Rich Content Dissemination in Hybrid Cellular and 802.11 Ad Hoc Networks

Ngoc Minh Do, *Student Member, IEEE,* Cheng-Hsin Hsu, *Member, IEEE,* and Nalini Venkatasubramanian, *Senior Member, IEEE*

***Abstract*—We design, implement, and evaluate a middleware system, *HybCAST*, that leverages a hybrid cellular and ad hoc network to disseminate rich contents from a source to all mobile devices in a predetermined region. HybCAST targets information dissemination over a range of scenarios (e.g., military operations, crisis alerting, and popular sporting events) in which high reliability and low latency are critical and existing fixed infrastructures such as wired networks, 802.11 access points are heavily loaded or partially destroyed. HybCAST implements a suite of protocols that: (i) structures the hybrid network into a hierarchy of two-level ad hoc clusters for better scalability, (ii) employ both data push and pull mechanisms for high reliability and low latency dissemination of rich content, and (iii) implement a near-optimal gateway selection algorithm to minimize the transmission redundancy. To demonstrate its practicality and efficiency, we have implemented and deployed the HybCAST middleware on several Android smartphones and an in-network Linux machine that acts as a dissemination server. The system is evaluated via real experiments using a UMTS network and extensive packet-level simulations. Our experimental results from a live network show that HybCAST achieves 100% reliability with shorter latencies and lower overall energy consumption. Simulation results confirm that HybCAST outperforms other state-of-the-art systems in the literature. For example, HybCAST exhibits a 5 times reduction in the dissemination latencies as compared to other hybrid dissemination protocols, while its energy consumption is a third of a cellular-only dissemination system. Furthermore, the simulation results demonstrate that HybCAST scales well and maintains good performance under varying numbers of mobile devices, diverse content sizes, and device mobility.**

***Index Terms*—Middleware, dissemination, multimedia, reliability, wireless networks, approximation algorithms**

I. Introduction

The mobile space has witnessed a surge in the amount of rich media being accessed on handheld wireless devices - images, maps, photographs, and video content are all a part of today's mobile experience. Reliable and timely rich content dissemination is becoming increasingly important in various usage scenarios, including military operations, crisis response, online meetings and conferences, and sporting events. For example, tactical networks supporting interactive maps can better support soldiers in battlefields, and emergency networks allowing video sharing can reduce the unnecessary latency due to verbal communications in rescue missions. Traditionally, rich content dissemination in military and emergency scenarios is done over multi-hop ad hoc networks. However, such an approach may not be reliable because ad hoc networks are inherently vulnerable to network segmentations. In contrast, rich content dissemination over cellular networks is more reliable because of the increased communication range of the cellular infrastructure.

While it was believed that cellular base stations may not be available for instant deployment, e.g. in battlefields and crisis scenes, recent technology advances allow cellular base stations to be rapidly deployed. For example, a Qualcomm military grade 3G CDMA base station can be deployed in 15 minutes [1] and a VNL WorldGSM solar-powered base station can be set up in 6 hours [2], making it possible to leverage the cellular infrastructure in such environments for more reliable rich content dissemination. Nonetheless, reliably disseminating rich content over cellular networks is difficult because these base stations often suffer from insufficient capacity. Unicast dissemination of rich content results in tremendous redundancy, and may lead to serious network congestion. Existing cellular deployments are also not suitable for rich content dissemination because they do not provide efficient support for multicast. In fact, most providers only support multicast of control data (90-character limit) which can be used for alarm and warning purposes [3]. Standards for supporting multicast in cellular networks exist, e.g. 3G Multimedia Broadcast Multicast Service (MBMS) [4]; however, adopting these standards requires upgrading the cellular base stations and all mobile devices. This incurs high deployment cost [5], and may not be commercially-viable in the near future. To address the capacity issue, we propose to offload some network traffic from the cellular networks to shorter range ad hoc networks formed by mobile devices. Most mobile devices nowadays come with multiple network interfaces that support ad hoc modes. We use IEEE 802.11 as the second network interface.[1] While each cellular base station covers a much larger area and thus provides always-on connectivity, 802.11 ad hoc networks typically have much shorter ranges, achieve better spectrum reuse, but only offer opportunistic connectivity. We exploit the complementary characteristics of cellular and 802.11 ad hoc networks by concurrently leveraging both networks to efficiently disseminate rich content. Fig. 1 illustrates a *hybrid* cellular and ad hoc network, in which rich contents are first uploaded to a dissemination server, and then disseminated over both cellular and ad hoc networks. We note that concurrently

Work-in-progress. Part of this work is under review for a conference publication.

[1]We acknowledge that 802.11 ad hoc mode is not widely used perhaps because lack of direct supports of service discovery and power saving mode. Recently published WiFi Direct [6] efficiently supports peer-to-peer communications and can be used to replace 802.11 ad hoc mode. Our proposed solution works among WiFi Direct devices as well.

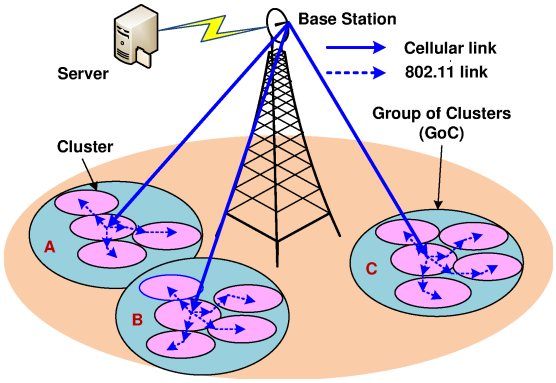

Fig. 1. The Proposed HybCAST architecture.

activating multiple network interfaces may lead to higher energy consumption on mobile devices. However, it is reported that 802.11 networks are more energy efficient for large file dissemination [7]. Through real experiments (Section VI) and extensive simulations (Section VII), we will show that enabling two interfaces for rich content dissemination actually consumes *less* overall energy than using only cellular network interface.

In this paper, we design, implement, and evaluate a middleware system, *HybCAST* for reliable and low latency dissemination. HybCAST addresses the following two key problems: (i) how to determine the dissemination paths among mobile devices, and (ii) how to reliably disseminate rich content over the dissemination paths without incurring excessive latency. HybCAST consists of a *Control Plane* responsible for computing and maintaining dissemination paths and a *Data Plane* responsible for distributing data over these paths.

The main contributions made in the paper are summarized:

1) We design a middleware system deployed at a server and several mobile devices for efficient rich content dissemination. We implement our system on a Linux server and multiple Android smartphones to demonstrate its feasibility. To our best knowledge, this is one of the first few real systems designed for disseminating rich content with high reliability and low latency over hybrid networks.
2) We conduct extensive simulations using Qualnet simulator [8] to study its performance under diverse environments. Our simulation results show that HybCAST enables reliable and low latency dissemination while consuming less energy than a cellular-only system. For example, HybCAST leads to 100% reliability, achieves more than 5 times shorter latency than another dissemination system, and consumes on average one-third of energy compared to cellular-only dissemination system.
3) The proposed system groups mobile devices in hierarchical clusters on ad hoc networks for higher stability toward mobility, and pushes content along the dissemination paths with controlled broadcast. We address the NP-Hard problem of selecting the optimum gateways among clusters by proposing a 1.2773-approximation algorithm in terms of total number of transmissions. This approximation algorithm allows us to control data redundancy in ad hoc networks, and thus reduces interference and bandwidth consumption. To our best knowledge, this is the first polynomial time algorithm with a provable approximation factor for the number of transmissions in cluster based ad hoc networks.

The rest of the paper is organized as follows. Section II describes related work. Section III overviews challenges and the design of the proposed system. We present the system's key components in Sections IV and V. We evaluate its performance in Sections VI and VII. Section VIII concludes the paper.

II. Related Work

Although employing an opportunistic ad hoc network to improve data throughput of a cellular network has been considered in the literature, the earlier studies have different design goal than the present paper. We classify these studies into two categories based on their transport mechanisms over the cellular connections: multicast and unicast.

Multicast based solutions include [9], [10], [11], [12], [13]. Bhatia et al. [12] proposed algorithms to solve the routing problem in hybrid networks. Those algorithms attempt to maximize throughput by finding a set of proxies which are devices receiving data through the multicast cellular connection and responsible for relaying data over the ad hoc network. Lao et al. [10] optimized data throughput by structuring the hybrid network into groups, each receiving data through either the multicast cellular network or the ad hoc network. Law et al. [9] employed the results from Gupta et al.'s work [14] to estimate the capacity of the hybrid network. In their architecture, a base station increases multicast data rate by reducing transmission range. Devices, which receive data from the base station, relay the data to devices which do not receive data due to being out of the base station's range.

While the solutions employing multicast cellular transfer significantly improve the data throughput, multicast in cellular networks may not be deployed in the near future because existing technologies do not support multicast of larger content. In practice, cellular network providers only support multicast short messages that are shorter than 90 characters for emergency situations [3]. Therefore, we do not use multicast as well as compare our work with the multicast based solutions.

Problems addressed by unicast based solutions include effective proxy selection [15], [16], optimal live video streaming [17], and video ads dissemination [18]. In [15], [16], a requesting device which wishes to download data from a base station looks for a proxy which is a device with highest cellular data around it and establish a path from the proxy to itself. Data is unicast from base station to the found proxy and then pushed to the requesting device along the found route. Do et al. [17] proposed scheduling algorithms to optimize live video streaming from a server to a set of devices. Videos are encoded into layers using a novel coding standard H.264/SVC to be scalable with the fluctuations of wireless network capacity. Hanano et al. [18] designed a system for reliable video ads dissemination. Devices interested in video ads register with a server and keep pulling video chunks from the server over the

cellular network or from neighboring devices over the ad hoc network until they get all chunks to recover the ads.

None of the aforementioned studies rigorously address the problem of achieving high reliability and low latency for rich content dissemination over a hybrid cellular and ad hoc network. Mostly, reliable dissemination of rich content has been studied in ad hoc networks. This is however challenging due to the characteristics of such networks, i.e., high levels of interference, short range, and high mobility. These properties generally cause high packet loss ratios, which in turn reduce reliability. Existing efforts either deal with small content dissemination [19], [20] or aim to support rich content dissemination over a small number of nodes [21]. Our work is the first effort combining both cellular and ad hoc networks for efficient and large scale rich content dissemination.

III. Challenges and Solution Approach

A. Challenges

We develop a middleware system that enables high reliability, low latency rich content dissemination over hybrid networks. This is not an easy task due to two main challenges:

- **Large content dissemination**: To deliver rich contents with sizes from hundreds of KB to hundreds of MB, we need to divide each content into multiple chunks whose size is smaller than the maximum allowable size predefined by the underlying communication protocol. A receiver can reconstruct the original content only if *all chunks* of that content *are successfully received*. Packet loss in wireless networks is common due to mobility, interference, collision, and fading. Dividing a content into more chunks, therefore, leads to a higher probability of content delivery failure.
- **Ad hoc network dynamics and interference**: In general, the proposed hybrid system should utilize the ad hoc network to the possible extent to reduce bandwidth consumption on the cellular network. However, managing connection between nodes in ad hoc networks is difficult due to constant mobility and short communication ranges of the ad hoc links. As a result, the dissemination of chunks from one mobile device to another along multi-hop paths frequently fails in high mobility environments. Furthermore, since mobile devices share the same ad hoc spectrum, and are vulnerable to high interference in heavily loaded environments. In short, ad hoc networks often suffer from high packet loss rates and high delivery latencies.

We design HybCAST to tackle these challenges. We provide an overview of HybCAST in the following section.

B. HybCAST System Architecture

HybCAST uses a dissemination server to structure the hybrid network, store and disseminate rich contents to mobile devices. Fig. 1 shows the system architecture, in which a middleware is installed on the server and on every mobile device. Note that, contents may be generated with a workstation on the Internet or a mobile devices on a cellular network. These contents are first uploaded to the server and saved in the Data Storage module. We summarize the terminologies used this paper in Table I.

Fig. 2 presents the system-level components in HybCAST. The Control Plane includes two modules: Metadata Management which performs metadata dissemination to notify mobile devices about an ongoing dissemination, and Network Management which structures the hybrid network and provides the network structure to the Data Plane. We present the details of the Control Plane in Section IV. The Data Plane performs actual content dissemination, and consists of two modules: Push and Pull. The Push component is responsible for fast propagation of chunks from the server to mobile devices. The Pull component is responsible for exchanging missing chunks in the network. Details about the Data Plane are given in Section V.

Fig. 3 shows the steps involved in content dissemination. The initial step is an awareness dissemination where the system disseminates metadata to let devices become aware of an ongoing dissemination and initiate structuring of the network. Next, the system pushes content chunks along paths formed in the structured network. Concurrently, devices pull chunks that are still missing subsequent to content push.

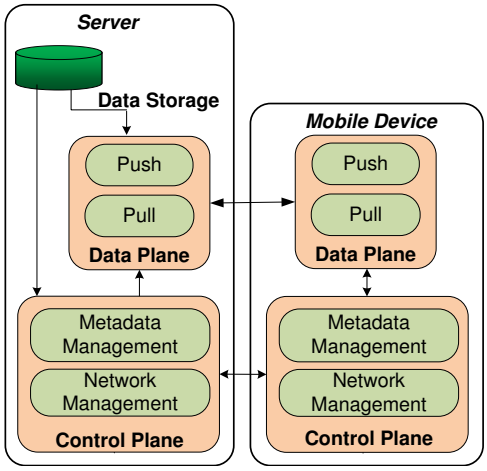

Fig. 2. The software components of HybCAST.

C. Discussions

The Push and Pull components of HybCAST are triggered after a mobile device receives the metadata from the Metadata Management component. In principle, the Pull component can retrieve the entire content using the Pull process once metadata is available; however, this can cause bottlenecks and increased latency in the system. In HybCAST, the Push and Pull components jointly achieve high reliability. In particular, the Push component performs fast pushing of content chunks from the server to mobile devices. This allows us to significantly reduce latency since the server proactively sends chunks to all mobile devices, which in turn will relay the chunks to others.

However, packet losses may occur in the push process. The Pull component mitigates this issue by allowing mobile devices to collect missing chunks. Intelligent clustering of

TABLE I
NOTATIONS USED IN THIS PAPER

Cluster Head (CHD)	A central node in a cluster which connects directly to all other nodes in its cluster (i.e., 1 hop).
Cluster Member (MEM)	A node that belongs to a cluster, but is not a CHD.
Orphan (ORP)	A node that does not belong to any cluster.
Gateway (GW)	A node that belongs to a cluster and has connections to nodes belonging to other clusters.
Group of Clusters (GoC)	Each GoC includes a central cluster and a set of clusters that are neighbors of the central cluster. The central cluster is selected based on a user-specified utility function, such as the cellular data rate.
GoC Head (GoCH)	The CHD of the central cluster in a GoC.

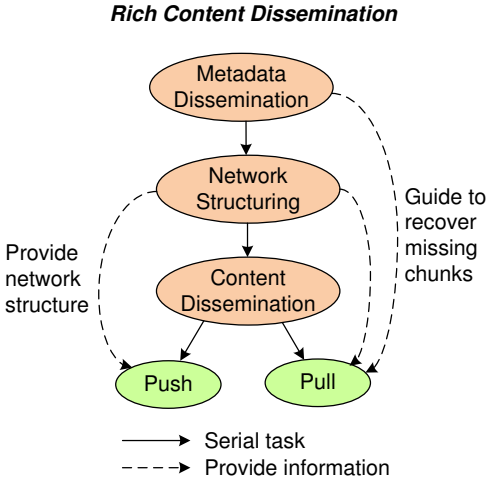

Fig. 3. Content dissemination in HybCAST.

nodes can also play a key role in reducing packet loss, which in turn reduces dissemination latencies. Packet loss is caused by two factors: interference and mobility. To reduce interference, the system needs to reduce redundant transmission. Our Push and Pull components employ the cluster based structure to avoid redundant transmission when pushing or pulling chunks in the hybrid network. Moreover, the cluster based structure is maintained despite changes caused by mobility. Since the connectivity between any two clusters is supported by multiple links across gateways, i.e., clusters are likely to remain connected if there is a link failure between two gateways.

IV. Control Plane

The task of the control plane is to (a) notify mobile devices about an ongoing dissemination and (b) manage the hybrid network by coordinating information exchange between the server and mobile devices. The control plane activities provide information to other components in the Data Plane.

A. Metadata Management

HybCAST employs the Metadata Management component to provide mobile devices information about an ongoing dissemination. Because mobile devices that are interested in a content or a category of contents need to sign up with the server, the server is able to maintain a list of receivers. The server performs a *metadata* dissemination before an actual content distribution. Once a device receives metadata about a content, it is content-aware, and the Pull component at the device can retrieve any or all chunks of the content. The metadata message also works as a request to a mobile device to turn on the 802.11 interface and start forming the 802.11 ad hoc network. Later, the mobile device turns off the 802.11 interface for energy conservation when it receives the entire content and is no longer needed for ad hoc data transfer.

The metadata is a message containing information about the content to be disseminated; it include the content name, content ID, and the number of chunks. The Metadata Management module at the server transmits a metadata message to each mobile device over the cellular network. Every mobile device, upon receiving the metadata, must confirm this to the server. The server resends metadata to the device until a confirmation is received. Notice that the metadata is small, and thus disseminating it to mobile devices consumes much less network resources than an actual rich content distribution does.

B. Network Management

The Network Management at mobile devices deals with the problem of structuring and maintaining the 802.11 ad hoc network. Each mobile device has a distinct ID, and communicates with other mobile devices over the 802.11 channel. To scale with network density, content size, and network mobility, mobile devices are grouped into *clusters*. We connect every two neighboring clusters via multiple links to enable clusters to communicate under device mobility. In each cluster, there is a *cluster head* (*CHD*). A mobile device in the cluster that is not a CHD is referred to as a *member* (*MEM*). A mobile device not belonging to any cluster is called an *orphan* (*ORP*).

When a mobile device receives a metadata message from the server over the cellular network, it enables the 802.11 interface, and sets itself initially as an ORP. Notice that a device powers off its 802.11 interface to save energy after it and all of its neighboring nodes have fully received the disseminated content. If an ORP is in some CHD's range, it will join the cluster of that CHD, and becomes a MEM. If not, it performs a cluster forming procedure. We select CHD based on a *user-specified* utility function $u(\cdot)$. That is, the device with the highest utility value among its ORP neighbors declares itself as a CHD, and immediately broadcasts a HELLO message. Various utility functions are possible, and we use *cellular date rate* as the utility function throughout this paper if not otherwise specified. The intuition is to minimize the cellular traffic load imposed by HybCAST. Surrounding ORP devices, upon receiving this message, become MEMs,

and a cluster is thus formed. If there are two ORP devices with the same utility in their range, the one with the lower ID becomes CHD. Each cluster has a unique cluster ID that is the ID of its CHD. After a CHD stays in its cluster for a period of time, if there exists a MEM whose utility minus an *utility gap* is higher than the utility of the CHD, the CHD exchanges its role with the MEM having the highest utility. The utility gap is to avoid clusters from being restructured frequently. Periodically, a mobile device broadcasts a HELLO message to notify its presence to neighbors. The HELLO message contains information such as the sender's ID, cluster ID, and the IDs of the sender's neighboring clusters. In this way, CHDs exchange information about their neighboring clusters. To shorten the latency of structuring the hybrid network, when devices forming clusters, mobile devices broadcast HELLO messages more frequently than after they join in clusters.

A CHD communicates cluster related information with the Network Management Module at the server - this information includes the cluster ID, current neighboring cluster IDs (includes updates when new neighbors are discovered or a neighboring cluster is disconnected). Thus, the server learns a global map of the 802.11 ad hoc network: segments and clusters in segments. For each segment, the Network Management at the server groups clusters into Groups of Clusters - GoCs. The CHD of the central cluster is referred to as *GoC Head* (*GoCH*). Fig. 4 shows the details of GoC A in Fig. 1. GoC A includes cluster 1 as the central cluster, and neighboring clusters 2, 4, and 5. Clusters in a segment is grouped by the server based on their utility, and the server informs GoCHs about clusters available in their GoCs.

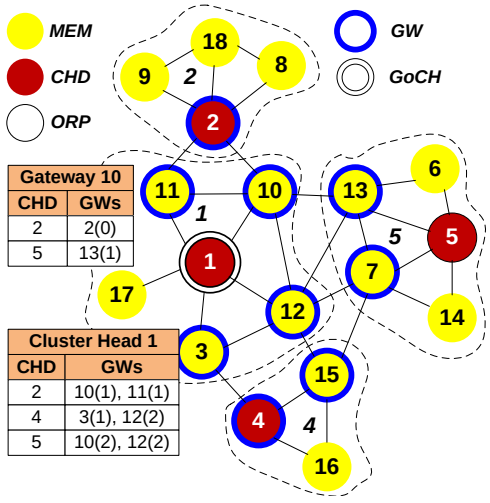

Fig. 4. Structure of a GoC in HybCAST.

We denote a mobile device to be a *gateway* (*GW*) if it connects to other clusters. By exchanging HELLO messages, the Network Management at CHD or GW is able to maintain a data structure, a *management table*, which is useful for pushing chunks from a cluster to its neighboring cluster. A management table entry for each CHD contains: (i) a neighboring cluster, (ii) a gateway leading to that cluster, and (iii) the hop count from the gateway to that cluster. For instance, in Fig. 4, CHD 1's management table has three records. The first record <2, {10(1), 11(1)}> indicates that CHD 1 is able to forward packets to cluster 2 through GW 10 or GW 11 which is one hop away from CHD 2.

C. Network Structuring Policy

HybCAST employs the cellular rate as the utility function by default. That is, a device with the highest data rate will be chosen as a CHD. However, doing so might quickly drain the CHDs' batteries because they not only receive data over the cellular network but also relay the data to MEMs over the 802.11 ad hoc network.

Service providers may address this issue by structuring the hybrid network using a composite utility function of the energy consumption and cellular data rate, such as: $u(E_r, C_r) = \sqrt{w_e(\frac{E_r}{E_{max}})^2 + w_r(\frac{C_r}{C_{max}})^2}$, where E_r and E_{max} are the residual energy and the maximum energy of the battery, and C_r and C_{max} are cellular data rate and the maximum rate of the cellular network, w_e and w_r are weights for energy and data rate such that $w_e + w_r = 1$. Note that, with $w_r = 1$, this structuring policy degrades to cellular rate based CHD and GoCH selections. With $w_e = 1$ this structuring policy becomes battery-biased (and rate-agnostic) CHD and GoCH selections.

V. Data Plane

A. Content Push

The Push component in the Data Plane at the server initiates the dissemination process by querying the Control Plane to get information about GoCs in the 802.11 ad hoc network. For each GoC, the Push component at the server unicasts chunks to the corresponding GoCH. Upon receiving chunks from the server, the GoCH employs its Push component to relay the chunks to all neighboring CHDs in the GoC using the 802.11 network. To do so, a GoCH has to choose a set of gateways for pushing the chunks to the neighboring CHDs. The algorithm of gateway selection is presented in Section V-B. Once gateways are selected, paths toward the neighboring CHDs are attached as routing information into the chunks, and the chunks are broadcast to the gateways and other MEMs. The gateways further forward the chunks to the neighboring CHDs by looking up their management tables. When the neighboring CHDs receive the chunks, they broadcast the chunks to their MEMs. Thus, all nodes in each GoC will receive the chunks.

Fig. 4 gives an illustrative example. A content's chunks are pushed by Push component at the server to CHD 1, whose Push component then relays the chunks to CHDs 2, 4, and 5 over the 802.11 ad hoc network. CHD 1 selects a set of gateways (such as GW 10 for CHDs 2 and 5, and GW 3 for CHD 4), attaches information of the gateways into the chunks, and broadcasts the chunks to gateways. When CHD 1 broadcasts, MEMs in cluster 1 including devices 3, 10, 11, 12, 17 will receive the chunks. The gateways forward the chunks to the neighboring CHDs with next hops selected from their management tables. For example, GW 10 picks up GW 13 to

forward the chunks to CHD 5 with the minimum cost, and uses CHD 2 for forwarding chunks to MEMs in CHD 2's cluster.

B. GoC Gateway Selection Problem (GS) for Content Push

Upon receiving a chunk from the server, a GoCH employs the Push component to select next hops (gateways) to further push the chunk to the neighboring CHDs in its GoC. The distance from the GoCH to its neighboring CHDs is at most three hops. In this section, we study the problem of gateway selection (GS) to minimize the number of transmissions. The number of transmissions is defined as the number of broadcasts to deliver a chunk to all nodes in a GoC. Minimizing the number of transmissions for pushing content in GoC reduces transmission redundancy, which leads to less interference, lower bandwidth consumption, and lower energy consumption in the 802.11 network.

We model the GS problem as follows. We construct a tree including a root that represents the GoCH, in which level 1 consists of a set of GWs directly connecting to the GoCH, and level 2 consists of all neighboring CHDs in that GoC. Each edge is assigned a weight w. If the edge is chosen, w transmissions will be incurred, excluding the transmission at the source. For instance, let's refer to Fig. 5. The weight of the edge 12-4 (from node 12 to node 4) is 2 since there are two transmissions performed by nodes 15 and 4, not counting node 12's transmission. As shown in this figure, the root has directed edges of weight 1 to nodes in level 1, and nodes in level 1 have directed edges of weight 1 or 2 to nodes in level 2. The constructed tree is referred to as a GS tree $T(r, l_1, l_2)$ where r is the root, l_1 and l_2 indicate the sets of nodes in levels 1 and 2, respectively. Using the tree, we convert the GS problem into another problem of finding a subtree in T that connects r to all nodes in l_2 through a subset of nodes in l_1 with a minimum cost, where the cost is defined as the total number of transmissions.

Theorem 1 (Hardness): Given a GS tree $T(r, l_1, l_2)$ and a positive value C, determine if there is a subtree rooted at r and spanning to all nodes in l_2 such that the total cost of the subtree is no greater than C is NP-Hard.

Proof: We show GS problem is NP-Hard by reducing a common NP-Complete problem, *Set Cover (SC)* [22], to it. Given a universe U of elements, a collection $S = \{S_1, S_2, \ldots, S_m\}$ of subsets of U, and an integer k, the problem SC is to find a collection of k sets in S whose union is equal to U. Given a SC instance (U, S, k), we construct a GS instance as follows: Create a root vertex r, a set vertex $v(s)$ for each set s in S, and an element vertex $v(u)$ for each element u in U. Add weight 1 directed edges from r to each $v(s)$, and add weight 1 directed edges from each $v(s)$ to each $v(u)$ if u is in S. It is easy to see the SC instance has a cost of k if and only if the GS instance has a cost of $k + |U|$. The reduction can clearly be done in polynomial time. ■

Since the GS problem is NP-Hard, we formulate it as an Integer Linear Programming (ILP) problem. We define $x_{i,j}$ to be a decision variable: $x_{i,j} = 1$ if the edge connecting node i to node j is selected; $x_{i,j} = 0$ if otherwise. Let $w_{i,j}$ be the weight of the edge connecting i to j. The formulation is written as:

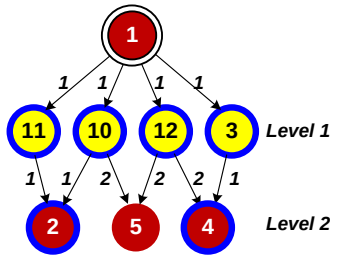

Fig. 5. The GS tree constructed from the GoC shown in Fig. 4.

$$\text{Minimize:} \quad \sum w_{i,j} \cdot x_{i,j}; \tag{1a}$$

$$\text{Subject to:} \quad \sum_{i \in l_1} x_{i,j} \leq 1 \ \forall j \in l_2; \tag{1b}$$

$$\sum_{i \in l_1, j \in l_2} x_{i,j} \geq |l_2| \ ; \tag{1c}$$

$$x_{i,j} \leq x_{r,i} \ \forall i \in l_1, \forall j \in l_2. \tag{1d}$$

The objective function in Eq. (1a) minimizes the cost of the obtained tree. The constraint in Eq. (1b) ensures that at most one edge enters a node in l_2. Eq. (1c) guarantees that all nodes in l_2 must be picked up in the obtained tree. Finally, Eq. (1d) ensures that if the edge directed from a node in l_1 is selected in the obtained tree, then the edge directed to that node must be selected.

Solving the ILP problem may take a prohibitively long time due to intensive computations. Therefore, we develop an efficient algorithm for the problem, which achieves a tight approximation factor in the following. Our approximation algorithm is based on GS tree's three properties: (i) the tree has only two levels; (ii) in the tree, the weights of edges are either 1 or 2; and (iii) in any solution, the sum of the weights of edges into nodes in l_2 is equal to or greater than that of the weights of edges out of the root.

Assume we have a GS tree T, we make the following observations about the tree.

Lemma 1: Another tree, T', is constructed from T by removing all weight 2 edges ingoing to a node in l_2 if there is at least a weight 1 edge ingoing to this node. If there is a subtree that is an optimal (OPT) solution found in T', it is also OPT in T.

Proof: Assuming that an OPT solution in T has a node u at l_2 attached to the solution by a weight 2 edge, e_2, and in T there exists another ingoing edge with weight of 1 to u, e_1. We can replace e_2 by e_1, and add the edge that connects the root to one end of e_1 located at l_1. This produces another OPT solution. ■

Lemma 2: In T', if a node in l_1 is selected by some approximation algorithm, then all of its children can be safely picked up.

Proof: In T', if a node u in l_2 has more than one ingoing edge, then all edges ingoing to this node must have the same weight (i.e., all of them have a weight of 1 or 2). So, if we decide to select a node in l_1, we can pick up all of its children. ■

Algorithm 1 The pseudo-code of the GS algorithm

```
// Input tree T_in(r, l_1, l_2) - Output tree T_out(r, l_1, l_2)
for each node i ∈ T_in.l_2 do
    if there exists a weight 1 edge ingoing to i then
        remove all weight 2 edges ingoing to i;
    end if
end for
while T_in.l_2 is not empty do
    find node S in T_in.l_1 with the maximum number of children;
    T_out.l_1 = T_out.l_1 + S; T_out.l_2 = T_out.l_2 + children(S);
    T_in.l_1 = T_in.l_1 − S; T_in.l_2 = T_in.l_2 − children(S);
end while
```

Proposed GS algorithm: From the above lemmas, we can find our approximated solution based on T', rather than T. The approximation algorithm works *greedily*: In each iteration, it picks up a node in level 1 with the maximum number of children in the current instance of the tree T', and removes it and all of its children from the current instance until there exists no node in l_2. We give the pseudocode in Algorithm 1.

For instance, T is the tree given in Fig. 5. By following the proposed GS algorithm, T' is created from T by removing the edge 12-4. The algorithm terminates in two iterations. The first iteration picks up node 10 and all of its children (2 and 5). Nodes 3 and 4 are picked up in the next iteration. The algorithm terminates since all nodes in level 2 are picked up.

Theorem 2 (Approximation factor): The GS algorithm finds a tree with the cost at most 1.2773 times the cost of the optimum tree.

Proof: We use a charge scheme to prove the theorem. Let c_i be the cost when node i in l_2 is picked up to an output tree T_{out} generated by the GS algorithm. The cost of T_{out} is the sum of the cost paid to pick up all leaves in l_2. Cost c_i includes cost u_i paid for picking up i's parent, and cost d_i paid for picking up the ingoing edge to node i. That is:

$$c_i = u_i + d_i. \tag{2}$$

Similar to the proof of Lemma 2, all ingoing edges to node i have the same weight, d_i.

Consider any node s in l_1 in some *OPT* tree T_{opt}. Node s has a set H_s including k children $\{h_{s,k}, h_{s,k-1}, ..., h_{s,1}\}$. Without loss of generality, the GS algorithm selects the children of s in the order of $\{h_{s,k}, h_{s,k-1}, ..., h_{s,1}\}$. At an iteration, a child $h_{s,i}$ in H_s is selected. The cost $u_{h_{s,i}}$ of $h_{s,i}$ is at most $\frac{w_{r,s}}{i}$ where $w_{r,s}$ is the weight of the edge from root r directed to node s. Since $w_{r,s}$ is 1, $u_{h_{s,i}}$ is at most $\frac{1}{i}$. So, the cost to pay for choosing s and all of its children is:

$$c_s = \sum_{h \in H_s} u_h + \sum_{h \in H_s} d_h, \tag{3}$$

where

$$\sum_{h \in H_s} u_h \leq 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{|H_s|} \leq 1 + \ln(|H_s|). \tag{4}$$

Thus, the approximation factor f of the GS algorithm is:

$$f = \frac{Cost_{T_{out}}}{Cost_{T_{opt}}} = \frac{\sum_{i=1}^{n}(\sum_{h \in H_{s_i}} u_h + \sum_{h \in H_{s_i}} d_h)}{\sum_{i=1}^{n}(1 + \sum_{h \in H_{s_i}} d_h)}, \tag{5}$$

where n is the number of nodes in l_1 of T_{opt}, and s_i is a node in l_1 of T_{opt}.

For any node s_i $(1 \leq i \leq n)$ belonging to l_1 of T_{opt}, an f^* is the upper bound of f in Eq. (5), if

$$\frac{\sum_{h \in H_{s_i}} u_h + \sum_{h \in H_{s_i}} d_h}{1 + \sum_{h \in H_{s_i}} d_h} \leq f^* \tag{6}$$

holds.

By Eq. (4), it is obvious that

$$\sum_{h \in H_{s_i}} u_h \leq 1 + \ln(|H_{s_i}|) \leq 1 + \ln(\sum_{h \in H_{s_i}} d_h). \tag{7}$$

Thus,

$$\frac{\sum_{h \in H_{s_i}} u_h + \sum_{h \in H_{s_i}} d_h}{1 + \sum_{h \in H_{s_i}} d_h} \leq \frac{1 + \ln(\sum_{h \in H_{s_i}} d_h) + \sum_{h \in H_{s_i}} d_h}{1 + \sum_{h \in H_{s_i}} d_h} \tag{8}$$

Let's consider function $g(t) = \frac{1+\ln(t)+t}{1+t}$ where $t = \sum_{j \in S_i} d_j$ $(t \geq 1)$. Its maximum value is 1.2773 at $t = 4$. Thus, if $f^* = 1.2773$, then Eq. (6) is satisfied for any node in l_1 of T_{opt}. That is, the GS algorithm generates a tree with the cost at most 1.2773 times the cost of the OPT tree. ■

Last, we notice that our GS problem is a special case of another NP-Complete problem called *Directed Steiner Tree (DST)* [23]. The general DST problem has been considered in literature before, with looser approximation factors. For example, Takahashi and Matsuyama [24] propose an algorithm to achieve an approximation factor of 2 for *undirected* Steiner Tree, but only achieves a guarantee of $O(k)$ for the *directed* version, where k is the number of nodes. Another algorithm proposed by Charikar et al. [25] achieves a factor of $l(l-1)k^{1/l}$, where l is the number of levels and k is the number of nodes. Our algorithm is partially inspired by these algorithms solving the general DST problem.

C. Pull Missing Chunks

The Pull component aims to collect missing chunks for mobile devices. The Pull component at mobile devices periodically informs its packet reception progress to its neighbors by broadcasting a GAP message consisting of missing packets encoded as missing gaps using the 802.11 ad hoc network. A gap $\{p_1, p_2\}$ indicates that the sender does not have packets from p_1 to p_2. The Pull component at CHDs works as coordinators to control missing chunk exchanges. CHDs request missing chunks by broadcasting a PULL message that contains a list of records, indicating the requested chunks and neighbors that already hold the requested chunks. Upon receiving the pull message, a neighbor transmits a requested chunk with a random delay to the requesting CHD to avoid bottleneck at this CHD. The 802.11 ad hoc network should be given a higher priority to save bandwidth for the cellular network. That is, so far collecting and providing missing chunks at CHDs have been performed in the 802.11 ad hoc network.

The pull mechanism described above works efficiently for exchanging missing packets within a cluster, and between two clusters if the shortest path between two CHDs is less than three hops. For chunks that are not delivered by above mechanisms, the Pull component at CHDs requests these chunks over the cellular network. More specifically, the Pull component at CHDs informs missing chunk gaps to the Pull component at the server. Once receiving missing chunks from the server, CHDs continue providing them to their neighbors. However, the Pull component at a CHD contacts the Pull component at the server only when (i) the CHD has all chunks that its neighboring nodes have, (ii) CHD's neighbors have all chunks owned by the CHD, and (iii) the CHD still has missing chunks. These conditions are to save the cellular network bandwidth for other purposes.

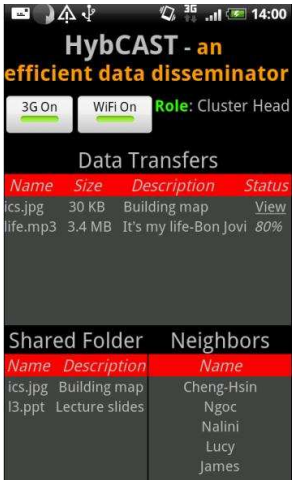

Fig. 6. A screenshot of the HybCAST client.

VI. Implementation: A Proof-of-Concept

We have implemented HybCAST in C and Java to demonstrate its practicality and efficiency. The server consists of 1,000 lines of C code, and is deployed on a Linux workstation with 2 GHz CPU and 1 GB memory. The client consists of 2,000 lines of C and Java code and is deployed on five HTC Android phones with T-Mobile UMTS access. Fig. 6 shows a screenshot of our client. HybCAST requires us to concurrently enable 3G and 802.11 interfaces, which is not supported by Vanilla Android. Therefore, we adopt WiFi Tether module [26] to enable 802.11 ad hoc mode manually. We compare the HybCAST system against a baseline system, in which only 3G unicast is used for data dissemination. In each experiment, we disseminate a data file with a size between 64 KB and 2 MB to all smartphones. We repeat each experiment 6 times and report the results from a representative run. Across all experiments, the average 3G rate is measured at 130 kbps, and the average 802.11 rate is about 3.5 Mbps. Three performance metrics are considered: (i) *reliability*, which is the fraction of smartphones that receive the complete data file, (ii) *latency*, which is the time difference when the system starts disseminating a data file and all smartphones get that file, and (iii) *battery level*, which is the residual battery capacity reported by Android's battery manager.

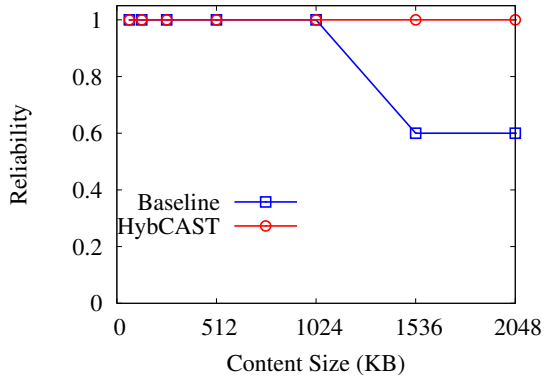

Fig. 7. Reliability - HybCAST implementation vs. baseline system.

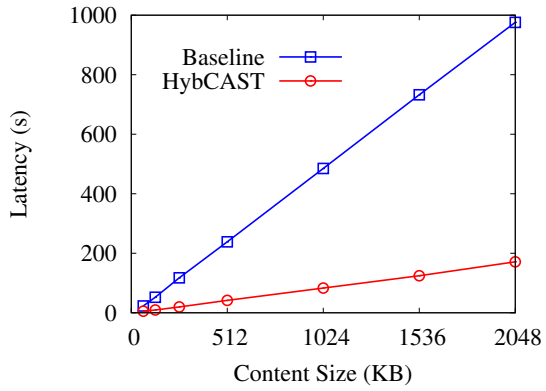

Fig. 8. Latency - HybCAST implementation vs. baseline system.

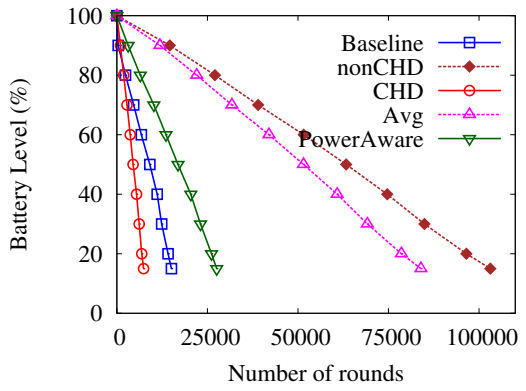

Fig. 9. Battery Life - HybCAST implementation vs. baseline system.

Reliability: Our results (Fig. 7) show that HybCAST achieves 100% reliability under all data sizes, while the baseline system exhibits only 60% reliability, i.e., only 3 out of 5 smartphones get the file, when data file is larger than 1.5 MB.

Latency: Fig. 8 reports the latency. The figure shows that the baseline system suffers from a much longer latency, about 7 times longer than HybCAST. We note that the latency of HybCAST already included the time of enabling the 802.11 interface upon receiving metadata message over the 3G network.

Energy consumption: We conduct the following experiments to study the implications of HybCAST on battery life. We fully charge the smartphones, and repeatedly disseminate a 1 MB data file until the battery level is below 15%. We report battery levels measured from the same smartphone, as batteries have diverse discharge patterns. We set the screen brightness at 50%. We record the battery level after each dissemination *round*; in each round a data file is fully disseminated to all smartphones. We consider the baseline system, and divide smartphones into CHD and nonCHD when using HybCAST for dissemination. We present the results in Fig. 9. This figure shows that the battery of a HybCAST nonCHD lasts 6.84 times longer than that of a baseline system. However, we notice that a HybCAST CHD has an inferior battery life, about 49% of the baseline system, which is attributed to the 802.11 power consumption. Nevertheless, the average energy consumption across all five smartphones is much lower than the baseline system. We also employ the battery-biased network structuring policy described in Section IV-C, and we denote its CHDs' energy consumption as PowerAware in Fig. 9. This figure shows that, with PowerAware, even CHDs results in 1.84 times longer battery life than the baseline system. This alleviates the concerns of additional energy consumption caused by enabling the 802.11 interface.

VII. Simulation Based Evaluation

While the proof-of-concept implementation (Section VI) shows the practicality of HybCAST, we acknowledge that the experimental results are limited by the number of smartphones. To cope with this limitation, we conduct large-scale simulations to evaluate HybCAST under diverse network conditions.

A. Settings

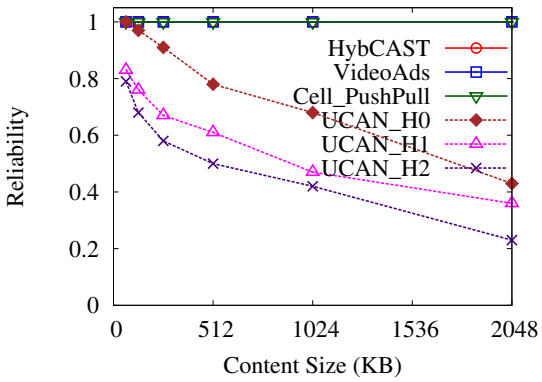

Fig. 10. HybCAST achieves 100% reliability for rich data dissemination.

We have implemented HybCAST in a well known packet-level simulator: Qualnet 5.0 [8]. In all simulations, mobile devices employ WiMAX to establish connections to a base station, and use IEEE 802.11 to form ad hoc networks. WiMAX supports a peak raw data rate of 14.4 Mbps, while the maximum data rate of 802.11 is 36 Mbps. We configure the WiMAX range to cover all mobile devices, and 802.11 range to be 200 m. We employ the Two-Ray propagation model in our simulations. Mobile devices are randomly placed in a 1000 by 1000 m^2 terrain. We use the Random Waypoint mobility model to generate device mobility. Maximum speed is 5 m/s with 30 second pause time, unless otherwise specified. UDP is used as the transport protocol.

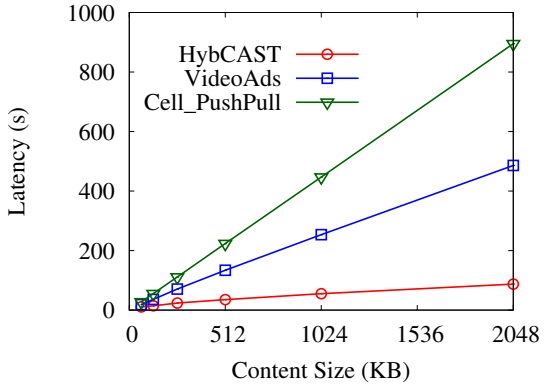

Fig. 11. HybCAST results shortest latency with any content size.

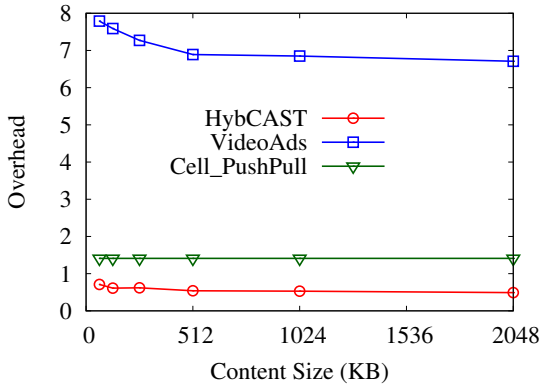

Fig. 12. HybCAST incurs low overhead with any content size.

We adopt four performance metrics in this evaluation: (i) *reliability*, (ii) *latency* (reliability and latency are already defined in Section VI), (iii) *overhead*, which is the ratio of the average number of bytes transmitted per node to the content size, and (iv) *energy consumption*, which is the average energy consumed by mobile devices. We present four sets of simulations to study how the system performs under different conditions of: (i) mobile device density, (ii) content size, (iii) mobile device mobility, and (iv) network structuring policy.

We compare HybCAST against two state-of-the-art systems: a push based system [15], [16] and a pull based system [18]. Lou et al. propose two algorithms in [15], [16]: on-demand and greedy. Because the greedy algorithm provides higher throughput, we implemented it for comparisons, in which a

mobile device floods a route request (RTREQ) over the 802.11 ad hoc network within H hops to discover an ad hoc route to the best proxy around it. The best proxy has the highest 3G data rate among the nearby mobile devices. The base station pushes data to the proxy which then relays data along the ad hoc route to the requester. Following their system's name, we refer the greedy algorithm with $H = 0$, 1, and 2 as UCAN_H0, UCAN_H1, and UCAN_H2 respectively in the figures. Note that UCAN_H0 means the server simply pushes data to mobile devices over the cellular network. This is in fact the baseline system in our experiments. Hanano et al. [18] design a pull based system in which mobile devices keep pulling chunks either from the server over the cellular network or from neighboring mobile devices over the ad hoc network. The chunks pulled by mobile devices are determined based on the probability they have been received by neighboring mobile devices. We refer to this system as VideoAds, based on the original application type. We also implemented a simplified system in which the server at first pushes data to mobile devices, and then retransmits missing chunks upon requests from devices over the cellular network. It is denoted as $Cell\_PushPull$ in the figures. We do not consider systems that require multicast-capable cellular base stations [12], [13] for comparisons, because existing base stations only support short multicast messages for alarm and warning purposes.

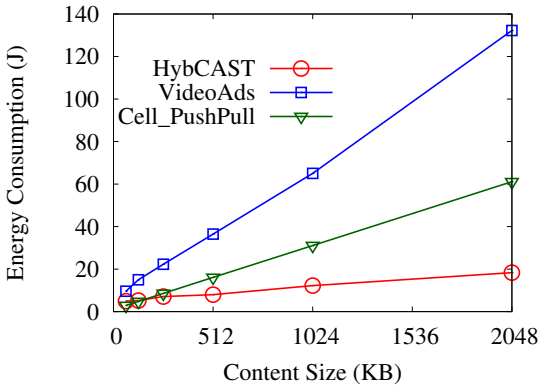

Fig. 13. HybCAST conserves energy even two interfaces enabled.

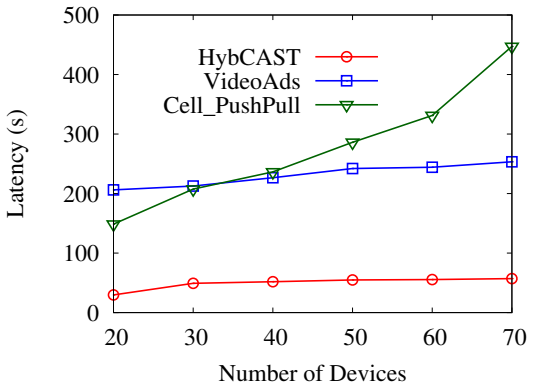

Fig. 14. HybCAST achieves low latency with diverse number of devices.

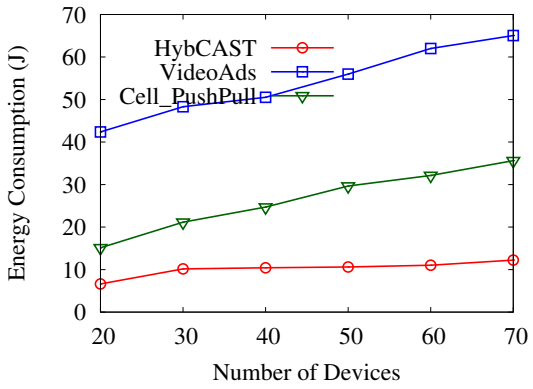

Fig. 15. HybCAST is energy efficient even with many devices.

B. Experimental Results

Impact of content size (Figs. 10, 11, 12 and 13): We vary the content size from 64 KB to 2 MB in 70-device networks to investigate the performance of the various systems. Fig. 10 shows that HybCAST, VideoAds, and Cell_PushPull achieve 100% reliability in all simulations, while UCAN achieves 100% reliability only when the content size is 64 KB and $H = 0$. This shows the importance of the Pull component. This figure also reveals that UCAN experiences a faster drop of reliability when H increases. This is because the 802.11 network suffers from a high packet loss ratio due to interference and mobility, and longer propagation paths amplify the problem. Since UCAN is not reliable for large content dissemination, we no longer consider it in the rest of this section.

We plot the latency in Fig. 11. This figure shows that HybCAST takes less than 1 minute to disseminate a 1 MB content to 70 mobile devices, which is about 10 times faster than Cell_PushPull and about 5 times faster than VideoAds. Fig. 12 shows that VideoAds results in 10 times higher overhead than HybCAST. This is because VideoAds is a pull-only system. In HybCAST, contents are first efficiently pushed to all mobile devices in a GoC only over CHDs and selected GWs and thus incurs lower overhead. Besides, HybCAST realizes a more efficient pull mechanism than VideoAds since missing chunks are exchanged under control of CHDs, which generates less overhead, consumes less uplink bandwidth, and reduces redundancy. Fig. 13 shows that HybCAST consumes the least energy. This is because faster dissemination and lower overhead allow mobile devices to put their network interfaces into sleep more often for higher energy saving.

Impact of mobile device density (Figs. 14 and 15): We vary the number of devices in the network from 20 to 70, and disseminate a 1 MB content. All simulations achieve 100% reliability. We plot the latency in Fig. 14, which shows that Cell_PushPull incurs long latency up to 8 minutes to deliver the content to 70 devices. VideoAds achieves a better latency than Cell_PushPull only in dense networks. In contrast, HybCAST constantly achieves the shortest latency: less than 1 minute. Our results show that VideoAds suffers from a significantly higher overhead than the other two systems, which

is consistent with Fig. 12. Fig. 15 shows that Cell_PushPull consumes up to 3 times more energy than HybCAST, and VideoAds consumes up to 6 times. This figure is consistent with Fig. 13.

Impact of mobility (Fig. 16): We vary the maximum speed from 3 to 18 m/s and disseminate a 1 MB content. We investigate the performance of HybCAST with 30-, 50-, and 70-device networks. All simulations achieve 100% reliability. Fig. 16 reports the achieved latency as the speed changes. This figure shows that HybCAST is scalable against node speed: increasing speed by six folds from 3 to 18 m/s only increases latency by about 10 seconds, which is merely a 10% increase. This is because connectivity between clusters is well maintained against mobility due to the existence of multiple links between them.

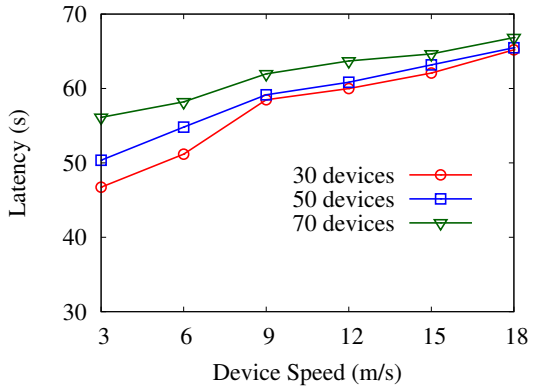

Fig. 16. HybCAST is scalable against node mobility.

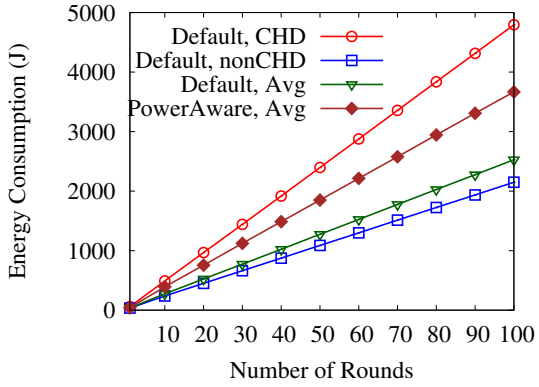

Fig. 17. Energy consumption with different policies for 100 dissemination rounds of a 5 MB content.

Energy consumption at CHDs and Policy (Fig. 17): HybCAST's default policy uses the cellular data rate to select CHDs (i.e., $w_e = 0, w_r = 1$). This may lead to faster battery draining of CHDs than that of nonCHDs because CHDs not only receive data over the cellular network but also relay the data over the 802.11 ad hoc network. In this section, we study the impact of a power aware policy for CHD selection (i.e., $w_e = 1, w_r = 0$), which is denoted as PowerAware in the figures. We perform 100 dissemination rounds in a static 70-device network. In each round, the server disseminates a 5 MB content. Fig. 17 compares two policies: default and PowerAware. With PowerAware, mobile devices take turns to be CHDs and thus consume roughly the same amount of energy: 3,667 J for 100 dissemination rounds (i.e., 500 MB). With the default policy, a device on average consumes less energy: only 2,153 J. This is because when CHDs are selected with high cellular date rate, dissemination terminates earlier, which leads to lower overall energy consumption. In particular, the default policy takes only 174 seconds on average to deliver a 5M content to 70 nodes, while PowerAware takes 253 seconds. Last, with the default policy, a CHD consumes up to 4,377 J, which is approximately 2.5 times higher than that of nonCHD. This illustrates a clear tradeoff between overall energy efficiency and fairness of energy consumption resulted by different policies.

VIII. Conclusion

In this paper, we studied the problem of efficient dissemination of rich content over hybrid wireless networks. We proposed HybCAST, which: (i) structures a hybrid network using a multilevel clustering approach and (ii) utilizes the clustering structure to achieve low latency, high reliability, low overhead dissemination of rich content. Our proof-of-concept implementation and extensive simulation results show that HybCAST always achieves high reliability. The clustering structure is stable under diverse device mobility, e.g., increasing device speed from 3 to 18 m/s only results in a 10% increase on latency. HybCAST leverages on both Push and Pull mechanisms to control overhead as the device density increases, contributes to lower packet loss ratios, and leads to shorter latencies since fewer missing chunks need to be recovered. Furthermore, HybCAST is also shown to consume less energy than cellular-only dissemination. We anticipate that the ability to use multiple networks in a plug-and-play manner for content dissemination will allow us to better meet the diverse needs of future generations of applications. This paper is an initial effort in that direction.

References

[1] "Qualcomm deployable base station, 3G secure CDMA cellular system," http://www.qualcomm.com/common/documents/brochures/QDBS_Cellular_new.pdf, March 2009.
[2] "VNL at Mobile World Congress 2009," http://www.vnl.in/blog/2009/vnl-at-mwc-2009/, March 2009.
[3] E. Stanley and J. Sutton, "Public response to alerts and warnings on mobile devices," *The National Academics*, April 2011.
[4] S. Parkvall, E. Englund, M. Lundevall, and J. Torsner, "Evolving 3G mobile systems: Broadband and broadcast services in WCDMA," *IEEE Communications Magazine*, vol. 44, no. 2, pp. 30–36, February 2006.
[5] Private communication with a leading European cellular service provider, October 2010.
[6] "Wi-Fi certified Wi-Fi Direct: Personal, portable Wi-Fi that goes with you anywhere, any time," http://www.wi-fi.org/Wi-Fi_Direct.php, August 2010.
[7] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. of ACM IMC*, Chicago, IL, November 2009, pp. 280–293.
[8] "Qualnet network simulator: *http://www.scalable-networks.com*."
[9] L. Law, K. Pelechrinis, S. Krishnamurthy, and M. Faloutsos, "Downlink capacity of hybrid cellular ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 243–256, February 2010.

[10] L. Lao and J. Cui, "Reducing multicast traffic load for cellular networks using ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 3, pp. 317–329, May 2006.

[11] J. Park and S. Kasera, "Enhancing cellular multicast performance using ad hoc networks," in *Proc. of IEEE WCNC*, New Orleans, LA, March 2005, pp. 2175–2181.

[12] R. Bhatia, L. Li, H. Luo, and R. Ramjee, "ICAM: Integrated cellular and ad hoc multicast," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 1004–1015, August 2006.

[13] S. Hua, Y. Guo, Y. Liu, H. Liu, and S. Panwar, "Scalable video multicast in hybrid 3G/ad-hoc networks," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 402–413, April 2011.

[14] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, March 2000.

[15] H. Luo, R. Ramjee, P. Sinha, L. Li, and S. Lu, "UCAN: a unified cellular and ad-hoc network architecture," in *Proc. of ACM MOBICOM*, San Diego, CA, September 2003, pp. 353–367.

[16] H. Luo, X. Meng, R. Ramjee, P. Sinha, and L. Li, "The design and evaluation of unified cellular and ad-hoc networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1060–1074, September 2007.

[17] N. Do, C. Hsu, S. Jatinder, and N. Venkatasubramanian, "Massive live video distribution over hybrid cellular and ad hoc networks," in *Proc. of IEEE WoWMoM*, Lucia, Italy, June 2011.

[18] H. Hanano, Y. Murata, N. Shibata, K. Yasumoto, and M. Ito, "Video ads dissemination through WiFi-cellular hybrid networks," in *Proc. of IEEE PerCom*, Galveston, TX, March 2009, pp. 1 – 6.

[19] W. Lou, "Double-covered broadcast (DCB): A simple reliable broadcast algorithm in MANETs," in *Proc. of IEEE INFOCOM*, Las Vegas, NV, March 2004, pp. 2084–2095.

[20] B. Xing, M. Deshpande, N. Venkatasubramanian, and S. Mehrotra, "Towards reliable application data broadcast in wireless ad hoc networks," in *Proc. of IEEE WCNC*, Las Vegas, NV, March 2008, pp. 12 591–2596.

[21] B. Xing, S. Mehrotra, and N. Venkatasubramanian, "RADCast: Enabling reliability guarantees for content dissemination in ad hoc networks," in *Proc. of IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009, pp. 1998–2006.

[22] M. Garey and D. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," *W. H. Freeman & Co.*, 1979.

[23] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 558–568, August 1996.

[24] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner tree problem in graphs," *Math. Japonica*, vol. 24, pp. 573–577, 1980.

[25] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed Steiner problem," San Francisco, CA, January 1998, pp. 192–200.

[26] "Android WiFi tether: *http://code.google.com/p/android-wifi-tether/*."