secure sharing and storage of content among such devices. ICEMAN (Information CEntric Mobile Ad-hoc Networking) is a system that allows applications to request content objects by their attributes, and integrates its API with utility-based dissemination, caching, and network-coding mechanisms to deliver content. ICEMAN is implemented based on the Haggle architecture running in the Android operating system, and supports distributed situational-awareness applications operating in networks subject to severe disruption. Its functionality is described, and performance results of the ICEMAN implementation running in mobile phones and the CORE/EMANE network emulation are presented for several test scenarios[1].**

I. Introduction

Tactical and emergency response scenarios require efficient, robust, and secure network communication to quickly deliver data for situational awareness applications. The dynamic and resource limited constraints in these networks require that nodes opportunistically communicate with limited global knowledge, and make efficient use of the scarce available bandwidth. However, despite the remarkable increase in sensing, storage, processing, and communication capabilities in mobile devices, efficient dissemination and storage of content on the volatile network edge remains a challenging research problem. This paper presents and evaluates the ICEMAN (Information-CEntric Mobile Ad-hoc Networking) system. ICEMAN is an information-centric system designed for situational awareness applications operating in networks subject to severe disruption.

Considerable work has been done on information-centric networking [5] over the past few years. Section II summarizes this prior work and motivates some of the design choices made in ICEMAN.

Section III summarizes the main functionality of ICEMAN, which uses a publish-subscribe paradigm like prior ICN approaches, but adopts a declarative attribute-based approach to naming where applications request content or services by means of specifying attributes, their weights, a satisfaction threshold, and the number of content matches. This unified approach enables both long-standing subscriptions and immediate response queries through a standard publish-subscribe API. ICEMAN is a generalization of the Haggle [12] architecture, which was designed for pocket-switching networks where information exchange occurs through proximity encounters among two peers. ICEMAN uses UDP broadcast for the dissemination of interests, and UDP and TCP for the exchange of content. It uses network coding, the exchange of Bloom filters, and utility-based content caching to make the dissemination of content more robust in the presence of severe disruptions to network connectivity.

Section IV evaluates the performance of ICEMAN in tactical scenarios, based on its implementation in Haggle [12] running on Android phones and emulated tactical networks using CORE and EMANE.

II. Related Work

The prior work on efficient content dissemination in networks subject to severe disruption can be categorized into address-centric networking (ACN) and information centric networking (ICN), which differ on how naming and addressing are done. In ACN approaches, network nodes establish routes proactively or on demand to the addresses of destinations where services or content reside, and a directory service provides the mapping between the names of services and content to the addresses where they are located. By contrast, in ICN approaches, network nodes establish routes to content and services using their names directly. Given that no efficient solutions exist for directory services operating in disrupted environments, we focus our summary of related work on ICN approaches applied to disruption-tolerant networks (DTN).

The key differences among ICN schemes stem from the ways in which content is named or routed. The main approaches to naming in ICN consist of using self-certifying flat names (e.g., [8]), human readable hierarchical names (e.g., [7]), or metadata expressed as attribute-value pairs (e.g., [2]). These three approaches are interrelated; however, we believe that a declarative attribute-based approach offers more flexibility and can enable more efficient information dissemination in tactical networks.

[1]This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR Systems Center Pacific (SSC Pacific) under Contract N66001-12-C-4051, by SRI International, and by the Jack Baskin Chair of Computer Engineering at UCSC. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Most ICN schemes aimed at DTNs are based on either the epidemic dissemination of content (e.g., [15]), the dissemination of interests (e.g., [13]), or the maintenance of distributed hash tables (DHT) to allow nodes to publish and subscribe to content at specific nodes or geographical locations (e.g., [4]). ICEMAN is based on persistent interest dissemination. The reasons for this choice are that epidemic content dissemination consumes considerable bandwidth; and DHT-based approaches proposed to date are such that content (or links to content) may have to be placed far away from where it is produced or consumed in a DTN, which can incur substantial delays.

Pocket switching networks [6] are a specific type of DTNs in which information dissemination takes place primarily through close-proximity encounters between peers using a point-to-point protocol. Haggle [12] is an important example of a pocket switching system. ICEMAN augments the Haggle implementation by supporting communication among peers using UDP broadcast packets, which enables nodes to communicate with multiple neighbors concurrently.

III. ICEMAN Design

A. Units of Information Dissemination

The fundamental unit of abstraction in ICEMAN is a *data object* (DO) associated with *metadata*, represented as a set of attribute-value pairs, and a *payload* represented by a file. Each data object has at least a creation timestamp attribute, so that its creation time is well defined. A *data object identifier* is defined as the SHA1 hash over all this information, which is globally unique with high probability.

ICEMAN uses a special type of data object, a *node description*, to propagate cache summaries and application interests. Node descriptions have a limited lifetime and are periodically disseminated over multiple hops. Each node maintains the node descriptions of other nodes, even if they are not neighbors. The cache summary uses a Bloom filter data structure to approximate the set of data object identifiers of all the data objects in the local cache.

B. Declarative Attribute-based Naming

ICEMAN takes a declarative naming approach in which subscribers identify content as weighted attribute-value pairs and specify a satisfaction threshold and maximum number of matches. The process of finding matching data objects in the local cache occurs whenever a data object is received. If the data object is a node description, then local data objects are matched to the interests of the new node description. Otherwise, the new data object is matched to interests for node descriptions in the local cache. A node effectively computes a *degree of satisfaction* metric for each $\langle$ node description, data object $\rangle$ pair which denotes the *satisfaction* of the match. Data objects are retrieved, ranked, and prioritized at each node using a lexicographical ordering based on the degree of satisfaction and the creation time stamp (i.e., greatest satisfaction and freshest first). Only data objects exceeding the threshold specified by an application constitute a match and are eligible for dissemination. This approach is very flexible, in that arbitrary combinations of conjunctions and disjunctions over attributes can be expressed by transforming them to disjunctive normal form and issuing multiple subscriptions. This approach is efficient, because the network provides content discovery and applications can specify their interest at an arbitrary level of precision to limit query results (and query resource consumption). Only data objects that the node does not have count towards the maximum matching bound (by checking the cache summary).

C. Content Transport and Dissemination

ICEMAN's approach to information dissemination is based on the realization that both proactive (push-based) and reactive (pull-based) algorithms are needed for efficiency. In the reactive case, when a data object match occurs, the matched node may, or may not, be a neighbor. If the node is a neighbor, then no routing occurs and the matched data objects are sent directly by consulting the content transport policy. Otherwise, ICEMAN consults the content dissemination policy to determine the neighbors to which the data objects should be forwarded in order to reach the remote node. Only data objects that the matched node does not have are eligible for dissemination. In the proactive case, ICEMAN pushes a newly received data object to neighbors, even if there is no matching node. The dissemination policy selects which dissemination algorithm to use dynamically, based on the type of content (e.g., its attributes or payload size).

After a neighbor has been selected to receive a data object, ICEMAN uses the content transport policy to decide how the data object will be delivered between the connected nodes. The transport protocols used in ICEMAN support an application-layer two-phase protocol, the *control protocol*, between the sender (who initiates the transfer) and the receiver. The metadata is sent by the sender in the first phase and the payload is sent in the second phase (if the receiver accepts the data object). This approach suppresses redundant transmissions at the cost of additional control messages. ICEMAN currently supports TCP, UDP unicast, and UDP broadcast. Prior to sending a data object to a neighbor, the sender first checks whether there is a Bloom filter hit for the data object in its local view of the neighbor's Bloom filter. Both UDP unicast and UDP broadcast can optionally disable the control protocol (only Bloom filter information is used to ensure delivery).

D. Utility-Based Caching

Content caching is needed to reduce latency and bandwidth use, and becomes essential for content dissemination if no end-to-end physical path exists between a source and a destination. ICEMAN supports hard and soft replacement policies through a user-specified composition of utility functions. ICEMAN identifies useful content and prioritizes evictions accordingly (least utility first).

ICEMAN frames the caching problem in terms of a local online utility maximization problem (an approach similar to Chand et al [3] and Obraczka et al [14]). The caching policy defines a utility function that assigns a real number between 0

(lowest utility) and 1 (highest utility) to each data object in the cache. This utility function is a composition of multiple utility functions that are content and context sensitive (i.e., they vary in time and space). Data objects that do not meet a minimum threshold (as specified by the policy) are immediately evicted. Each utility function is multiplied by a policy specified weight constant between 0 and 1. Once the cache exceeds a certain watermark capacity, the pipeline chooses which data objects to evict to bring the cache capacity below the watermark. This eviction selection is posed as a 0-1 knapsack problem in which the watermark capacity is the bag size, the data object payload size is the cost, and the computed utility is the benefit.

The *Relative Time-to-live (RTTL)* function uses a specified attribute to determine how long a node should keep the data object. If the node has stored the data object for less than the specified time, then the utility is 1, and 0 otherwise.

The *Partial Order (PO)* function uses specified attributes to define a partial order on classes of data objects. If the data object can be replaced by another data object in the local cache, then the utility is 0, and 1 otherwise.

The *Immunity (IMM)* function assigns a utility 1 for a specified amount of time upon receiving the data object (it does not use an attribute), and 0 otherwise.

The *Neighborhood (NBR)* function uses the number of occurrences k of the data object within the one-hop neighborhood of size n to assign a utility of 0 with probability $\frac{k}{n+c}$, and 1 otherwise ($c = 2$ in our evaluation).

The *Least Recently Frequently Used (LRFU)* function computes a utility based on the recency and frequency of access as defined in [9]. We set $p = 2, \lambda = 0.01$ and normalize over the difference between the maximum and minimum computed LRFU value. We define an access as either a data object insertion or successfully sending the data object to a neighbor.

Different utility functions can be composed using sum, min and max functions. Each utility function returns a real number between $[0, 1]$ for each data object d based on the state of node i at time t (s_t^i). A non-zero threshold (0.10) is used; hence, if any constituent function within min assigns a utility of 0, then the entire utility is 0 and the data object is evicted. We state four example equations that aim to remove stale data and preserve fresh and popular data. We say that an individual constituent utility function imposes a *hard constraint* if it can evict a data object independently of the values of the other utility functions (it is *soft* otherwise).

$$
\begin{aligned}
U_1(d, s_t^i) &= \min\{\mathsf{RTTL}(d,t), \mathsf{PO}(d, s_t^i)\} \\
U_2(d, s_t^i) &= \min\{\mathsf{RTTL}(d,t), \mathsf{PO}(d, s_t^i), \\
&\qquad \max\{\mathsf{IMM}(d,t), \mathsf{NBR}(d, s_t^i)\}\} \\
U_3(d, s_t^i) &= \min\{\mathsf{RTTL}(d,t), \mathsf{PO}(d, s_t^i), \\
&\qquad \max\{\mathsf{IMM}(d,t), \mathsf{LRFU}(d, s_t^i)\}\} \\
U_4(d, s_t^i) &= \min\{\mathsf{RTTL}(d,t), \mathsf{PO}(d, s_t^i), \\
&\qquad \max\{\mathsf{IMM}(d,t), 0.7\mathsf{LRFU}(d, s_t^i) \\
&\qquad\quad + 0.3\mathsf{NBR}(d, s_t^i)\}\}
\end{aligned}
$$

E. Network Coding and Fragmentation

To improve the delivery of content, ICEMAN uses UDP broadcast and applies network coding and/or fragmentation at the content level. We use 32 KB blocks/fragments in this evaluation. Blocks and fragments are cached and disseminated by intermediate nodes. Both coded blocks and uncoded fragments remain unchanged; i.e., different from random-linear network coding, ICEMAN does not perform mixing of blocks at intermediate nodes. However, network coding and fragmentation differ: with network coding the source and seed nodes send innovative blocks with high probability while fragmentation can select only from a finite set of fragments. Additionally, network coding leverages peer nodes becoming seeds. These seeds can then send innovative blocks upon content reconstruction [10]. To separate the effect of coding from fragmentation, we also evaluated ICEMAN without coding but with fragmentation enabled (the data object is split into smaller fragments, where each missing fragment after randomization is disseminated separately).

F. Security

When content is published it is symmetrically encrypted to preserve its confidentiality. Nodes have their signing keys certified by one or more authorities. Each node only accepts content from a neighbor if they share a certification authority. The key is then transformed into an access capability using *multi-authority attribute-based encryption*. A policy framed over node attributes ensures that only authorized subscribers gain access to the content. Policies support a range of operators, including conjunction and disjunction. Policies can combine attributes from multiple authorities and nodes can receive their attributes from multiple authorities. After an access policy has been used, the associated symmetric key is cached and reused when subsequent publish operations are performed with the same policy.

IV. Performance Evaluation

A. Evaluation Parameters

We conducted three studies to evaluate the effectiveness of several policies for content transport, content dissemination, and content caching in ICEMAN. We used a single motivating scenario (Fig. 1) with three different traffic classes of situational awareness applications (Figs. 2, 3). The applications and scenario model a 30-node tactical scenario with an explicit social hierarchy (three squads of 10 members each). We use the Nomadic Community Mobility model to model squad movement[2]. In this model, groups are performing a random walk around their reference points which follow a Random Waypoint Model. There is high network connectivity within a squad, but intra-squad connectivity is very limited. Occasionally squads pass one-another and induce brief periods of high connectivity.

[2]We used the BonnMotion scenario generator: http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion/

Nodes	30
Area (m^2)	450×450
Duration (min)	30
Cool Down (min)	10
Mobility Seeds	4
Mobility Model	Nomadic
Min Velocity (m/s)	1.3
Max Velocity (m/s)	1.5
Max Pause (s)	2
Avg. Group Size	10
Std. Group Size	0
Ref. Max Pause (s)	2
Max Cluster Distance (m)	75
MAC	802.11 (EMANE)
Rate (Mbps)	24
Range Radius (m)	~ 45

Fig. 1. Scenario parameters used across all studies. No publications occurred during the last "cool down" minutes of the test. EMANE used an antenna gain of -5 dbi and a system noise factor of 4 db.

	A1	A2	A3
Sizes (KB)	1;5;10	250;500;1000	250;500;1000
Pub. Dist.	10s	exp., $1/\lambda = 60s$	exp., $1/\lambda = 300s$
Total Pub.	1840	690	152
Max Recv.	115200	2070	1520
Publishers	all	all	all
Subscribers	all	squad leaders	within squad

Fig. 2. Traffic classes. "Pub. Dist." refers to the inter-publication delay distribution; "exp." refers to an exponential distribution with parameter λ; a uniform distribution is used otherwise. "Max Recv." is the theoretical maximum number of data objects that can be received for the specified class, given a fully connected network without mobility, CPU, storage, and bandwidth constraints.

Each node runs applications that generate different situational awareness classes of traffic. A1 models a blue force tracking application that generates small sized content at a continuous rate (e.g., GPS coordinates or health vitals every few seconds). Everyone publishes content to everyone else. A1 is subject to RTTL with a value of 10 seconds, and PO (ordered by creation time). A1 always uses proactive replication for dissemination and UDP broadcast without control (the same policy as applied to node descriptions). A2 models a sensing application that collects data in response to random events, such as taking a photo of a suspicious vehicle, map annotations, or audio recordings. Content is pushed to the squad leaders who share the content with other squad leaders (content is pushed up the chain of command). A2 is subject to RTTL with a value of 900 seconds. A3 models intra-squad communication (e.g., the squad leader pushes an operational order to the squad). A3 is subject to RTTL with a value of 400 seconds. All content is delivered to the requesting application, leaving the cached content subject to the specified caching policy.

The mobility seed is varied to generate runs, and we report metrics over these runs. Different transport, dissemination, and caching policies are enabled for different studies (Fig. 3). We enable all three traffic classes across all of our studies, but examine metrics only for the relevant classes for the study. The content transport study varies across four policies, as defined in Fig. 4. We do not fragment or code data objects with payload sizes less than 32KB.

Study	Transport	Dissemination	Caching (Capacity)
Transport	*	ID	U1 (200MB)
Dissemination	T3	*	U1 (200MB)
Caching	T3	ID	* (20MB)

Fig. 3. Policies for each study. An asterisk indicates that the parameter was varied in the study.

	T1	T2	T3	T4
Transport	TCP	TCP	UDP-B	UDP-B
Control	yes	yes	no	no
Management	none	coding	coding	frag.

Fig. 4. The four evaluated content transport policies. UDP-B refers to UDP broadcast. Control refers to the control protocol. These polices are only applicable to A2 and A3 (A1 is always disseminated epidemically using UDP broadcast without control and without management).

The content dissemination study varies across the following four dissemination mechanisms:

- *Interest-driven approach* (ID): Interests are disseminated throughout the network and data objects traverse reverse paths established by interest requests (similar to [13]).
- *Mobility-driven approach* (MD): Data objects are forwarded to the neighbor with the highest probability of meeting the destination, based on prior knowledge of contact frequency and duration (i.e., [11]).
- *Proactive replication* (PR): Data objects are forwarded to all nodes, regardless of their interest.
- *NONE*: Data objects will only be delivered to nodes that are interested in them. Multi-hop routing through a relay that is not actively interested in the content will not occur.

The content caching study varies across the four example polices defined for $U_*(d, s_t^i)$ in Section III-D.

We use the real-time Common Open Resource Emulator, CORE 4.3 [1], to emulate networks and virtual hosts. We set a cpulimit of 15% for each virtual host[3].

The metrics of interest across all our experiments are total data objects delivered, data object delivery latency distribution, total transmit/receive bytes, and total data object delivered bytes. Total data objects delivered (and delivered bytes) only includes data objects that are delivered to an application that is interested in them. Data object delivery latency is the amount of time that it takes the data object to be received by the subscribing application, starting as soon as the data object is published. Total transmit and receive bytes includes all traffic that occurs on the channel.

B. Evaluation Results

Fig. 5 shows the performance of the four transport policies. We found that TCP has poor performance due to medium contention and the need for many-to-many communication. UDP broadcast with network coding achieved better performance than with fragmentation, because with high probability each

[3]http://cpulimit.sourceforge.net. This value was picked to roughly match the CPU resources on the target Nexus S phones.

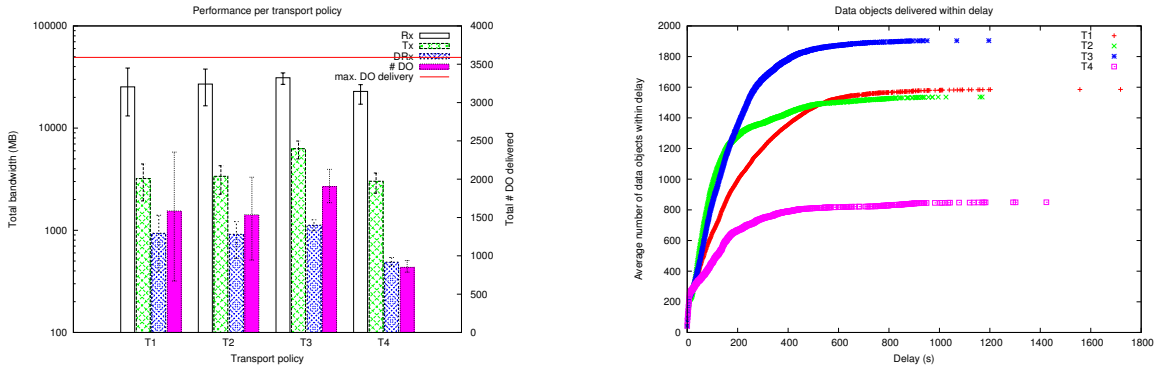

Fig. 5. Transport study results for A2 and A3. The error bars are the average, min, and max of 4 runs. We examine metrics for traffic classes A2 and A3, as these applications compose the majority of the network traffic in a situational awareness deployment. The average data objects received breakdown for (A1, A2, A3) is as follows. T1: 54265.80, 458.25, 1127.25; T2: 57645.20, 354.00, 1182.50; T3: 54600.20, 425.00, 1479.00; T4: 68526.50, 137.25, 712.25.

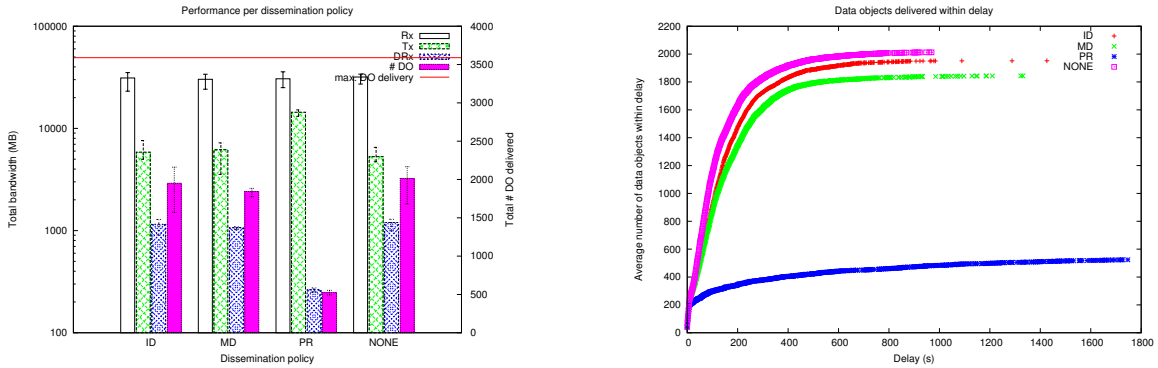

Fig. 6. Dissemination study results for A2 and A3. The error bars are the average, min, and max of 4 runs. The average data objects received breakdown for (A1, A2, A3) is as follows. ID: 57143.00, 479.75, 1472.25; MD: 47832.00, 364.00, 1480.00; PR: 14626.80, 164.25, 359.50; NONE: 64391.00, 506.75, 1506.25.

transmission provides new information to each overhearing node who has not yet reconstructed the content.[4]

Fig. 6 graphs the performance of the four dissemination policies. Interest-driven, mobility-driven, and NONE had similar delivery ratios due to the mobility and multi-subscriber application traffic which limited the number of multi-hop routing opportunities. For these particular settings, NONE utilized the channel more efficiently by waiting to meet the subscribers directly.[5] Proactive replication had poor delivery since it overwhelmed the resource constrained channel.

Fig. 7 shows the performance of the four caching policies. We set a small cache capacity of 20MB to examine the performance of the policies in an extremely constrained setting. The performance differences across the policies diminished with increasing cache capacity. The composition of both hard (RTTL, PO) and soft constraint policies achieved higher performance than individual hard constraints. The soft constraint policies achieved higher delivery by intelligently removing the least relevant data objects from the network first.

To understand the energy requirements for ICEMAN, we conducted several lifetime studies on Nexus S phones across different policies. Table I summarizes these results. The topology for these tests is a 10-node cluster with each node performing as both a subscriber and a publisher. Each node publishes six files with sizes ranging from 43.9 KB to 354 KB, staggered by 25 seconds every 900 seconds. The shortest life-time across all experiments is 4 hours and 42 minutes, and the minimum number of data objects published is 1173. With UDP broadcast, network coding yields a higher delivery percentage and better latency than fragmentation. For reference, atomic transfer over TCP has better latency and lower energy consumption in this static many-to-many scenario, but it is often not an option for tactical disruptive environments with intermittent connectivity and many-to-many communications (as our emulation study demonstrates).

TABLE I
NEXUS S PHONE BATTERY LIFE-TIME RESULTS COMPARISON OF UDP BROADCAST WITH CODING, UDP BROADCAST WITH FRAGMENTATION, AND TCP WITH ATOMIC TRANSFER. THE TEST ENDS WHEN THE LAST PHONE RUNS OUT OF BATTERY POWER.

Transport	Coding	Fragmentation	Atomic
Avg. Delay (sec)	22.258	86.333	5.989
Avg. Energy Spent per DO (%)	0.008979	0.010233	0.007125
Avg. Battery Lifetime (min)	281.8	333.9	361.3
Number of Published DOs	1173	1374	1466
Number of Received DOs	11137	9772	14036
Delivery Rate (%)	94.94	71.12	95.74

To measure the impact of our security approach, Table II shows the end-to-end latency (in seconds) of publishing a data object with a varying number of node attributes in the access

[4]In other scenarios with a higher degree of intermittent connectivity, we found that network coding improved data object delivery even over TCP.

[5]In other scenarios with less connectivity and fewer subscribers per data object, we found that ID and MD can outperform NONE and PR.

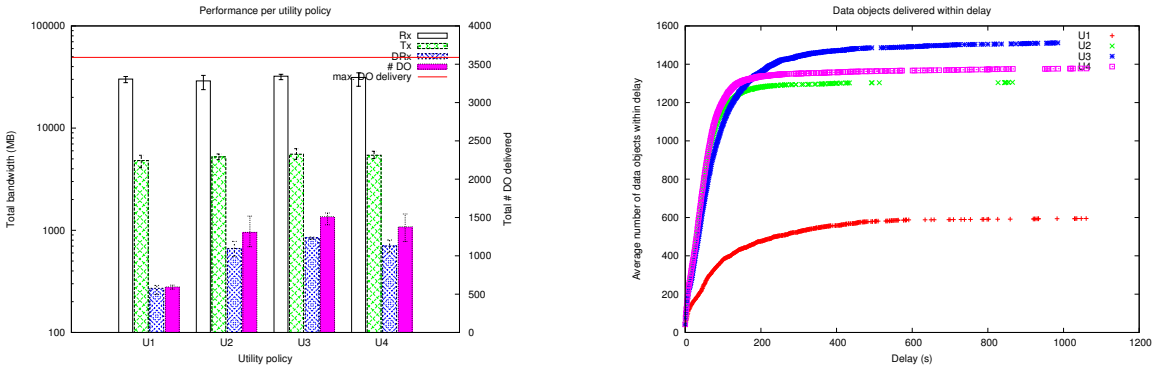

Fig. 7. Caching study results for A2 and A3. The error bars are the average, min, and max of 4 runs. The average data objects received breakdown for (A1, A2, A3) is as follows. U1: 19160.00, 127.25, 467.75; U2: 55032.50, 332.50, 973.25; U3: 50727.00, 257.75, 1254.00; U4: 55877.20, 324.50, 1053.50.

TABLE II
PUBLISH-SUBSCRIBE TIME (IN SECONDS) WITH NEW/CACHED ACCESS POLICIES.

Size Attributes	200 KB	400 KB	600 KB	800 KB	1 MB
0	0.31/0.31	0.50/0.50	0.64/0.64	0.81/0.80	0.96/0.96
2	0.89/0.40	1.02/0.62	1.17/0.87	1.48/0.99	1.66/1.16
4	1.22/0.45	1.36/0.62	1.61/0.82	1.88/1.00	2.23/1.38
6	1.59/0.44	1.75/0.71	1.87/0.86	2.21/1.03	3.29/1.26
8	1.97/0.38	1.98/0.57	2.16/0.82	2.74/1.03	2.85/1.17

policy, for various data object sizes. Zero attributes indicates that no encryption is performed. Since each policy used is the conjunction of all attributes, the time reported is the maximum that will be used; if a policy contains disjunctions, the time required will be lower. Each value reported is the average of four runs. We see that publishing with cached access policies (the normal case in practice) imposes dramatically lower overhead when compared to publishing with new access policies.

V. Conclusion

ICEMAN is an information-centric architecture that supports situational awareness applications at the tactical edge. Through extensive evaluation of ICEMAN in scenarios that model tactical mobility and applications, we found that a mixture of content-based policies are necessary to achieve the best performance. These results suggest further exploring dynamic selection of transport, dissemination, and caching policies, based on network context. Our experiments on battery life-time on Android phones demonstrate the feasibility of our approach (especially network coding) on current hardware. Our security evaluation demonstrates that the high cost of attribute-based encryption is quickly amortized when policies are reused, and in practice only the cost of symmetric encryption remains. ICEMAN's compositional architecture enables users to easily mix and evaluate an assortment of policies, to support efficient communication at the tactical edge for multiple simultaneous classes of situational awareness traffic.

References

[1] J. Ahrenholz, C. Danilov, T.R. Henderson, and J.H. Kim. CORE: A real-time network emulator. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008.

[2] Antonio Carzaniga and Alexander L. Wolf. Content-based networking: A new communication infrastructure. In *Revised Papers from the NSF Workshop on Developing an Infrastructure for Mobile and Wireless Systems*, IMWS '01, pages 59–68, London, UK, UK, 2002. Springer-Verlag.

[3] N. Chand, RC Joshi, and M. Misra. Cooperative caching in mobile ad hoc networks based on data utility. *Mobile Information Systems*, 3:19–37, 2007.

[4] M. Varvello et al. On the design of content-centric manets. In *WONS 2011*, 2011.

[5] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[6] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket switched networks and human mobility in conference environments. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, WDTN '05, pages 244–251, New York, NY, USA, 2005. ACM.

[7] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[8] T. Koponen, M. Chawla, B.G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 181–192. ACM, 2007.

[9] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim. LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE transactions on Computers*, 50(12):1352–1361, 2001.

[10] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in vanet. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, MobiShare '06, pages 1–5, New York, NY, USA, 2006. ACM.

[11] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. In *SIGMOBILE Mobile Computing and Communication Review*, 2004.

[12] E. Nördstrom, P. Gunningberg, and C. Rohner. A search-based network architecture for mobile devices. *TR, Department of Information Technology, Uppsala University*, 2009.

[13] Ignacio Solis and J. J. Garcia-Luna-Aceves. Robust content dissemination in disrupted environments. In *Proceedings of the Third ACM Workshop on Challenged Networks*, CHANTS '08, pages 3–10, New York, NY, USA, 2008. ACM.

[14] K. Obraczka T. Spyropoulos, T. Turletti. Routing in delay-tolerant networks comprising heterogeneous node populations. *IEEE Transactions on Mobile Computing*, pages 1132–1147, 2009.

[15] Armin Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000.