STRATEGIC WHITE PAPER

PROVIDING LOCAL CONTENT DISCOVERY AND SHARING IN MOBILE TACTICAL NETWORKS

Mary R. Schurgot*, Jairo Estebany†, Lloyd Greenwald*, Yang Guoy†, Mark Smithy†, David Stott*, Matteo Varvelloy†, Limin Wangy†

*LGS Innovations LLC, USA

†Bell Labs, Alcatel-Lucent, USA

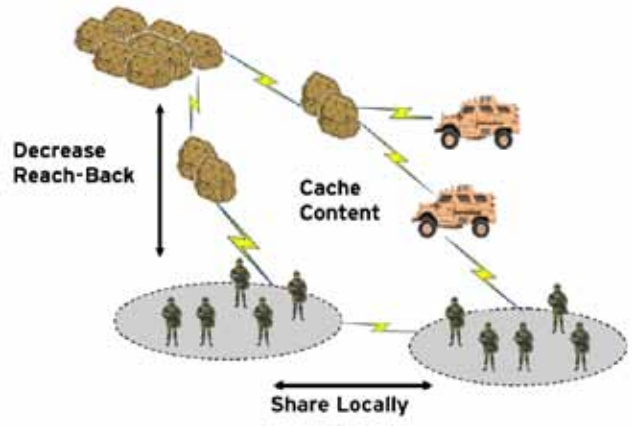
***Abstract*—** The tactical edge is filled with rich content, providing real-time intelligence and situation awareness to the warfighter. The current network architecture lacks support for lateral sharing of this critical content. Content is currently accessed from known servers and may require reach-back over bandwidth-constrained links. In this work we present SCALE, a Scalable Content-centric Architecture ensuring Locality and Efficiency. SCALE focuses on sharing data as it is created and is optimized for ad hoc collections of mobile nodes, rather than defined clients and servers. Any node can create and share content and nodes can discover content without knowing ahead of time where to look for it. SCALE provides mechanisms by which content is replicated throughout the network by organically caching content as it transits through the network or prepopulating content caches for existing content. SCALE provides a distributed index based on multi-node resolution, in which nodes dynamically take on responsibility for subsets of the content index based solely on being in a particular geographic location. We provide an evaluation of SCALE's mechanisms for providing local content discovery and sharing based on a SCALE prototype running on Android devices. These results show that SCALE can improve the probability that content is available locally and decrease the number of hops needed to access content. Dynamic ad hoc content sharing apps (e.g. social networking, microblogging, photo sharing, search, chat channels, etc.) can be built on top of SCALE to provide enhanced situation awareness and command and control operations for mobile tactical users

TABLE OF CONTENTS

I. INTRODUCTION

In this work, we describe a prototype implementation of SCALE, a Scalable Content-centric Architecture ensuring Locality and Efficiency. SCALE is being developed as part of the DARPA Content-Based Mobile Edge Networking (CBMEN) program and shares objectives with the overall program, namely improving warfighter access to content compared to centralized content storage solutions by decreasing reach-back and increasing local availability. Increasing local availability of content helps reduce content retrieval latency and gains available bandwidth, especially over bandwidth-constrained tactical links. The SCALE features of local content sharing and decreased reach-back are graphically depicted in Fig. 1. SCALE is an example of a content-centric networking (CCN) architecture [1], [2], [3]. CCN improves on previous ad hoc approaches to content delivery by providing a simple, unified, flexible communication architecture. Data is requested by name, not server address, removing the primacy of network topology and end-to-end connectivity in content delivery and further providing robustness to node failures, mobility and attacks. In [4] we describe our initial research on bringing content-centric networking into the mobile ad hoc networking (MANET) environment. In that work, we demonstrate how content popularity is a primary driver of design decisions for how to develop CCN architectures for MANETs.

Fig. 1. SCALE permits local content sharing of content, achieving decreased reachback to centralized servers.



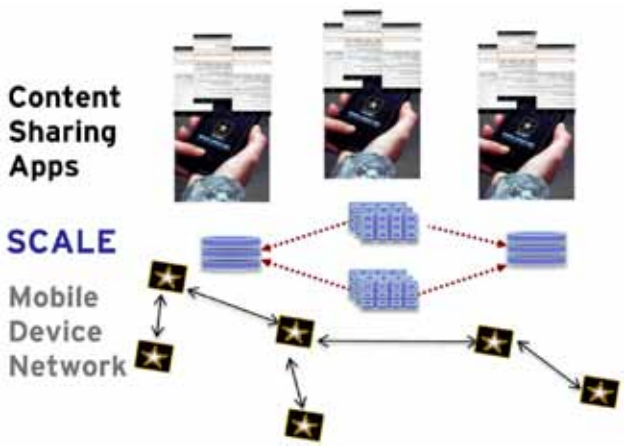
The current version of SCALE provides content discovery using multi-node resolution, content availability using on-path caching, content routing using location-based routing, and the ability to name content using a sequence of attribute/value meta-data components. Two important properties of SCALE are 1) fast discovery of content and fast local retrieval of popular content for mobile tactical users, and 2) self-organizing infrastructure to support dynamic ad hoc content sharing (e.g. social networking, microblogging, photo sharing, search, chat channels, etc.) for mobile tactical users.

SCALE can be contrasted with TIGR (Tactical Ground Reporting system), a groundbreaking information management system that has been deployed to the field to provide new capabilities for the warfighter [5]. TIGR was initially developed as part of the DARPA ASSIST program and provides multimedia geospatial information management. TIGR is based on a distributed client-server database system that includes a distributed repository system to store and share combat information, e.g. patrol brief data. TIGR distributes media rich information to the tactical edge with a range of deployment scenarios that accommodate varying degrees of network connectivity. TIGR focuses on having content available even when the network is not available for extended periods. Locality is achieved through local servers, search indices and caches, replicating content that is needed locally, based on replication filters and local policy. TIGR replication can use available network connectivity or disk exchange when there is no connnectivity. TIGR includes a handheld version that works by preloading mission data onto handheld device prior to mission execution. Post mission any content created by the handheld user is uploaded to the TIGR server. If connected during a mission, a TIGR handheld has the ability to sync data to the server over the air.

By contrast to TIGR, SCALE focuses on sharing data as it is created and is optimized for ad hoc collections of nodes, rather than defined clients and servers. Any node can create and share content and nodes can discover content without knowing ahead of time where to look for it. Rather than providing a fixed content replication infrastructure, SCALE provides mechanisms by which content is replicated throughout the network by organically caching content as it transits through the network or pre-populating content caches for existing content. Content can be discovered using in-network queries or subscriptions based on content meta-data. Rather than replicating search indices for locally replicated content, as in TIGR,

SCALE provides a distributed index based on multi-node resolution, in which nodes dynamically take on responsibility for subsets of the content index based solely on being in a particular geographic location.

Fig. 2. SCALE is designed as a layer that sits between the mobile device network and content sharing apps.



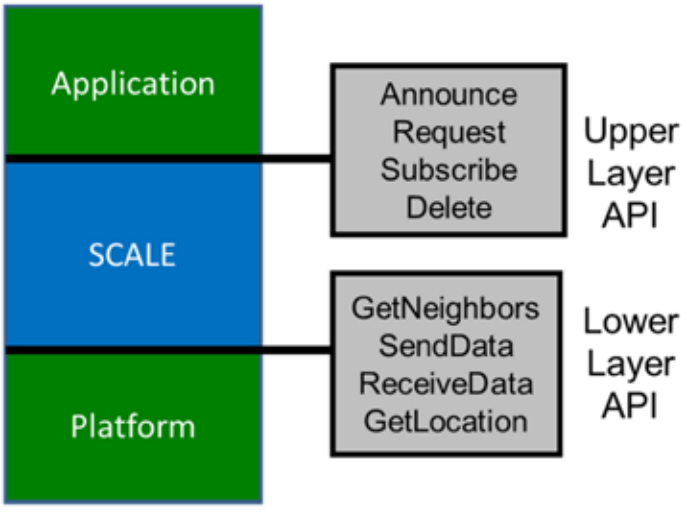
SCALE can be a stand-alone edge content sharing solution or can be deployed as a complement to a TIGR deployment; providing ad hoc real-time content sharing to complement TIGR's content distribution infrastructure. Additionally, SCALE can support a variety of novel and existing content sharing applications. For example, applications developed under DARPA's Transformative Apps program can be ported to SCALE to support a diverse set of missions, including tactical battlefield, humanitarian, disaster recovery, etc. SCALE will increase the utility of both TIGR and Transformative Apps by introducing content sharing at the tactical edge, while supporting the rich set of apps created in these programs.

SCALE is designed as a layer that sits between the mobile device network and content sharing apps, as depicted in Fig. 2. SCALE supports on-demand queries for content as well as persistent subscriptions for content. Both queries and subscriptions can be based on complete content names or by name components or ranges of values for name components. SCALE specifies an API to content sharing apps that includes the ability to announce, request and subscribe to content; as well as an API to interact with the mobile device platform to allow for sending and receiving of messages, neighbor discovery, and node location discovery, as depicted in Fig. 3.

In this paper we focus on the mechanisms that SCALE uses to ensure locality and efficiency in content discovery, content sharing and mobility support. SCALE ensures locality and efficiency through on-path caching, content discovery with local resolution, and mobility support with local mechanisms for locating content that has moved and exchanging content pointer information as resolution nodes move. In [6] and [7] we provide detail on our on-path caching solution and multicomponent naming/range query solution, respectively. In this paper we focus on an overview of these mechanisms and initial performance results based on our prototype SCALE implementation.

We describe each mechanism in Section II and provide preliminary performance results based on a prototype implementation of SCALE on Android™ Galaxy™ Nexus™ phones[1] in Section III.

Fig. 3. SCALE provides an API of basic operations to content sharing apps as well as an API to interact with the underlying mobile device network.



[1]Galaxy is a trademark of Samsung Electronics Co., Ltd. Nexus and Android are trademarks of Google Inc

II. SCALE SYSTEM DESIGN

A SCALE-enabled network is an ad hoc network of devices running the SCALE system software. Nodes are distinguished by an identifier, the "nodeID", and geographic location. We assume a location service is available at each node. Devices can communicate directly without a fixed infrastructure using the GPSR (Greedy Perimeter Stateless Routing) protocol [8]. GPSR is a geographic-based protocol and relies on the locations of the nodes to route messages towards a geographic destination. The SCALE implementation of GPSR also includes the Cross Link Detection Protocol (CLDP) extension discussed in [9].

SCALE provides for three main functions: publishing content, querying for content, and subscribing to content. We describe those operations here.

Publish – When a node publishes content, a publish request is passed from the application to the SCALE layer. The content is described by meta-data, which forms the content name. The composition of a name will be discussed further in Section II-A. As in Geographic Hash Table (GHT) [10], a hash of the resolution location. A pointer to the content is routed towards this resolution location via an `announce` message; the pointer includes the content name and a reference to the content host including nodeID and geographic coordinates. The routing algorithm terminates when the `announce` message reaches the node closest to this resolution location. This resolution node then becomes responsible for all of the pointers for content with this name.

Query – In a content-based network, content is requested by name. With SCALE, there are two steps for a query: content discovery and content retrieval. The SCALE layer handles query requests similar to publish requests in that the provided content name is hashed and a resolution location is identified. During the content discovery step, a `find` message is routed to the resolution node responsible for pointers to the requested name. If the content exists in the network, a `resolution` message is returned to the requester. The requester then initiates the content retrieval step with a get message to request the content from the publisher. The publisher responds with a `content` message, which contains the requested content. As discussed in Section II-B, content can be cached by intermediate nodes on the path from the publisher to the requester.

Subscribe – The system allows for persistent queries: nodes can subscribe to content of interest. A `subscription` message is routed to the resolution node responsible for pointers to the content of interest. The resolution node stores the requester's interest (content name) in its Subscription Table with the requester's nodeID and location. When an `announce` message matching the subscription reaches the resolution node, the resolution node forwards a `resolution` message to the requester and the process proceeds as with a query. The subscription persists until the application unsubscribes from the content.

A. NAMING FOR LOCAL CONTENT DISCOVERY

The content name is provided by the application during a publish, query, or subscribe operation. A content name is composed of a set of B attribute/value pairs, which we call components. For example, to describe this paper, we could use the following components separated by a semi-colon: `type=conference;length=6;year=2013;name=paper19`. Each component describes a specific attribute of the content, e.g., it is a conference paper of length six pages, and it was prepared in 2013. Its unique identifier is paper19. Instead of computing the hash of the full name, `type=conference;length=6;year=2013;name=paper19`, we hash each component individually; in this example, we derive `h(type=conference)`, `h(length=6)`, `h(year=2013)`, `h(name=paper19)`. As previously described, the result of a hash operation is a pair of coordinates in a two dimensional space. Thus, B resolution locations are selected per content. Each resolution node is responsible for a particular component, but stores the full content name for query resolution.

SCALE supports both point and range queries. A "point query" is a query for a specific content and requires the iden tification of a unique attribute value; whereas a "range query" is a query for content whose attributes are within a requested range. In the following, we detail how SCALE supports both query types.

Point Query – The requester computes the B content pointers from the content name; then, it selects the closest pointer to its current location by computing the Euclidean distance between its coordinates and all pointers. Finally, it sends a `find` message addressed to the closest pointer requesting the desired content; the `find` message is forwarded to the corresponding resolution node via GPSR. The resolution node uses the name of the content requested to lookup the nodeID and most recent location of the content host for this content and returns it to the requester via GPSR.

Fig. 4. Protocol messages in SCALE; Point query example.

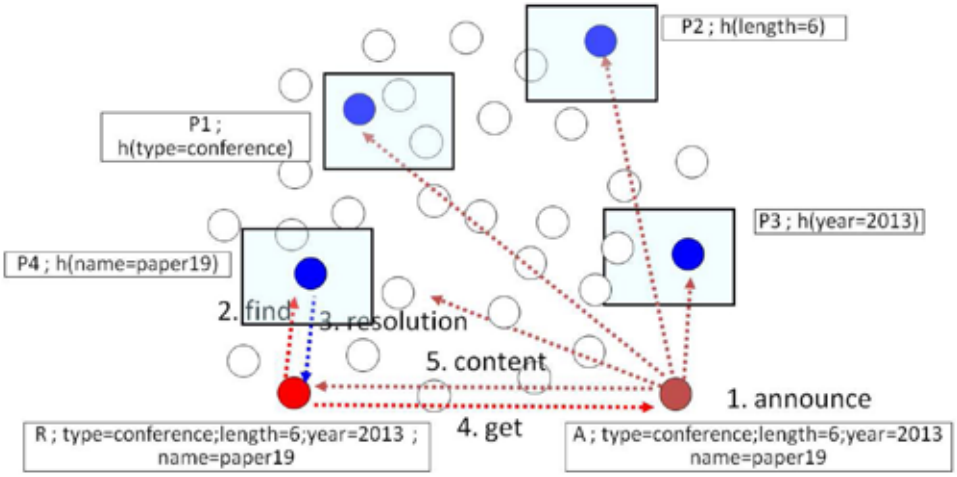
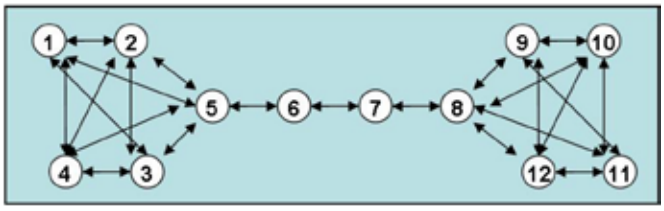


Figure 4 graphically shows how SCALE works in the case of a point query and a single content item with name `type=con ference;length=6;year=2013;name=paper19` located at node A. First node A announces the availability of such content by sending announce messages to nodes P1, P2, P3 and P4, the resolution nodes for the B components. Then node R requests content `type=conference;length=6;year=2013;name=paper19`: to do so, a find message is sent to node P4 which is responsible for the closest pointer `name=paper19`. Node P4 replies with the information about the content host, i.e., its nodeID and location. Node R uses such information to send a get message to A, to which A replies with a `content` message that contain the requested data.

Range query – If a unique identifier is not included in the content name, we consider the query a range query. The current prototype implementation for a range query is as follows. For a single dimension range query, the content name is composed of a single attribute/value pair; thus, only one resolution node is responsible for all of the pointers to fill this query. A query for all conference papers `type=conference` would be resolved by node P1 in Figure 4.

A multi-dimensional range query is the extension of a single dimension range query to multiple attributes. As in the point query case, the `announce` message can be routed toward the closest resolver for a multi-dimensional range query. SCALE also supports queries for content within numerical ranges via database queries. For example, a query for all conference papers between 2010 and 2013 could be extracted from the database of node P1. Future extensions of our prototype implementation will dynamically create resolution nodes for frequently requested ranges as described in [7].

Fig. 5. 12-Node Dumbbell Network Topology



B. CACHING FOR LOCAL CONTENT RETRIEVAL

In SCALE, nodes are equipped with storage space and can naturally serve as caches for future content retrieval. Caching is a proven technique in helping web content distribution over the Internet [11], [12]. As content requests typically obey the Zipf distribution [13], copies of popular objects are stored at multiple caches and become available locally. Even a small size cache can greatly reduce the network traffic, alleviate the server workload, and shorten the access latency.

On-path caching is especially suitable for content distribution over MANETs. In on-path caching, a SCALE node intercepts the content requests passing through it. If the requested object is in the cache, the object is sent to the requester and the request will not be propagated further upstream. Otherwise, intermediate nodes forward the request along the regular routing path toward the content host. If a cache does not contain the requested object, the request will eventually be served by the original host. LRU (least recently used) is a well studied and very effective cache replacement strategy. We employ LRU for cache replacement in SCALE.

On-path caching in SCALE provides for high content availability, reductions in network traffic, and short content retrieval latency. SCALE mitigates the challenges introduced by lossy wireless links, bandwidth limitations, and mobile nodes by bringing content closer to the user through on-path caching. We developed performance models in [6] to evaluate the expected caching performance in SCALE with respect to average content availability, average number of messages, and average latency. The benefits of on-path caching are significant. In all cases examined in [6] and the prototype experiments described in Section III, on-path caching improves the average content availability and the average number of messages compared to a solution without caching (i.e. cache size of zero).

C. MOBILITY MANAGEMENT

The mobility of content hosts and resolution nodes can detract from SCALE's performance. When a content host moves without re-announcing its location to the resolution nodes, a requester might fail to contact it and thus not fetch the desired content. Similarly, if the resolution nodes move and do not transfer their pointers to new resolution nodes, a requester might not be able to retrieve content due to a failure of the resolution operation. SCALE implements mechanisms to handle mobility of both content hosts and resolution nodes. In the following, we first describe the solution we adopt for content host mobility and then we focus on mobility of resolution nodes.

A naïve solution to handle content host mobility consists of re-announcing content with ***announce frequency F***. Reannouncing content availability every ***F*** can be very inefficient, for example when a content host does not change its location between two successive announcements. Similarly, if a host moves right after an `announce` message was sent the announce information will be stale for the whole time 1F .

The announce frequency ***F*** should be low enough to minimize the network traffic overhead associated with the announce operation; yet ***F*** should be high enough to reduce stale information at the resolution nodes. SCALE implements a lazy announce mechanism (low ***F***) coupled with the following mechanisms to react in the presence of stale announce information.

Local Flooding – A `get` message is forwarded to the last known location of a content host. If the node is not found, a flooding operation is started. The closest node to this location, as indicated by GPSR, first checks its neighbors looking for the requested node. If unavailable, it asks its neighbors to look for the content host in their neighbor list. We repeat this operation until the content host is found or when a maximum numberof hops is reached.

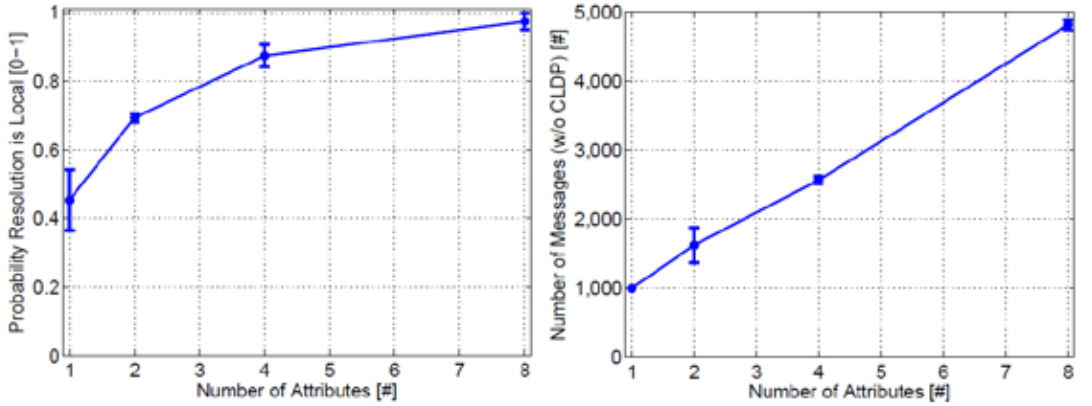
Re-announcing – Once a requester has located the content host via local flooding, it updates the resolution node which it has previously contacted with the most recent location of the content host. Since a requester only updates the resolution node it has previously contacted, this more precise updated information about a content host's location is only available where such content is more popular (high number of requests). The rationale is that stale announce information for a given content can be tolerated in the portion of the network where such content is unpopular (low number of requests). Note that since a node only updates its closest resolution node, this re-announce operation results in fewer network messages compared to a classic announce.

We now focus on the mobility of resolution nodes. A simple mechanism to mitigate the effects of resolution node mobility is to assign ***K*** resolution nodes per component, i.e. create ***K***-1 replicas. Each primary resolution node then simply monitors its neighbors independently. As one of its neighbors gets closer than itself to a resolution location for pointer ***P***, we act as follows. The "old" resolution node asks the candidate "new" resolution node whether it is already a resolution node for ***P***. If yes, this node then becomes the primary resolution node. If no, the old resolution node transfers the announce information for ***P*** to the new primary resolution node.

III. SCALE PROTOTYPE AND EVALUATION RESULTS

The basic SCALE functionality has been implemented over a network of Galaxy Nexus phones connected in ad hoc mode running Android 4.1.1. The devices communicate over-theair via Wi-Fi and run the SCALE software. SCALE assumes neighbor discovery and location services are provided by lower layers. Each phone senses its unique location and dynamically discovers its mobile neighbors. A testing application exercises the SCALE layer by publishing, querying, and subscribing to content. The content actions (publish, query, subscribe) for an experiment are predefined and loaded onto the devices.

Fig. 6. Local Content Discovery



(a) Probability a resolution node is within one hop

(b) Number of sent messages increases with the number of components

For evaluation purposes, we use the CORE emulator [14] to network a collection of Android Virtual Devices (AVDs) [15]. As a first step to testing the functionality of the SCALE software and characterizing the behavior of our algorithms, the emulated network was configured according to the CORE defaults: packet error rate of 0, jitter of 0, link delay of 20 ms, connectivity based on a pixel distance of 275, and a bandwidth of 54 Mbps. The emulated devices (AVDs) were configured with the following parameters: target API 16, ARM processor, SD card of 240 MiB, and 512 MB of RAM. In the remainder of this section we present a series of experiments performed using this setup to demonstrate the local content discovery and local content retrieval enabled by SCALE.

A. LOCAL CONTENT DISCOVERY

During the content discovery process, SCALE sends a `find` message to locate the resolution node responsible for the desired content name. As discussed in Section II-A, ***B*** components make up a content name. The resolution location for the component which hashes to the location closest to the requester is selected. The expectation is that as the number of components increases, the likelihood a resolution node is found within a requester's local neighborhood also increases. The local neighborhood includes nodes within one hop of the requester. We demonstrate this property using the 12-node dumbbell topology (without mobility) illustrated in Fig. 5.

Node 1 in Fig. 5 publishes 30 pieces of content of size 1023 bytes. Each content item is assigned a random name of ***B*** components, i.e. the attribute is a random string and the value is a random string. For each run of the experiment, ***B*** is varied: 1; 2; 4; 8. Node 10 requests all 30 pieces of content using all ***B*** attributes of the name. We measure the number of hops the `resolution` message travels, and the total number of messages (minus CLDP traffic). The local query count ***L*** is increased

each time a query is resolved in 0 or 1 hops. The probability a query is resolved by a local node is calculated by dividing this final number ***L*** by the total number of queries 30. Fig. 6(a) shows the five trial average probability with error bars representing one standard deviation for the 12-node dumbbell experiment. Note that as the number of components increases, the probability increases as expected. The addition of components incurs messaging overhead seen in Fig. 6(b). The cost of additional small `announce` messages may be a reasonable tradeoff to resolve a query within one hop for a time-sensitive application.

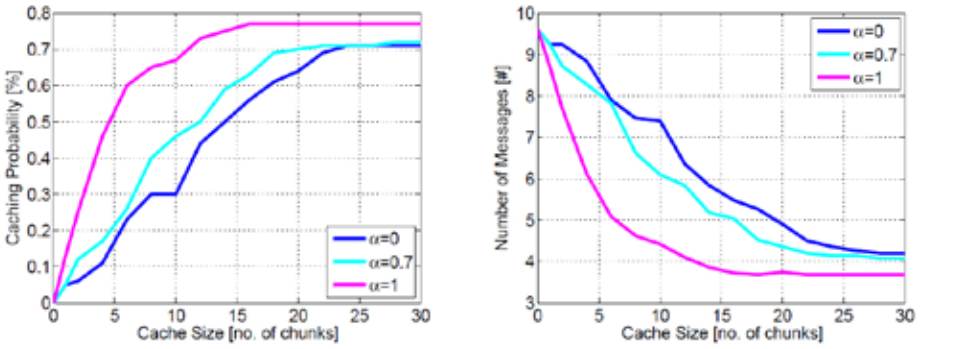
B. LOCAL CONTENT RETRIEVAL

To demonstrate the benefits of on-path caching of content, we design an additional experiment using the dumbbell topology of Fig. 5. The publisher (node 1) is located on the left side of the dumbbell and announces 30 content items using a single component. The rightmost nodes (8-12) each randomly request twenty pieces of content according to the Zipf distribution. The storage size of the caches and the Zipf parameters are varied. Figure 7(a) shows the probability a request is filled by an intermediate node's cache as the cache size increases and is changed: 0; 0:7; 1. As the cache size increases and the content popularity becomes more skewed, content becomes more local to the requester through on-path caching. We also observe an improvement in terms of the number of round trip messages needed to satisfy a query as the parameters are varied.

C. MOBILITY MANAGEMENT

In order to evaluate mobility management, we leverage an event-driven simulator we have developed. In fact, the goal of this evaluation is to estimate how our solutions perform under real mobility which is hard to realize with real phones and users. The simulator takes as input a mobility trace and maintains GPSR information with a resolution of one second. On top of GPSR, we implemented SCALE's mechanisms to announce content, resolve queries and handle the mobility of both content hosts and resolution nodes.

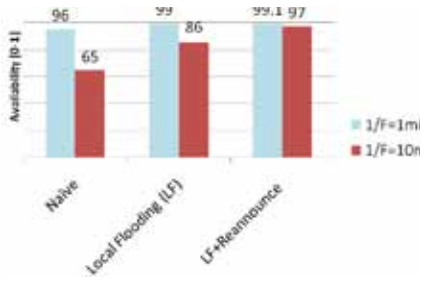
Fig. 7. Local Content Retrieval



(a) Probability Request Filled by Cache (b) Average Number of Messages to Fill Request

We compare the availability of the na¨ıve solution with local flooding (LF) and LF coupled with re-announcing (cf. Section II-C) assuming "high" and "low" frequencies, $1/F$ =1min and $1/F$ =10min, and the KAIST[2] campus trace from CRAWDAD [17]. Figure 8 summarizes the results. When $1/F$ =1min, the three solutions perform very similarly: extending the na¨ıve solution with both LF and re-announcing only increases the availability from 96 to 99.1%. The benefits of LF+reannouncing are more visible when $1/F$ =10min; in this case, LF increases the availability by 21%, e.g.,from 65 to 86%. The re-announce mechanism enables a further gain of 11%, from 85 to 97%, for a total improvement of 32% provided by the combined solution.

Fig. 8. Mobility Management



[2]The KAIST trace spans 92 humans walking in a campus network, and has a length of 23h18m [16].

IV. CONCLUSION

We have presented an overview of SCALE, a Scalable Content-centric Architecture ensuring Locality and Efficiency. We have focused on the mechanisms that SCALE uses to ensure locality and efficiency in content discovery, content sharing and mobility support. We provided results based on a prototype implementation of SCALE on Android Galaxy Nexus phones. These results showed that SCALE can improve the probability that content is available locally and decrease the number of hops needed to access content. These improvements become greater as the distribution of queries becomes more skewed toward highly popular content items. We also showed that content discovery by name can occur locally with few hops as a function of the number of resolution nodes we use as part of our ad hoc distributed content index. Finally, we provided simulation performance results that showed the expected impact of mobility support, comparing the naïve approach to mobility support with our local mechanisms.

V. ACKNOWLEDGMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) CBMEN Program and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001- 12-C-4197. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views, official policy or position of DARPA, SSC Pacific, the Department of Defense or the U.S. Government. Distribution Statement A: Approved for Public Release, Distribution Unlimited. The authors gratefully acknowledge USC Information Sciences Institute (ISI) for their efforts as mobile system integrators on the DARPA CBMEN Program.

ABOUT LGS INNOVATIONS

LGS Innovations delivers next generation solutions that solve the most complex networking and communications challenges facing the U.S. Federal Government, State and Local Governments, Foreign Governments, and commercial enterprises. LGS offers groundbreaking research and development and builds advanced wireless, optical, and wired products and applications customized for specific mission environments. These solutions provide unique information and security advantages that lead to the operational success of its customers. LGS' offerings include:

- Campus and building networking solutions for military bases, hospitals, and corporate centers
- Maritime applications for in-port and at sea communications
- Global networks (long-haul communications, including undersea cable)
- Enterprise voice, video, and data networking
- 4G wireless deployable communications for public safety, battlefield, and emergency and first responder communities
- Network engineering, integration, and installation
- Cloud and data center infrastructure
- Video teleconferencing and IPTV suites
- Research and development in advanced multimedia/RF communications, cybersecurity, sensing technologies, and photonics

LGS Innovations is a U.S.-owned company headquartered in Herndon, Virginia, with offices in Colorado, Illinois, Maryland, New Jersey, New Mexico, and North Carolina. Formerly a subsidiary of Alcatel-Lucent, LGS is the exclusive reseller of Alcatel-Lucent products and services to the U.S. Federal Government and any other entity when the end customer is the U.S. Federal Government. LGS maintains strong ties to Bell Labs and its technologies, employing more than 450 scientists and engineers and a total of nearly 700 employees worldwide. To learn more about LGS Innovations, visit www.lgsinnovations.com.

VI. REFERENCES

[1] M. Gritter and D. Cheriton, "An Architecture for Content Routing Support in the Internet," in USITS'01, (San Francisco, CA, USA), March 2001.

[2] V. Jacobson, D. K. Smetters, J. D. Thronton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Network Named Content," in CoNEXT'09, (Rome, Italy), December 2009.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," in SIGCOMM 2007, (Kyoto, Japan), August 2007.

[4] M. Varvello, I. Rimac, U. Lee, L. Greenwald, and V. Hilt, "On the Design of Content-Centric MANETs," in WONS'2011, (Bardonecchia, Italy), Jan. 2011.

[5] B. Ewy, M. Swink, S. Pennington, J. Evans, J. Kim, C. Ling, S. Earp, and M. Maeda, "TIGR in Iraq and Afghanistan: Network-adaptive distribution of media rich tactical data," in IEEE Military Communications Conference (MILCOM), pp. 1–7, Oct 2009.

[6] Y. Guo, L. Greenwald, M. Schurgot, and M. Varvello, "Performance Model for a Cache Enabled Content Distribution Framework over MANET," tech. rep., 2013.

[7] M. Varvello, J. Esteban, M. Smith, L. Greenwald, Y. Guo, and M. Schurgot, "ERMINE: Enabling Range Queries in MANET with Information- Centric Networking," tech. rep., 2013.

[8] B. Karp, "GPSR: Greedy perimeter stateless routing for wireless networks," in MobiCom00, (Boston, MA, USA), pp. 243–254, Aug. 2000.

[9] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic Routing Made Practical," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05, pp. 217–230, 2005.

[10] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," in WSNA'02, (New York, NY, USA), pp. 78–87, ACM, 2002.

[11] J. Wang, "A Survey of Web Caching Schemes for the Internet," ACM SIGCOMM Computer Comm. Rev., vol. 29, no. 5, pp. 36–46, 1999.

[12] B. Davison, "A Web Caching Primer," IEEE Internet Computing, vol. 5, no. 4, pp. 38–45, 2001.

[13] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in Infocom'99, April 1999.

[14] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim, "CORE: A realtime network emulator," in IEEE Military Communications Conference (MILCOM), pp. 1–7, 2008.

[15] "Android Virtual Device." http://developer.android.com/tools/devices/ index.html.

[16] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, "CRAWDAD data set ncsu/mobilitymodels (v. 2009-07-23)." Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels, July 2009

[17] "Crawdad." http://crawdad.cs.dartmouth.edu/.