Enhancing Opportunistic Networks with Legacy Nodes

Marcin Nagy
Aalto University, Comnet, Finland
marcin.nagy@aalto.fi

Teemu Kärkkäinen
Aalto University, Comnet, Finland
teemu.karkkainen@aalto.fi

Jörg Ott
Aalto University, Comnet, Finland
jorg.ott@aalto.fi

ABSTRACT

Mobile opportunistic networking utilizes device-to-device communication to provide messaging and content sharing mechanisms between mobile users without the need for supporting infrastructure networks. However, enabling opportunistic networking in practice requires a sufficient number of users to download, install, and run the respective routing and application software to provide sufficient node density, and thus connectivity for the network to actually function. In this paper, we explore reaching out to nodes that have not (yet) installed any dedicated software to: (1) allow them to access public content in an opportunistic network to possibly seed their interest and (2) instrument them to assist as (limited) message carriers to improve connectivity. We report on our system design and implementation and offer performance insights gained from simulations and initial experiments.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Store and forward networks*

Keywords

DTN; opportunistic networking; HTTP; web storage; cookies

1. INTRODUCTION

In mobile opportunistic networks, two or more devices establish communication links between each other, as they come into reach of their short-range radios, such as WLAN or Bluetooth. These *contacts* may occur ad-hoc and, during the contact, the devices exchange messages they have stored following the rules of some opportunistic routing protocol [2]. When users and devices move, they *carry* their stored messages until the next contact: messages are moved hop-by-hop from the originator to their destination(s) and may be stored for extended periods of time along the way, so that usually no "instant" end-to-end path, and feedback loop exists. Routing protocols govern whether: (1) a single copy of a message is forwarded in the system, (2) a (usually small) finite number of message replicas are created (controlled replication), (3) no limit on the number of message copies is imposed (flooding). Messages are usually tagged with an expiry time ("time-to-live"), so that they do not clog the network when they are no longer considered useful. The performance of an opportunistic ad-hoc network obviously depends on the number of nodes present, the area they are spread over, their mobility patterns, among other factors. The performance fundamentals have been studied quite extensively (e.g., [7, 29, 25]).

Applications to be used in such a networking environment have to be delay-tolerant (i.e., not expect an instant response), should be robust (i.e., not rely on every message being delivered), may only have loose demands on serialization and synchronization, and have to avoid protocols using handshakes (such as challenge-response, or feature negotiation mechanisms) [5, 19]. This suggests making application data units (ADUs) carried in messages *self-contained*. Consequently, messages should embed the entire content, the necessary context for its interpretation, and possibly required credentials. In addition, the effect of protocol operations should be made *idempotent* with minimal cross-message dependencies. Such applications differ from typical web applications in which users enter (e.g., queries), and obtain virtually immediate responses; cloud applications where contents are stored, updated, and maintained consistent in data centers; and conversational applications where messages and media are delivered reliably and virtually instantly. Sample applications include content sharing [16, 10], distributed text and voice messaging [8, 9], and social networks [17, 11].

With their distinct requirements, opportunistic networking applications clearly form a niche. Here, Metcalfe's law appears to apply even stronger, because nodes serve both as application peers to interact with and as the communication substrate. Rather than connecting via the Internet to backend services, users require nearby peers to establish a network and run the distributed applications. Even though some applications (e.g., *Twimight* [8]) are "dual" in nature (i.e., can connect to the Internet but also support "offline" operation), the incentive for a user to install (and continuously run) an application appears limited: After installation, there is usually no content nor anybody to talk to immediately, unless users set up an application in a group. Even then, once the group disperses, data exchange ceases, as other message carriers would be needed.

In this paper, we explore *technical* mechanisms to increase the number of potential users of mobile opportunistic networks as one step of addressing this chicken and egg problem: (1) We stipulate that making shared content from an opportunistic network accessible to users who have not installed opportunistic networking software on their smart devices (and may never do so), which we call *legacy* devices, may increase its value and may get further users interested in participating. To this end, we extend the messaging mechanisms to include presentation code for rendering and interacting with message content. (2) We provide low-cost, autonomous

CHANTS'14, September 7, 2014, Maui, Hawaii, USA.

ACM 978-1-4503-3071-8/14/09 ...$15.00.
http://dx.doi.org/10.1145/2645672.2645681.

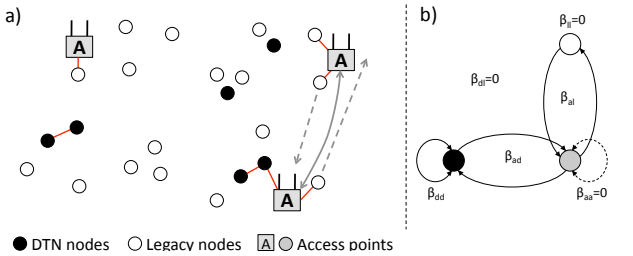

Figure 1: System model: a) Sample setup also hinting that there will likely be (many) more legacy than DTN nodes. Red lines indicate communication links. The dashed arrows show sample node movement, which yields a virtual "backbone" link between two access points (solid grey arrow). b) Formalized possible interactions with meeting probabilities. Some access points could be connected via (wireless) backhaul (dashed line) but we do not assume so in this paper.

infrastructure devices, *Liberouter* nodes [11], to serve as access points (APs) and storage for other mobile nodes and to offer a web interface to the content for legacy users as part of a captive portal. This portal also serves as app store, so that interested users can install the opportunistic router and related applications.[1] (3) We instrument legacy nodes connecting to the *Liberouters* and visiting the web pages to serve as additional message carriers between different *Liberouters* using cookies and web storage, so that the connectivity between *Liberouters* increases. While we have presented the *Liberouter* concept before [11], the contributions of this paper are on its extensions to embrace legacy nodes.

2. SYSTEM MODEL

Our system model (figure 1a) includes three classes of nodes: mobile nodes that run the opportunistic routing system, termed *DTN nodes* (d), the *legacy nodes* (l) that do not, and DTN-enabled access points (a). DTN nodes can establish communication links (thin red lines) with one another when in radio range as well as with APs, but they cannot communicate with legacy nodes. Legacy nodes can only interact with access points, but not with each other, nor with DTN nodes. In addition to talking to all mobile nodes, APs could also have backhaul links among each other, but we do not assume so in this paper. We consider APs to be sparsely spread, stationary, and hence never interact directly. Figure 1b shows the possible message exchanges.

Research on mobile opportunistic networking has explored networks comprising only mobile DTN nodes as well as adding access points (throwboxes, infostations, *Liberouters*) for capacity enhancement, data seeding, Internet access, and other functions [6, 23, 30, 1, 20, 16]. These functions include also creating wormholes between access points by means of Internet backhaul, or long range radio links. Our inclusion of legacy nodes provides a different flavor of a backbone network, because legacy nodes are limited in their interaction to access points. They create an opportunistic backhaul whose performance is a function of the number and movement patterns of the legacy nodes (grey arrows in figure 1a).

This setting yields an opportunistic network with heterogeneous nodes, the performance of which has been studied for certain contact patterns and routing protocols [25]. Intuitively, we expect the delivery performance—measured as the fraction of sent messages that are delivered and the delivery delay—to improve, as we add more access points (because they yield additional contacts and thus improve connectivity) and legacy nodes (because they increase connectivity between access points). We determine the contact rate estimates β for the Random Waypoint mobility model following Lemma 1 of [7]) using simulations for the setup of section 5.1: $\beta_{dd} = 0.109978h^{-1}$, $\beta_{ad} = \beta_{al} = 0.128984h^{-1}$, and $\beta_{aa} = \beta_{ll} = \beta_{dl} = 0$. Using the results (Lemma 6.1) of [25], we can calculate how messages spread for arbitrary combinations of DTN, legacy, and AP nodes. Figure 2 illustrates the effect on a single message spreading in an *empty* network comprising 8 APs and a variable number of DTN and legacy nodes moving in an area of $1\times1\text{km}^2$. Particularly for a low density of opportunistic nodes, inclusion of legacy nodes should improve spreading, and thus the message delivery probability within a finite period. In the following, we explore the system design to realize this potential and evaluate its performance in more complex settings.

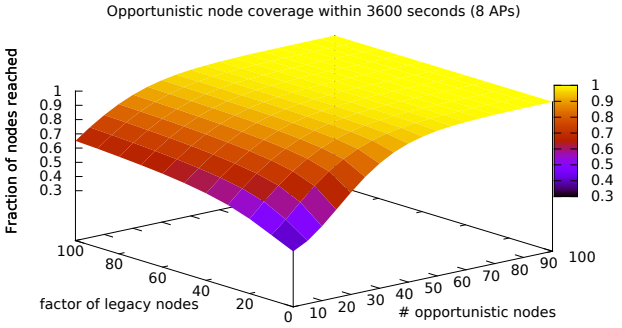

Figure 2: Potential performance impact of legacy nodes

3. OPPORTUNISTIC NETWORKING AND THE LIBEROUTER FRAMEWORK

Our starting point is the SCAMPI opportunistic router platform [12]. It is responsible for discovering nearby nodes, opening communication links, and implementing store-carry-forward networking that enables the asynchronous multi-hop messaging service between nodes. The implementation follows the DTNRG networking architecture and protocol specifications [3, 22, 4] with some extensions. The communication services offered to applications include unicast messaging, publish/subscribe services, and probabilistic geo-tagged content sharing (*Floating Content* [10]). Applications can exchange messages of virtually arbitrary size. Metadata attached to the messages (key-value pairs) support the above services, provide hooks for security mechanisms, and allow applications to provide "typed" auxiliary information.

We consider mobile nodes, at this point Android devices, running the SCAMPI router platform along with one or more applications. In addition, we foresee deployment of a (small) number of standalone *Liberouters* as infrastructure nodes. A *Liberouter* is a low-cost embedded Linux device (e.g., Raspberry Pi) that appears as an open WLAN access point using a predefined ESSID ("LIBEROUTER"), and thus allows arbitrary devices to connect to it. It runs an instance of the SCAMPI router platform, and thus complements the opportunistic network. Mobile devices near a *Liberouter* and running the SCAMPI router automatically discover, connect to the *Liberouter* and exchange messages.

In addition, each *Liberouter* features a captive web portal with two main functions: (1) Mobile nodes can be set up for opportunistic networking: the SCAMPI router platform and applications are available locally for download, so that a single *Liberouter* can bootstrap a community of nodes; (2) It can directly provide users with web-based access to content and enable forwarding via legacy

[1]We consider it important to also seed contents to *Liberouters* (e.g., from the Internet), but this aspect is orthogonal to this paper.

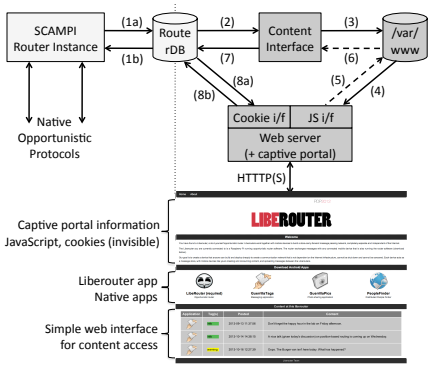

Figure 3: Interfacing message-based communication applications to legacy devices using web technology.

nodes. The former allows devices with native *Liberouter* support to access the services (currently Android), while the latter supports any platform with a modern web browser (e.g., iOS).

A smart mobile device could also act as a *Liberouter* variant by operating as an access point and allowing other devices to connect to it. Running mobile devices opportunistically as access points has been suggested to circumvent the problem of direct interaction between mobile devices, as WLAN ad-hoc mode is not widely supported by mobile operating systems APIs, and devices often require manual "peering" or explicit user interaction before they can communicate (e.g., WiFi Direct) [26, 28]. Combining this with a web server and a captive portal yields mobile *opportunistic Liberouters*. Their implementation poses further challenges due to OS diversity and limitations, however, and is subject to future work.

Unlike devices that run the SCAMPI router platform, legacy nodes require users to actively connect to a *Liberouter* AP once[2] and to at least open the web browser of the device and load any web page upon connecting to the access point to start using opportunistic services. We are working on automating this further to minimize the user interactions to just opening the web browser and the liberouter page once (and keeping it open) by performing automatic page loading using established JavaScript mechanisms (e.g., web sockets). Then, a mobile device would automatically to connect to *Liberouters* and serve as a legacy router as long as the liberouter page is open in one tab, giving users explicit control over this function. Memory utilization is limited to the web storage and cookie capacity available to a single domain (as for all other web sites), thus avoiding a negative impact on other apps.

4. WEB-BASED OPERATION

In this section, we address the interaction of the *Liberouter* framework with *legacy* devices that do not run the *Liberouter* code. The basic idea is to implement a set of web-based extensions for the captive portals of *Liberouter* APs. Figure 3 illustrates these extensions. We leverage the fact that the SCAMPI router stores encoded messages in the *Liberouter* file system (1a) to provide two data access paths: one, (8a) and (8b), allows for message forwarding (see section 4.1), while the other, (2)–(4) and (5)–(7), allows for stored content access and upload (see section 4.2).

4.1 Message Forwarding

The SCAMPI router uses the file system to persistently store encoded messages. We have implemented an additional *file system convergence layer* to validate and import messages from a dedicated *incoming* directory (1b). We provide mobile nodes with two ways of retrieving messages and injecting them into the *Liberouter*, namely HTTP cookies [14] and JavaScript *Web Storage* [27]. On the *Liberouter*, both extensions are implemented in PHP.

[2]After an initial association with this WLAN, today's mobile devices can easily be configured to automatically reconnect.

Browser	Max. cookie size	Max. cookies per domain
Chrome/Android	4kB	180
Firefox	4kB	150
Safari/iPad	8kB per domain	
Opera	4kB	175

Table 1: Cookie capacities of popular web browsers.

4.1.1 Cookies

We make use of HTTP cookies [14] to store small messages. Since all *Liberouter* instances advertise the same domain in their captive portal, cookies retrieved from one *Liberouter* are offered to the next one. To maximize content volume stored in cookies, we have measured the available cookie capacity for existing web browsers (see table 1) and use the *User-Agent* HTTP header to infer how much we can store in the browser. As many HTTP servers limit the HTTP request length to 8 KB, we have tailored a version of *lighttpd* to allow for request lengths up to 720 KB. Nevertheless, the amount of data that can be transferred using cookies is quite limited, but it suffices to carry a number of short messages (e.g., *Twimight* tweets or *Guerrilla Tags* messages).

We prefix cookie names with "lrbp-" and then add the text representation of the SHA-1 hash of a message as the cookie name (to provide name collision resistance), assign the *base64* encoding of the complete message as the value, and map the message TTL to the cookie expiry time. We use realm "liberouter" and path of "/". When a node connects to the *Liberouter*, it offers its stored cookies. The *cookie i/f* module retrieves the messages currently stored in the *Liberouter*, calculates the SHA-1 hashes (i.e., cookie names) for these local messages smaller than a threshold (web browser-dependent), and determines the difference set between the local and offered cookies. The *Liberouter* then saves the new messages in the *incoming* directory. Finally, it chooses maximum number of cookies (accepted by the browser) to return for storage in the browser of the mobile node, randomly mixing received and own cookies to ensure that contents carried further by the mobile nodes does not get biased towards the *Liberouter*'s own content. The *Liberouter* keeps track of the mobile node and performs cookie exchange only once during a given time period (e.g., one hour) or per session, whatever lasts longer, to avoid unnecessary overhead.

4.1.2 JavaScript and HTML5

As cookie capacity limits communication to the exchange of only small messages, we use the HTML5 *Web Storage* API[3] and provide JavaScript code that interacts with the *Liberouter*'s *JS i/f* module to exchange larger messages. We use the same identification mechanisms as for cookies. The JavaScript code exports a list of messages that the mobile node carries (from the *localStorage*) and sends it to the *Liberouter*, which then computes the difference set between the messages stored in the *Liberouter* and the ones on the mobile node. It dynamically crafts and parametrizes a page with code that obtains the new messages from the mobile node. Finally, *Liberouter* makes the mobile retrieve a random mix of messages to be stored in *localStorage* for carrying and forwarding.

To ensure that there is no overlap of messages that are transmitted via both mechanisms, we make use of the *User-Agent* HTTP header to find out what is the maximum allowable size of the single cookie, and put into the *localStorage* only messages that are larger. Defining a right value for the upper size of a message that can be inserted into the *localStorage* is a trade-off between size of the message and number of messages we want to transport inside a single

[3]Current browsers allows for up to 5.2 MB of data per site.

legacy node. In our current implementation, we have decided to use *localStorage* only for messages that are not larger than 200kB (size of the large document). This upper message size allows us to convey on average about 30 messages inside one node.

We also do not coordinate simultaneous message exchanges with multiple mobile nodes. To avoid concurrency problems, each set of incoming messages is dropped into its own subdirectory and only moved to the incoming directory once the transfer has been completed. The file names for the messages are derived from the SHA-1 hashes (and those are validated by the *Liberouter* after message reception) so that no two different messages can overwrite each other.

4.2 Content Access

The above mechanisms increase the available network capacity by engaging legacy devices in message exchange. However, these mechanisms alone do not offer an incentive for the device owner to use them. Thus, to encourage users of legacy devices to benefit from the *Liberouter* system, we have developed the generic *content interface* module that provides *read access* to the content stored on the *Liberouter* AP and *write access* to generate their own content. Obviously it is also possible to use the device just for message forwarding (e.g., to limit energy consumption). In such cases, users would just limit their interaction with the system to the main page.

4.2.1 Read access

To provide a generic *read access* to the stored content, we have developed the *message presentation toolchain* that comprises two main units. The first one (see (2) in figure 3), parses messages from the SCAMPI database and extracts metadata to create HTML5 code. This code is then used for message presentation and stored in the file system. Each message should embed a small *python* script, as metadata, to generate HTML5 code that describes it. For messages that do not contain such script, the unit generates HTML5 code based on other embedded metadata. Messages that contain composite content (e.g., conversation thread) are always required to embed script for their data to be properly presented. The second unit (see (3) in figure 3) reads generated code from the file system, and arranges it to be presented to the user via the web page.

4.2.2 Write access

The heart of the *write access* unit is a web form. If a user wants to interact with some stored content (e.g., reply to a tweet), the *python* script that has been used to generate the older message is used to produce such a form. For new messages of selected applications, a generic form is offered. The *write access* unit allows users to upload text messages, files, and metadata using a simple JavaScript interface to the dedicated server directory (see (5)–(6) on figure 3). The content of this directory is tracked by the SCAMPI app working inside the *content interface* module. The SCAMPI app generates valid *Liberouter* messages from the uploaded file contents and stores it in the SCAMPI router database (see (7) on figure 3).

4.2.3 Security considerations

Executing scripts of unknown content to generate message presentation data, or uploading arbitrary data to the system raises security concerns. To address this problem, all *python* scripts are executed in their own dedicated directory with no access rights to anything except for the message metadata.

In opportunistic networks, *read access* to messages can generally be limited by using encryption, which makes it easy to completely prevent access via the web interface. Allowing access for selected legacy users requires suitable key management mechanisms (at the protocol level and for system integration), which we leave for further study. Generating new messages via the *write access* raises the question of data origin identity. The web interface allows uploading a content signature for origin authentication. Users who read the signed content may verify authenticity of the signature with the help of public key of the creator that can be obtained, e.g., via the *PeerShare* system [18]. For responses, an original message posted could also include a list of (and credentials for) authorized responders [15], so that receivers as well as *Liberouters* can determine if a certain response is valid or not.

Parameter	Values
Mobility models	RWP (1×1km^2), SPMBM, OPP (Helsinki 4.5×3.4 km^2)
# DTN nodes N_d	RWP: {10, 20, 50}, SPMBM, OPP: {50, 100, 200}
# legacy nodes N_l	$\{0, 1, 5, 10\} \times N_d$
Radio range	RWP: 20 m, SPMBM, OPP: 50 m
Net data rate	2 Mbit/s (no interference)
DTN routing	Epidemic, (Binary) Spray-and-Wait (6, 12 copies)
DTN node buffer	mobile node: 10 MB, AP: 500 MB
Legacy node buffer	150 cookies per 4 KB and 2.6 MB web storage
Message size	U(10 B, 1 MB)
Messaging interval	$d = U(0.75, 1.25) \times t$ for $t \in \{12, 60, 300\}$s
Message lifetime	RWP: 3 h, SPMBM, OPP: 1.5 h

Table 2: Overview of simulation parameters.

5. VALIDATION

5.1 Simulations

We evaluate the system performance using the ONE simulator [13] to which we add a *LiberouterApplication* class that is agnostic to the underlying routing protocol and is run, e.g., on access points and a *LegacyRouter* class for legacy nodes that will only interact with the nodes running the former. We choose three different mobility models: 1) RWP: Random Waypoint in a 1×1km^2 area with $N_d \in \{10, 20, 50\}$ DTN nodes, and 8 or 16 access points spread out symmetrically (to reflect the performance models of section 2. 2) SPMBM: Shortest Path Map-Based Movement between waypoints chosen from the Helsinki downtown map (4.5×3.4km^2) [13] for $N_d \in \{50, 100, 200\}$ pedestrians moving with speeds $v = U(0.5, 1.5)$m/s without predefined points of interest, plus 11–325 stationary access points as per [21]. 3) OPP: Shortest Path Map-Based Movement as in the identical configuration to 2), but using {10, 20, 50, 100}% of the mobile DTN nodes as access points. We vary the number of legacy nodes $N_l \in \{0, 1, 5, 10\} \times N_d$.

Node discovery and access point association are instantaneous, but each node can send/receive message to/from only one other node at a time. We use Epidemic routing and controlled replication (6, 12 message copies) using binary and regular Spray-and-Wait [24].[4] Every *messaging interval*, a single message of random size and lifetime is generated at one random DTN node destined to another random one: 12 s is referred to as high, 60 s as medium, and 300 s as low load. Table 2 summarizes all simulation parameters. a simulation run lasts for 12 h; we conduct 10 runs each and report the mean performance metrics. We report on the aggregate observations of the 600+ different simulation settings, as we are interested in the general impact of instrumenting legacy nodes.

We observe quite a diverse behavior across our three models: expectedly, the message delivery rate is lowest for RWP (as the unrestricted movement nodes meet less frequently) and decreases with increasing offered load, as node buffers and exchange capacity per contact get saturated. For RWP, we observe delivery rates without legacy nodes of 7.5–41%, 29–88%, and 58–97% for high,

[4]For controlled replication, an access point will store a message copy in a legacy node only if the remaining copy count is at least two; the copy count is not updated when forwarding to legacy nodes so that the total number of copies in the system grows.

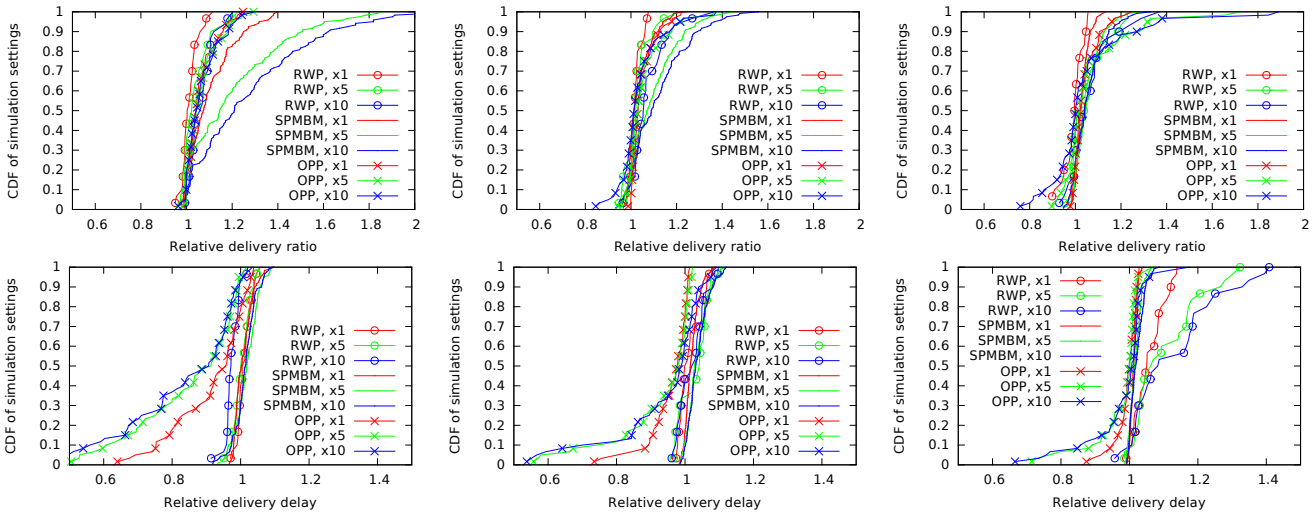

Figure 4: Simulation results for the relative delivery ratio (top row) and relative delivery delay (bottom row) for different messaging intervals: 300 s (low load, left), 60 s (medium load, middle), 12 s (high load, right). ×1, ×5, ×10 indicate the factor of legacy nodes.

medium, and low load, respectively. SPMBM and OPP show similar behavior under varying load, but achieve higher delivery ratios for low load. SPMBM exhibits larger variations (e.g., 32–99%), as opposed to OPP (77–100%) because of the larger number of nodes and the stationary APs, which may "capture" messages in scenarios with low controlled replication.

Figure 4 shows the relative improvement of the delivery ratio (top row) and the relative delivery delay (bottom row) for all mobility models for low, medium, and high loads (left to right) and legacy node numbers. In all these figures, we show CDFs across all the simulation settings (normalized as different mobility models have different permutations) to indicate the general trends.

We observe a positive impact of legacy nodes for most of the simulation settings and that the number of legacy nodes dominates the improvement of both delivery ratio and latency independent of the offered load and mobility model. Across all scenarios, we find a (small) number of cases where legacy nodes inclusion leads to (minor) performance degradation. This effect is more pronounced for higher loads and often with Epidemic routing. This happens because individual legacy nodes may compete with DTN nodes for obtaining replicas from APs and because they increase message spread and thus contribute to the overall load and buffer fill level.

Comparing all three models, we find that the impact of the number of legacy nodes on the delivery ratio is highest for SPMBM, decreasing with increasing load; varies most for OPP with the load, and is modest and not heavily load-dependent for RWP. SPMBM sees such a strong impact because the additional access points deployed (up to 325) yield many points of contact for DTN and legacy nodes to interact and feature higher storage capacity. The number of APs is smaller for RWP and contacts are generally less frequent as mobility is less restricted, so that the inclusion of legacy nodes yields only limited extra connectivity. OPP features mobile APs, so that each AP comes in contact with more DTN and legacy nodes and the network gets better connected already with a few legacy nodes. Moreover, mobile APs can be source and destination, so that direct delivery of messages through legacy nodes is possible. Hence, adding some legacy nodes already provides some gain, but increasing their number has little effect for low loads. For medium and high loads, OPP exhibits a broader performance spread (top right), ranging from 25% degradation to an 80+% improvement. These extremes stem, on the one hand, for degradation, from an already well connected network (200 nodes, all APs). On the other hand, the notable outliers for improvement (see also figure 5) are controlled replication with the smallest node population (50 nodes), again all acting as access points. In these—unlikely—setups all mobiles are also APs so that legacy nodes can relay messages between any two nodes and contribute most to increase connectivity, which has a positive impact for small populations (cf. figure 2), but may exacerbate the situation in already well connected networks as the legacy nodes essentially perform flooding.

Looking at delivery delay (figure 4 bottom row), we see another important impact: a substantial reduction in delivery delay for OPP, which is far less pronounced for SPMBM, while RWP sees mostly latency increases. The delay reduction is mainly because the APs are mobile (OPP), whereas their number has a lesser effect. An increase in mean latency is partly a secondary effect of more messages being delivered, particularly visible for RWP and especially at high load (bottom right).

Finally, we summarize the relative performance in a scatter plot in figure 5 to show different impacts we observe per mobility model: RWP experiences modest gains in delivery ratio, which results in increased delays. SPMBM can gain quite a substantial delivery ratio improvements, but the delays barely change. OPP experiences mostly latency reduction with modest delivery rate improvements. Across all mobility models, we find a stronger—positive as well as negative—impact when larger numbers of legacy nodes are present but, overall, the positive impact dominates. In summary, including legacy nodes can provide a performance gain as long as the additional messaging load—as a function of the added connectivity and the offered load—does not saturate the system.

5.2 Initial Experiments

We finally perform experiments to understand how message forwarding via the web interface compares to the native DTN protocol.

Message forwarding transaction time. We run experiments (10 trials) using an iPad 3 (iOS 7.0.4) and Raspberry Pi's *Liberouter* connected over WiFi. Because of iPad software constraints, we can store only about 30 cookies in the browser. We provision the Raspberry Pi with different numbers of messages {100,500,1000,5000}; the message sizes are independently chosen from U(140 B,1 MB), representing a size range from Twitter messages to a photo. The mean total execution time (table 3) increases linearly both for cookies and local storage, unsurprisingly, but at different rates: a 50-fold increase in stored messages yields a 42-fold increase for cookie, but

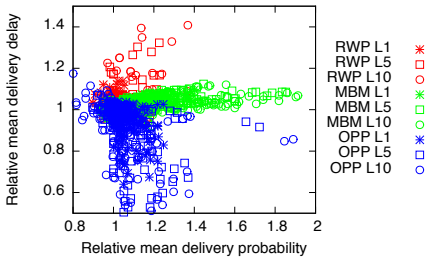

Figure 5: Scatter plot of all simulations showing delivery ratio vs. delivery delay, both relative to no legacy nodes.

Number of messages	Cookies		Local storage	
	avg	std	avg	std
100	0.075	0.022	1.891	1.077
500	0.305	0.043	3.447	1.076
1000	0.567	0.005	8.42	0.675
5000	3.27	0.889	24.42	3.162

Table 3: Mean message forwarding transaction times (and standard deviation) for different message numbers in the AP.

only a 13-fold increase when using local storage. Since cookies are smaller (<350B), their transmission time is minimal compared to the local I/O operations and computation on the *Liberouter*. Note that our current implementation performs the hashing for message comparison on demand when a new legacy node connects (rather than continuously as a background task and storing the results), so that the results are suboptimal. Especially with such improvements, exchanging messages with legacy nodes is clearly feasible.

Web interface vs native opportunistic protocol. We also compare the throughput of native DTN and web-based data exchange. We use a Samsung Galaxy Tablet GT-P3100 running Android 4.1.2 as the mobile node. We preload 30 cookies and 70 local storage items into its browser (total size 1.3 MB) and connect it via WLAN to the *Liberouter*, which holds 70 non-overlapping messages. We measure the time it takes to complete the data exchange and repeat the experiment 10 times. The web interface obtains an average throughput of 631 KB/s ($\sigma = 85$ KB/s), the native interface only 100 KB/s ($\sigma = 50$ KB/s). The performance difference stems from the handshake protocol carried out between the *Liberouter* and the mobile device prior to the actual message exchange, which requires a number of steps at different layers at both nodes, while the web interface is unilaterally run by the *Liberouter* AP. This deficit amortizes with larger exchanged volumes.

6. CONCLUSION

In this paper, we have explored the idea of making contents exchanged in opportunistic network accessible to users who do not run the corresponding software on their devices and leveraging their devices to assist in message spreading. Theory suggests a positive impact on overall system performance, which our simulations confirm for most of the—still limited—scenarios we explored. We implemented our design in our *Liberouter* and carried out a set of initial experiments, showing that both web-based interaction and message carriage are feasible. One open issue is how to further automate the instrumentation of legacy nodes, so that message forwarding could operate in the background.

Besides a broader performance evaluation (also using traces if suitable ones become available), future directions include more comprehensive support for presentation and interaction code embedded in messages, adding security for *read* and *write* access, and improving the design of the (currently very simple) web interface with proper usability considerations—along with content seeding.

Acknowledgements

This research has received funding from the EC FP7 project PRECIOUS under grant agreement no. 611366 and from EIT ICT Labs.

7. REFERENCES

[1] A. Bujari, C.E. Palazzi, D. Maggiorini, C. Quadri, and G.P. Rossi. A solution for mobile dtn in a real urban scenario. In *Proc. of IEEE WCNC Workshops*, April 2012.

[2] Y. Cao and Z. Sun. Routing in Delay/Disruption Tolerant Networks: A Taxonomy, Survey and Challenges. *IEEE Comm. Surveys & Tutorials*, 15, 2012.

[3] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-tolerant networking architecture. RFC 4838, April 2007.

[4] Mike Demmer, Jörg Ott, and Simon Perreault. Delay Tolerant Networking TCP Convergence Layer Protocol. Experimental RFC 7242, June 2014.

[5] Kevin Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proc. of ACM SIGCOMM*, 2003.

[6] R. H. Frenkiel, B. R. Badrinath, J. Borres, and R. D. Yates. The Infostations challenge: balancing cost and ubiquity in delivering wireless data. *IEEE Wireless Communications*, 7(2):66–71, 2000.

[7] Robin Groenevelt, Philippe Nain, and Ger Koole. The Message Delay in Mobile Ad Hoc Networks. In *Proc. of IFIP Performance*, October 2005.

[8] Theus Hossmann, Franck Legendre, Paolo Carta, Per Gunningberg, and Christian Rohner. Twitter in Disaster Mode: Opportunistic Communication and Distribution of Sensor Data in Emergencies. In *Proc. of ExtremCom*, 2011.

[9] Md. Tarikul Islam, Anssi Turkulainen, Teemu Kärkkäinen, Mikko Pitkänen, and Jörg Ott. Practical Voice Communications in Challenged Networks. In *Proc. ExtremeCom*, 2009.

[10] Jussi Kangasharju, Jörg Ott, and Ossi Karkulahti. Floating Content: Information Availability in Urban Environments. In *Proc. of IEEE Percom, Work in Progress session*, March 2010.

[11] Teemu Kärkkäinen and Jörg Ott. Liberouter: Towards Autonomous Neighborhood Networking. In *Proc. of IEEE WONS*, 2014.

[12] Teemu Kärkkäinen, Mikko Pitkänen, Paul Houghton, and Jörg Ott. Scampi application platform. In *Proc. of ACM MobiCom CHANTS workshop*, 2012.

[13] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proc. of SIMUTools*, 2009.

[14] David M. Kristol and Lou Montulli. HTTP State Management Mechanism. RFC 2109, February 1997.

[15] Arseny Kurnikov, Teemu Kärkkäinen, and Jörg Ott. Data to the People (poster). In *ACM MobiCom workshop on Challenged Networks (CHANTS)*, 2013.

[16] V. Lenders, M. May, G. Karlsson, and C. Wacha. Wireless ad hoc podcasting. *ACM/SIGMOBILE Mobile Comp. and Comm. Rev.*, 12(1), 2008.

[17] Anders Lindgren. Social networking in a disconnected network: fbDTN: facebook over DTN. In *Proc. of ACM CHANTS workshop (demo)*, Sep 2011.

[18] Marcin Nagy, N. Asokan, and Jörg Ott. PeerShare: PeerShare: A System Secure Distribution of Sensitive Data Among Social Contacts. In *Proc. of NordSec*, October 2013.

[19] Jörg Ott. 404 Not Found? – The Quest for DTN Applications. Keynote abstract. In *Proc. of MobiOpp 2012*, March 2012.

[20] F. De Pellegrini, I. Carreras, D. Miorandi, I. Chlamtac, C., and Moiso. R-P2P: A Data Centric DTN Middleware with Interconnected Throwboxes. In *Proc. of Autonomics*, 2008.

[21] Mikko Pitkänen, Teemu Kärkkäinen, and Jörg Ott. Opportunistic Web Access via WLAN Hotspots. In *Proc. of IEEE PerCom*, April 2010.

[22] Keith Scott and Scott Burleigh. Bundle Protocol Specification. RFC 5050, November 2007.

[23] Giuseppe Sollazzo, Mirco Musolesi, and Cecilia Mascolo. TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks. In *Proc. of MobiOpp*, 2007.

[24] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *ACM Workshop on Delay-Tolerant Networking*, 2005.

[25] Thrasyvoulos Spyropoulos, Thierry Turletti, and Katia Obrazcka. Routing in Delay Tolerant Networks Comprising Heterogeneous Populations of Nodes. *IEEE Trans. on Mobile Computing*, 8, 8 2009.

[26] Sacha Trifunovic, Bernhard Distl, Dominik Schatzmann, and Franck Legendre. WiFi-Opp: Ad-Hoc-less Opportunistic Networking. In *Proc. of ACM MobiCom CHANTS workshop*, Sep 2011.

[27] W3C. Web Storage, 2013.

[28] Hanno Wirtz, Tobias Heer, Robert Backhaus, and Klaus Wehrle. Establishing Mobile Ad-Hoc Networks in 802.11 Infrastructure Mode. In *Proc. of ACM MobiCom CHANTS workshop*, Sep 2011.

[29] Xiaolan Zhang, Giovannia Neglia, Jim Kurose, and Don Towsley. Performance Modeling of Epidemic Routing. In *Proc. of IFIP NETWORKING*, 2006.

[30] Wenrui Zhao, Yang Chen, Mostafa Ammar, Mark Corner, Brian Levine, and Ellen Zegura. Capacity enhancement using throwboxes in dtns. In *Proc. of IEEE MASS*, October 2006.