Opportunistic Content Dissemination Performance in Dense Network Segments

Teemu Kärkkäinen
Technical University of Munich
kaerkkae@in.tum.de

Mika Välimaa, Esa Hyytiä
Aalto University
fistname.lastname@aalto.fi

Jörg Ott
Technical University of Munich
ott@in.tum.de

ABSTRACT

Many of the existing opportunistic networking systems have been designed assuming a small number links per node and have trouble scaling to large numbers of potential concurrent communication partners. In the real world we often find wireless local area networks with large numbers of connected users – in particular in open Wi-Fi networks provided by cities, airports, conferences and other venues. In this paper we build a 50 client opportunistic network in a single Wi-Fi access point and use it to uncover scaling problems and to suggest mechanisms to improve the performance. The ability to scale to high density network segments creates new, realistic use cases for opportunistic networking applications.

CCS Concepts

•**Networks** → **Wireless access points, base stations and infrastructure; Network performance analysis;** *Network protocol design; Peer-to-peer protocols;*

Keywords

Dense wireless networks, opportunistic networking, performance analysis, wireless testbed

1. INTRODUCTION

Trace analysis has shown that human mobility is characterized by large periods of time spent in a dense cluster and infrequent flights between the clusters [3, 13, 9]. This is intuitive since humans tend to aggregate and spend their time in places such as cafeterias, shopping malls, office buildings, and class rooms. These aggregation points are often covered with Wi-Fi networks since ubiquitous Internet access is expected by users of smartphones and laptops. The resulting wireless local area networks can have hundreds of connected mobile devices – we call these *dense network segments*.

We have previously shown that Wi-Fi networks can be used to create direct contacts between mobile devices to compose an opportunistic network [7, 8]. Much of the existing opportunistic networking research and most system implementations [2, 4, 6] assume that, at any given time, each node only has a small number of neighbors to which it can connect. This assumption does not hold in dense network segments with hundreds of potential communication partners.

In this paper we show that the basic opportunistic networking mechanisms do not scale to dense segments and suggest mechanisms for improving the resulting performance problems. We construct a 50 node testbed from embedded Linux devices, connected to the same Wi-Fi access point, running an opportunistic networking middleware, which is used to run experiments to evaluate message dissemination performance. We make three main contributions: 1) We design and deploy a testbed made from real devices suitable for evaluating dense segment scenarios. 2) We analyze content distribution performance in a dense network segment using a commercially available implementation of basic opportunistic networking mechanisms. 3) We propose and evaluate topology control and request randomization mechanisms for improving the performance.

CHANTS'16, October 03-07 2016, New York City, NY, USA

ISBN 978-1-4503-4256-8/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2979683.2979692

2. EXPERIMENTAL SETUP

The goal for our experimental setup is to replicate the real world dense wireless network segment scenario as closely as possible, while maintaining the control and repeatability necessary for producing statistically significant results. To this end, we built a testbed of 50 embedded Linux devices connected to a single Wi-Fi access point, along with a test suite software for automatically executing the experiments and collecting the results. This creates a realistic environment that includes all the layers from the physical radio communication to the application layer, running on standard off-the-shelf hardware and software.

2.1 System Model

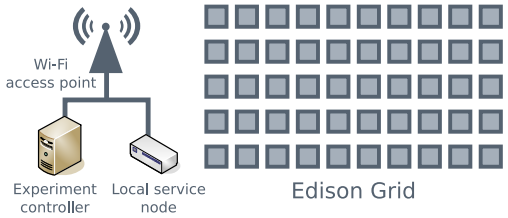


Figure 1: Experiment system model.

Our system model – shown in Figure 1 – replicates a single access point wireless local area network. The main components are the wireless *access point*, the *local service node*, the *client grid*, and the *experiment controller*. The dense wireless segment is created by connecting the client devices in the grid to the access point.

The local services needed to support the segment, e.g., DHCP and NTP servers, are run on the local service node connected to the access point. The experiment controller runs the experiments by, e.g., flashing the client devices with the correct operating system images, running the test cases, and collecting the results. Due to the limitation of our chosen client hardware, the experiment control signaling is done over the same wireless network as the experiment itself. This has the potential to interfere with the measurements, so our test suite is specifically designed to generate minimal control traffic while the experiment is in progress.

2.2 Communication Model

We assume a simple model for the communicating nodes, which matches the basic behavior of various opportunistic communication platforms. For our tests we use the Scampi [6] opportunistic networking middleware implementation which gives us control of the basic opportunistic mechanisms that we need. We focus on the relative scaling properties, not the absolute performance of Scampi.

The basic functions that each node implements are: 1) *peer discovery*, 2) *link creation* between peers, 3) *control messaging*, and 4) *content exchange*. These are all done on the local network segment through the access point. As a first step after connecting to a network segment, each node first discovers directly reachable peers using, e.g., local broadcast messaging. Next, the node will choose some set of discovered peers to which it opens links, resulting in a logical network *topology*. We consider the *full mesh* topology where every node opens a link to every discovered node, and *limited mesh* where every node randomly opens n links – "n-link limited mesh". Both sides of a new link will send their content vectors, containing the identifiers of the cached messages, to the other side. The nodes then request a list of all the messages that they wish to receive from the peer, which the peer will start transmitting in order, one at a time, over the link. The peers will continue to send content vectors whenever their cache contents change. In this paper we assume that nodes will request all messages from the peers that they do not already have, resulting in *epidemic spreading* of the content. More intelligent routing and content dissemination strategies exist, but we focus on the baseline case against which other algorithms can be compared later.

2.3 Hardware and Software Setup

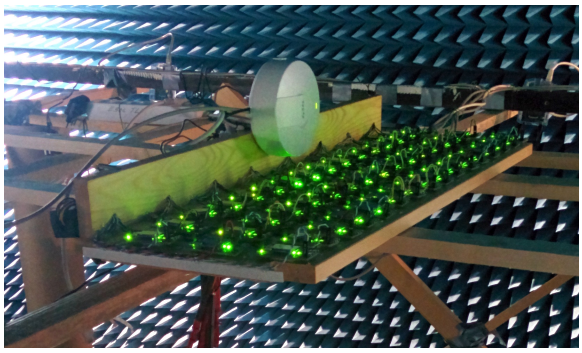


Figure 2: Experiment hardware setup.

The physical configuration of the hardware is shown in Figure 2. The white circular device the middle of the figure is the D-Link DWL-6600AP chosen as the access point. We evaluated a number of access point devices, including Cisco WAP610N and Zyxel NWA1121-NI, and found them incapable of supporting 50 clients. We further found that the choice of the access point has a major impact on the overall system performance, and provide a brief analysis and performance comparison of two hardware models in Section 3.5.

For the client hardware, glowing green in the figure, we selected embedded Linux devices. In reality the clients are likely to be smartphones, but building a testbed with them presents multiple challenges: 1) Most are closed platforms, which makes it hard to control the software to the degree necessary; 2) they include much unneeded functionality, such as the screen and cellular connectivity; 3) they are physically larger and not designed to be easily mounted in a fixed setup; and 4) the cost is 2-10 times higher than embedded devices.

The specific platform we selected is the Intel Edison system-on-chip with Intel's mini breakout board, which is similar in hardware performance to typical smartphones. It is a dual core 500 MHz x86 Intel Atom based system with 1 GB of RAM and 4 GB of on-board flash memory, and an integrated 802.11n Wi-Fi chipset operating in the 2.4 GHz and 5 GHz bands. The devices are powered by two power supply units via the breakout board's 12 V input pins with ferrite beads added to filter out high frequency noise.

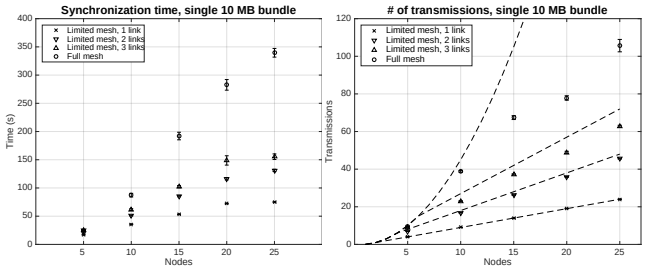
For the set of experiments documented in this paper, the client devices run a custom operating system image based on the Intel Yocto distribution (*ww05-15*, Linux 3.10.7). The image is stripped down from various components not needed for our tests, and with Oracle Java 8 runtime and Scampi router added – otherwise the image is close to standard.

To run the experiments, we built extensible test suite software in Python, which is run in the experiment controller. The test suite first sets up the initial state in the clients, including clearing the previous test state, synchronizing the clocks and copying over the correct configuration files. Then the test case is invoked, which implements the test specific logic. Finally the results are gathered from the clients, and the previous steps are repeated until the required number of iterations have been run. For the set of experiments reported in this paper – a node entering and disseminating content into a dense segment – the test logic is quite simple: First the required number of clients is randomly picked from the client grid and initialized with empty Scampi instances. Next a random client is picked as the entering node, firewalled off from the other clients and its Scampi instance seeded with content. Finally the firewall is lifted and Scampi instances in the clients are allowed to start synchronizing their state. During the synchronization process no control signaling is done by the test suite so as not to interfere with the experiment and the end of the test is detected from the power usage (i.e., when all radios have gone idle).

3. DENSE SEGMENT DISSEMINATION

We focus on evaluating a static scenario, where a single node connects to a dense network segment already containing a number of other nodes. We are interested in determining the efficiency with which the content carried by the arriving node can be disseminated to the other nodes. This matches a real world scenario of a user entering a cafeteria, library, or another busy location with a WLAN network, while carrying content received from a previous hotspot.

In this section we analyze the following aspects of message dissemination: 1) Dissemination performance of a *single message* carried into a dense segment by a node. 2) The impact of the *logical topology* – the TCP links opened between the nodes in the segment – on the dissemination performance. 3) The *spreading speed* for the message within the segment. 4) Whether the single message dissemination performance generalizes to a case with *multiple messages*. 5) The performance impact of the network *hardware*.



(a) Synchronization time. (b) Transmission count.

Figure 3: Static scenario dissemination performance.

3.1 Single Message Dissemination

To evaluate the basic message dissemination performance we set up a test with N nodes, $N-1$ of which were connected to the access point with empty caches ($5 \leq N \leq 25$) and a full mesh of interconnecting links between them. We then connected a node with a 10 MB message in its buffer to the access point and measured the *synchronization* time, i.e., the time taken until all N nodes have a copy of the message. Further, we recorded the number of Scampi message transmissions that took place during the synchronization.

The results show that both the synchronization time and the number of message transmissions to reach the synchronized state increase as a function of N (circles in Figure 3a and 3b). We also confirmed that when the number of nodes is kept constant, the synchronization time only depends on the message size and the total network capacity.

In a simple model we would expect that the node entering the segment will open links to all the other nodes and then send its message over the links in parallel. Although the transmissions over the links are logically parallel, the underlying shared wireless medium serializes and interleaves the packets, leading to a synchronized network roughly in time $T = (N-1) \times S/C$, where S is the size of the message and C the total capacity of the wireless segment. However, in the results we observe the synchronization time growing at a faster rate, implying that there is another phenomena hurting the performance. This can be clearly seen in the number of transmissions, which in a simple model should equal the number of nodes but in our results grows at almost six times that rate – going from 10 to 15 nodes results in almost 30 more transmissions, when it should result in only 5 more according to the simple model.

We have identified the cause of this growth to be what we call *request-stacking* behavior. The main cause of this behavior is that while the node entering the segment will start sending the message to all the peers simultaneously, due to packet drops and other lower level behavior the transmissions will not finish at the same time. Instead, one of the transmissions will finish first, at which point the node will advertise the received message to all its peers. The peers that have not yet received the full message from the original source will then request a copy, which takes up capacity in the wireless segment and thus slows down the ongoing original message transfers. This process will continue as the second node finishes receiving the message, and so on, resulting in slower than expected synchronization time and – given the lack of transfer cancellation – more transmissions.

The aggressive request mechanism that causes request-stacking is a valid approach in the face of unpredictable and highly dynamic contacts. In general it makes sense to aggressively try every opportunity to get a copy of the message even if a transmission is already ongoing, since the new contact could have much higher capacity or the existing transmission might get cut before finishing. However, in dense, largely stable network segments this approach leads to severely degraded message dissemination performance.

3.2 Impact of Logical Topology

The request-stacking phenomena observed in the previous section gets worse the more direct neighbors each node has, with the worst case being the full mesh topology where every node is directly connected to all other nodes. We therefore continue by looking at the message dissemination performance when the logical topology is limited, i.e., where each node opens links only to a subset of the potential peers.

Random topologies: In the full mesh topology, after receiving a message, each node will start a transmission to every other node that has not yet received the message. This means that the worst case number of transmissions will be

$$C = \binom{N}{2} = \frac{1}{2}(N^2 - N), \tag{1}$$

out of these the number of duplicates is

$$C_e \equiv C - (N-1) = \frac{1}{2}(N^2 - 3N + 2). \tag{2}$$

Therefore the number of duplicate transmissions is limited by $O(N^2)$, which makes the full mesh, with the naïve dissemination strategy, poorly scalable in dense segments.

The natural way to reduce duplicate transmissions is to reduce the number of direct neighbor links that each node maintains. With fewer direct neighbors, each node will advertise its newly received messages to fewer other nodes, and thus fewer duplicate retransmissions will occur. We evaluate 1, 2 and 3 link limited mesh topologies (see 2.2), the results of which are given in Figure 3 (triangles and crosses).

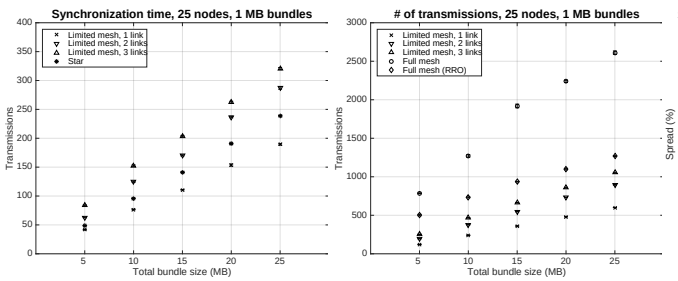
From the results it is clear that the duplicate transmission count is roughly proportional to the number of links in the logical topology. And a smaller number of links in the logical topology in turn results in shorter overall synchronization time, all the way down to 1-link limited mesh. The impact of the limited topology on performance is substantial already in relatively small segments of 10 nodes, and increases as the density increases; in the densest analyzed segment the 1-link limited mesh synchronizes 4.5 times faster than a full mesh.

We can derive an estimate for the transmission counts in limited mesh topologies, shown as dashed lines in Figure 3b. First, since we assume that the topology limitation is done by limiting the number of peer links each node *opens*, a node will have on average $2M$ open links. In a network with N nodes, the topology will approach full mesh when $M \geq (N-1)/2$. Next, from the Figure 3b we can see that there is a linear relationship between the link count M and the number of nodes N. From this, we can use the full mesh threshold as a base value to which a linear increment of M times the number of new nodes is added to form the basic equation for estimating limited mesh performance:

$$C = \begin{cases} \frac{1}{2}(N^2 - N), & 0 < N \leq 2M \\ (N-1) \cdot M, & N > 2M \end{cases} \tag{3}$$

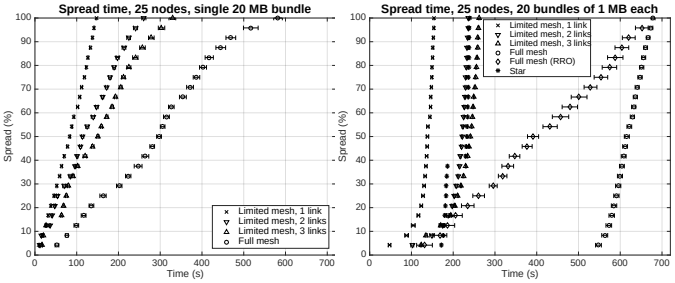
for all $N > 0$ and $M > 0$ describes the behavior of an M-link limited mesh topology in a network with N nodes.

Star topology: Another approach to reach a more efficient logical topology is to build it explicitly. Two obvious topologies that cannot have duplicate messages transfers are a *spanning tree* and



(a) Synchronization time. (b) Transmission count.

Figure 4: Multiple message dissemination performance.



(a) Single message. (b) 20 messages.

Figure 5: Spreading speed of the messages.

a *star topology* with one central "hub" to which the rest of the nodes connect. Out of these, the former would require running a distributed algorithm to create and maintain the spanning tree as nodes enter and exit the segment, while the latter can be organized simply by, e.g., having every node open a link to the node with the lowest address. The network operator could even provide a fixed node in the segment to act as the hub. Therefore the star topology is likely the more practical topology.

We evaluated the performance of the star topology in the same static scenario as the random topologies, and found that its performance was consistently between the performance of 1-link and 2-link limited mesh topologies. An example of this is given in Figure 4a for different amounts of data to be synchronized. All topologies show linear growth as a function of the amount of data to be transferred, with the star topology performing slightly worse than 1-link limited mesh. This performance difference occurs even though in both cases the number of transmissions is almost the same – with star topology having no duplicate transmissions. We attribute this to the different pattern of parallel transfers in the topologies causing different congestion characteristics on the lower layers. The star topology will start all the transfers in parallel after the hub has received the message, while the limited mesh will have fewer parallel transfers. More parallel transfers means more nodes competing for their share of the shared medium, leading to, for example, less efficient allocation of the resources by the CSMA-CA MAC mechanism.

3.3 Message Spreading Speed

In the previous sections we focused on the time and transmissions taken to reach a synchronized state, but did not consider the details of how the state was reached. To fill this gap, we measured the fraction of nodes reached over time as the content was disseminating in the network.

The results – depicted in Figure 5a – show that the previously observed performance behavior is consistent throughout the message distribution process. The full mesh topology performs worst while limiting the logical topology improves performance all the way down to the case of 1-link limited mesh, which is the optimal topology. The star topology overall seems to fall between the 1-link and 2-link limited mesh topologies, but has a linear behavior over time. This is as expected, since the limited mesh will have few concurrent transfers in the beginning that will finish quickly, while the star topology starts all the transfers concurrently. There is a discontinuity for the star topology at around 200 seconds due to a hardware effect, which we examine in more detail in Section 3.5. We also confirmed that the spread speed is linear w.r.t. the file size regardless of the topology. This supports the hypothesis that the limiting performance factor is the network node count and not the message size or count.

Importantly, the dissemination starts slower in random topologies with a higher number of links, as can be seen when comparing the spread speed between 1-link limited mesh and full mesh for the first data points (crosses and circles in Figure 5a). This is significant in scenarios where a node visits a dense segment for a short period of time, for example when walking by a cafe and connecting briefly to its Wi-Fi access point. In such scenarios using a full mesh topology could mean that the node does not have time to fully transfer the message before exiting the segment. Therefore, new nodes entering a dense segment should have a small number of links to maximize the number of messages they can disseminate to the segment before exiting.

3.4 Multiple Message Dissemination

The analysis thus far has mainly considered a single message being carried into the network segment. To analyze the behavior of disseminating multiple messages, which is the realistic case, we ran a set of experiments with 25 nodes and 5 to 25 messages being disseminated instead of a single message. We found that the number of transmissions grows linearly in every topology as a function of the number of messages. We found no other behavior than the increased capacity taken to transfer the increased amount of data.

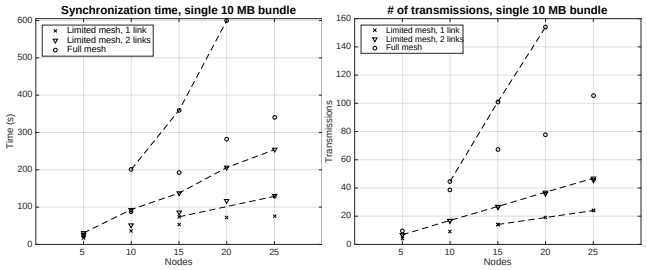
We did, however, find the same request-stacking behavior described previously. Each node will request all k messages the same way as it would for a single bundle, resulting in duplicate transmissions. The number of transmissions for k messages, N nodes and a full mesh topology is at worst

$$C = k + 2k + 3k + \cdots + (N-1)k = \frac{k}{2}(N^2 - N), \qquad (4)$$

which is exactly k times the single bundle case given in eq. 1.

When looking at the spread speed – shown in Figure 5b – the previously described slow start phenomena is even more severe. This is because in the single message case the message will be requested concurrently by at worst $N-1$ nodes, after which the number of concurrent transfers will decrease, freeing up capacity. In the multiple message case, however, the number of concurrent transfers stays at $N-1$ until the first node in the network has received all the messages, only after which capacity is gradually freed. In the example shown in the figure, it takes over six minutes for the first node in the network to receive all the content when there are 20 MB of data to be disseminated.

Random Request Order (RRO): A simple strategy to alleviate both the request-stacking and slow start phenomena – particularly



(a) Synchronization time. (b) Transmission count.

Figure 6: Synchronization performance of the DWL and NWA devices. NWA results are connected by dashed lines.

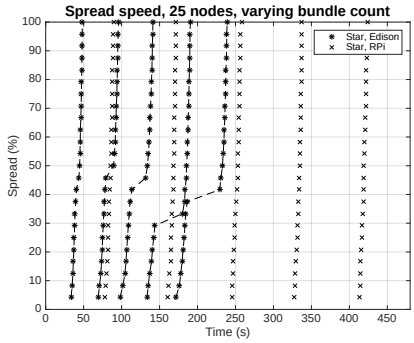


Figure 7: Spread speed of different hub hardware with varying message count.

in the full mesh topology – is to randomize the order in which nodes transfer messages. When nodes are transferring different messages between different pairs, the number of duplicate transfers due to request-stacking should decrease, and thus the spread speed should increase.

To analyze the impact of RRO, we apply a simple Fisher-Yates shuffle to the content request lists exchanged by the nodes in the full mesh topology, where the impact is most visible. The result is a significant decrease in the number of transmissions required to synchronize the network – as shown in Figure 4b – with the full mesh RRO approaching the limited mesh performance. We can further see from Figure 5 (diamonds) that the spread speed behavior of the full mesh is radically changed by the RRO. While the total time taken to synchronize the full mesh topology is almost the same in both cases, applying RRO spreads the set of messages in the network more evenly over time. This reduces the slow start phenomena; the first node to receives all the messages in almost the same as in a 2-link limited mesh.

While we have considered basic epidemic spreading of content, the results have implications for more advanced algorithms – for example utility or interest based dissemination. In particular, such algorithms should be wary of mechanisms that might lead to identical request rankings in a large number of co-located peers, and if necessary, add randomization techniques to prevent it.

3.5 Hardware Performance

During the testing we noted that hardware performance has a significant impact on the message dissemination performance. The impact is significant already in small – 15 node – scenarios, which makes the hardware choices critical for the system performance in real world deployments. Therefore we document these observations here even though detailed hardware performance analysis is outside the scope.

We found the client hardware – dual core 500 MHz x86 with 1 GB RAM – is capable of running the Scampi software without significant bottlenecks, and saturating the 802.11n 5 GHz Wi-Fi adapter. The performance will, however, be limited by the speed at which incoming messages can be written to the persistent memory – which is a requirement for store-carry-forward networking. We benchmarked the Edison clients, which have integrated persistent flash memory, and found them to be capable of writing 7 ± 1 MB/s. This is the upper limit at which each device can receive content regardless of the network performance if persisting each message is required. Overall we do not foresee deployment issues arising from the raw hardware performance of the client devices; the limitation with real mobile devices is likely to be the energy use and limited battery capacity [14].

In contrast, the choice of the Wi-Fi access point hardware does have a significant impact on the system performance. We evaluated two access points – DWL-6600AP and NWA1121-NI – the former being advertised as a "professional" and the latter as a "consumer" grade device. The results show a significant difference both in stability and in performance, to the benefit of the DWL device. Figure 6a shows the difference in content synchronization time of a 10 MB message between the two devices. With all other things being equal, using the more capable DWL device cuts the content dissemination time in *half* already with only 15 devices in the segment. Further, the NWA device collapses with 25 clients, while the DWL device continues to function even with the full 50 client devices active. The difference is due to the packet/frame forwarding performance of the device, as Figure 6b shows that the number of Scampi message transfers remains the same while the transmission time differs (for limited mesh). This also implies that if the number of concurrent message transfers is small, even a less robust device can handle the load.

In the analysis of the *star topology* earlier, we noted a discontinuity in the spread speed (Figure 5b) when an Edison is used as the hub device. The discontinuity means that at a time the dissemination process comes to a halt, but then after about a minute suddenly continues as normal. We do not have a full explanation for this, but we can narrow it down to a hardware issue by comparing the Edison against a Raspberry Pi 1 B as the hub node – shown in Figure 7. From the figure it can be clearly seen that the Edison consistently has a discontinuity while the RPi does not. It can also be noted that the more powerful Edison is also consistently more performant as the hub node than the RPi (single core 700 MHz ARM with 512 MB RAM). However, overall we found the Edison and Raspberry Pi devices to be capable of serving as hub nodes even for a large number of clients and messages.

4. RELATED WORK

Various studies of opportunistic networking implementation performance have been done, particularly in the field of Delay-Tolerant Networking (DTN) – an established field with multiple mature implementations to evaluate. Pöttner et al. compared the performance of three DTN implementations [12] – the same protocols used in this paper – focusing on the achieved throughput as a function of the message size. The experiments were done with two or three nodes running on computers, while our testbed has up to 50 clients running on embedded devices. Similar performance analysis has also been done on embedded devices similar to ones used by us [5]

and on Android devices [11]; again with a small number of devices, which does not show the dense segment scaling properties. Performance analysis on DTN implementations on a scale similar to ours was done by Beuran et al. [1], with a QOMB testbed built on a wired network of computers with links generated based on a mobility model. A similar emulated testbed – TUNIE – has also been proposed by Li et al. [10]. In contrast, our testbed has real embedded devices connected to a real Wi-Fi access point, and we focus on the realistic dense segment content dissemination scenario rather than pinging in mobile scenarios.

5. CONCLUSION

In this paper we have analyzed the scaling properties of opportunistic content dissemination in dense network segments via a realistic testbed of 50 nodes. The results show that the basic opportunistic networking mechanisms have severe scaling problems already with 10 devices sharing the same wireless medium – which establishes a baseline comparison case for developing more intelligent mechanism in the future. Although we focused on infrastructure Wi-Fi, the same structure – and hence the same problems – will also arise with Wi-Fi Direct, where one of the devices acts as a soft AP to which the other devices connect. The performance problems stem from the large number of duplicate transmissions caused by the combination of redundant links in the logical topology and the aggressive replication by the message flooding. Finally, we discovered that the access point hardware selection is critical to the performance of even small deployments of over ten clients in a segment, while the client hardware performance does not cause bottlenecks.

We also proposed and evaluated improvements to alleviate both sources of the scaling issues. First, we showed that randomly limiting the number links in the logical topology significantly improves the scaling performance compared to a full mesh topology. We found the optimal topology in our scenarios to be the random 1-link limited mesh, while a star topology with a lightweight central hub device is also viable. Second, we alleviated the excessive duplication caused by the message flooding by randomizing the order in which nodes exchange their messages. The simple randomization mechanism significantly improves the content dissemination performance even in a full mesh topology. This implies that even the more intelligent content dissemination algorithm designs should consider adding randomness to the decisions in order to avoid parallel transfers of the same content in environment with large numbers of co-located peers.

At least three areas of future work follow from the results in this paper: First, intelligent algorithms are needed to build efficient logical topologies. This could include dynamic, time-variant topology control where a node entering a dense segment would first open a small number of links in order to disseminate its content quickly – in case it only stays for a short time – while slowly building more links over time to eventually start behaving as star node. Second, simulation models – which are often link-oriented – could be updated to reflect the shared medium, dense segment characteristics. Third, the protocol stack could be redesigned to take advantage of the inherent broadcast nature of the wireless communications. In principle a node should be able to broadcast its content to nearby peers in constant time, but the current unicast-oriented protocol stacks are not designed for it. These developments could open interesting scenarios for new opportunistic networking applications in dense segments that are common in the real world – large public Wi-Fi networks.

Acknowledgments

The authors would like to thank Viktor Nässi from Aalto University Comnet department for his valuable help in setting up the experimental infrastructure.

This work was supported by the European Commission Horizon 2020 Programme RIFE Project Grant No. 644663.

6. REFERENCES

[1] R. Beuran, S. Miwa, and Y. Shinoda. Performance evaluation of dtn implementations on a large-scale network emulation testbed. In *Proc. of ACM CHANTS '12*.

[2] M. Doering, S. Lahde, J. Morgenroth, and L. Wolf. Ibr-dtn: An efficient implementation for embedded systems. In *Proc. of ACM CHANTS '08*, 2008.

[3] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[4] O. R. Helgason, E. A. Yavuz, S. T. Kouyoumdjieva, L. Pajevic, and G. Karlsson. A mobile peer-to-peer system for opportunistic content-centric networking. In *Proc. of ACM SIGCOMM MobiHeld '10*, 2010.

[5] G. M. Johnson Williams, B. Walker, and A. Hennessy. A performance comparison of dtn bundle protocol implementations on resource constrained nodes. In *Proc. of ACM CHANTS '12*, 2012.

[6] T. Kärkkäinen, M. Pitkänen, P. Houghton, and J. Ott. Scampi application platform. In *Proc. of ACM CHANTS '12*, 2012.

[7] T. Kärkkäinen, M. Pitkänen, and J. Ott. Enabling ad-hoc-style communication in public wlan hot-spots. *SIGMOBILE Mob. Comput. Commun. Rev.*, 17(1):4–13, July 2013.

[8] J. Lakkakorpi, T. Kärkkäinen, and J. Ott. Protecting regular customer traffic from ad-hoc traffic in public wlan hot-spots. In *PIMRC*, pages 2114–2119, 2013.

[9] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong. Slaw: A new mobility model for human walks. In *INFOCOM 2009, IEEE*, 2009.

[10] Y. Li, P. Hui, D. Jin, and S. Chen. Delay-tolerant network protocol testing and evaluation. *IEEE Communications Magazine*, 53(1):258–266, 2015.

[11] H. Ntareme and S. Domancich. Security and performance aspects of bytewalla: A delay tolerant network on smartphones. In *IEEE WiMob '11*, 2011.

[12] W.-B. Pöttner, J. Morgenroth, S. Schildt, and L. Wolf. Performance comparison of dtn bundle protocol implementations. In *Proc. of ACM CHANTS '11*, 2011.

[13] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong. On the levy-walk nature of human mobility. *IEEE/ACM Trans. Netw.*, 19(3):630–643, June 2011.

[14] S. Trifunovic, A. Picu, T. Hossmann, and K. A. Hummel. Slicing the battery pie: Fair and efficient energy usage in device-to-device communication via role switching. In *Proc. of ACM CHANTS '13*, 2013.