BeachNet: Propagation-based Information Sharing in Mostly Static Networks

Jörg Ott, Ari Keränen, Esa Hyytiä
Aalto University, School of Electrical Engineering
Department of Communications and Networking, Finland

ABSTRACT

Delay-tolerant and opportunistic networking are widely investigated for information exchange in (sparse) mobile scenarios, in the evaluation of which numerous mobility models and traces have been employed. These networking techniques may also be applied in static and fairly dense scenarios for non-directed information sharing, in which nodes may but need not be connected all the time. We present different information dissemination algorithms for content sharing, taking a network formed by mostly immobile users on a beach as one example and evaluate them through simulations.

1. INTRODUCTION

Opportunistic or delay-tolerant networking (DTN) is one solution to overcome assumptions of traditional mobile ad-hoc networks that mobile nodes would generally be sufficiently dense to form a mostly connected network in which end-to-end paths can be established. Consequently, a lot of research has gone into investigating sparse scenarios using a variety of mobility models as well as mobility traces of humans and vehicles to study the performance of routing and information sharing protocols for such setups. While some efforts also went into denser scenarios and explored, e.g., the idea of bridging traditional MANETs and mobile DTNs dynamically [16, 25], static ad-hoc networks have not received much attention. These are "extreme" scenarios from the perspective of opportunistic networking because the frequent assumptions of sparse node population and infrequent contacts are violated.

Quite a few semi-static scenarios with fairly high node density can be envisioned: from summer beaches with sunbathers to picnic areas in parks to stadiums and possibly to amusement parks. All these share three key properties: 1) Any given user is likely to have more than one other mobile user in communication range at any given time. 2) The non-moving users will often dominate over the mobile ones at any given instant. 3) They have an open population of nodes, i.e., nodes appear and disappear at random.

We are interested in understanding options for content distribution in such scenarios: A source node wants to share a piece of content with a group of nodes, where the group members may or may not be known to the source (unicasting is a special case).

In our semi-static scenario, nodes may form well-connected networks in large parts. But it is unpredictable when a node will be in a such an area and whether local cues, e.g., readings from its immediate surroundings, hold for the entire area. Moreover, not all nodes may be connected all the time so that some notion of delay tolerance is required for robustness. Especially, content sharing requires semantics that make content wait for users yet to arrive or become reachable. Finally, MANET-style end-to-end forwarding is known to suffer from performance degradation as the number of hops increases [20] making it unappealing (cf. a node in a P2P overlay holding a content copy within an ad-hoc network).

To accommodate our target operating environment and to overcome potential issues of end-to-end communication, we opt for content dissemination using the *store-carry-forward* principles of opportunistic networking, even though most of our nodes are *not* mobile. While many approaches rely on node mobility for spreading content to users across space [13, 7] or keeping content available in a limited area [10, 15, 8, 23], we require the content to spread while the nodes are stationary.

An obvious solution is to apply epidemic spreading, which, however, requires the total content volume to fit into each node's storage. Lifting this restriction requires either coordinating which node is responsible for which content (as in P2P overlays), or making the content move gradually over time. We pursue the latter approach to avoid network-wide coordination and perform content replication solely upon local information. We present the case study of *BeachNet* as a sufficiently representative example of stationary nodes spread out across a bounded area, which is in this case stretched out and thus more challenging (section 3). We derive two algorithms operating on local information (section 4), compare their performance against epidemic spreading using simulations (section 5), and conclude with a brief discussion (section 6).

2. RELATED WORK

BeachNet nodes form an ad-hoc network, albeit with limited mobility. Standard routing protocols for *ad-hoc networks*, such as AODV [19] and DSDV [18], are designed to find an end-to-end path in order to deliver packet between given nodes [17], including group communications [5], for time-synchronous content delivery. MANET protocols can deal with topology changes, but do not support sparse or partitioned (but otherwise dense) networks well. Moreover, wireless multi-hop communication suffers from performance degradation with an increasing number of hops [20]. Proposals for combining MANET and DTN routing [16, 25] are able to overcome temporary disconnections but they support only unicasting (and they would consider the connected parts as a big network partition, subject to multihop performance degradation).

One alternative to overcome the above limitations is constructing overlays for storing replicas to achieve decoupling in space and

ExtremeCom2011, September 26-30, 2011, Manaus, Brazil.

time. Traditional *peer-to-peer (P2P)* file sharing protocols, such as BitTorrent, organize content storage among their peers and provide structured (e.g., using DHTs) or unstructured ways of locating and retrieving content. These protocols assume a fully connected (IP) network with routing functionality, while also mobile P2P networking has been proposed (see, e.g. [21]). In any case, the assumption is that a file is stored fully or in part on one or more nodes, a subset of which may be queried for retrieval. The risk of network partitions and the associated search and maintenance issues make this approach appear less favorable. BeachNet assumes smaller file sizes, allowing contents to be moved around proactively.

Another alternative is delay-tolerant [3] or mobile opportunistic networking from which we borrow the main concepts. Related to BeachNet are particularly proactive content distribution via content channels (e.g., *PodNet* [13]) or in geographical regions (e.g., *Floating Content* [15]), multicasting [28], and (socio-aware) publish-subscribe mechanisms for opportunistic networks (e.g., [27]). We cannot rely on human mobility (as in PodNet or Floating Content), avoid state maintenance (as needed for multicasting), and cannot rely on social ties between users as aid for content delivery.

We focus on methods for *moving* a finite set of copies across the given area so that the interested nodes obtain the information within a reasonable time frame when, a priori, it is not known who and where the interested users actually are. BeachNet also does not implement any rendezvous mechanisms for data objects. We thus create a special purpose "best effort content channel" for communication in a naturally bounded area (because a beach ends).

With largely stationary nodes, BeachNet also features elements of *sensor networks*, where typical application is information gathering in or monitoring of a given area and relaying the findings for further analysis. Part of this process is often data compression and aggregation [1, 26]. Energy saving, e.g., by means of sleep-wake cycles [12, 4], is one important aspect for their longevity which we apply to BeachNet. However, in BeachNet there may not be a dedicated destination for a content, but instead the content actively moves around restlessly seeking for nodes interested in it. Due to battery operation, the unnecessary transmissions should still be avoided, and hence a balance must be sought between the availability (e.g., mean acquisition time) and the energy consumption (e.g., transmission frequency and fraction of sleeping time).

Numerous adaptive mechanisms for determining sleep-wake cycles for (mobile) node to improve neighbor discovery were developed, including [24, 2, 9, 14]. While we envision algorithms like those to be incorporated into BeachNet nodes for minimizing energy usage while maximizing interactions, we have to defer in-depth considerations of such alternatives to future work and explore only fully synchronized or random operation in this paper.

3. BEACHNET OVERVIEW

We seek to address information sharing (one-to-one and one-to-many communication) in environments characterized by the following properties that hold for beaches as representative scenario.

1. *Large static node setups with minimal mobility:* This implies that we cannot rely on nodes being active message carriers but rather messages have to move actively. While some nodes may move around (when coming and going) or pass constantly (cf. promenades along a beach), thus making them nice complementary message carriers, we leave such elements for further study and just focus on the fixed node setup.
2. *Variable node populations both in terms of number and identities of nodes:* These are nodes coming and going or being active (awake) and inactive (asleep). As a result, network partitions may appear and disappear unpredictably over time, rendering a "global" view of the network difficult (if not impossible) to achieve and maintain.
3. *A wide range from sparse to dense node populations:* This prevents us from assuming a static and well-connected network in which tasks (e.g., storing a copy of a message) can be assigned to individual nodes (cf. the above node dynamics).

Each node changes its behavior over time: between walking or driving prior to arrival at and after departure from the beach and being at the beach. Except for (non-)movement and relatively constant device neighborhoods, no further cues are immediately available at the device to determine its operating environment.[1] However, inferring the operational conditions is subject of future work.

Assume a source node S wants to share a message M_k with one or more other receiver nodes R_i in such a setting. When originating M_k at time t_k, S does not know how many potential receivers (if any) are around and where they are located, nor if one or more will arrive in the future (and when and where). Also, a given receiver R_i may not know before a time T_i that it is interested in a particular message. For one-to-one communication with M_k addressed to R_i, T_i is the very time R_i appears on the beach; for one-to-many messages, T_i is when R_i declares interest in a topic matching M_k.

Considering our scenario characteristics, we do not attempt to build up routing information (for subscriptions to content channels or for individual nodes), but rather design data-driven dissemination mechanisms. The basic requirement is that any content item spreads throughout the network and reaches (at least) majority of the nodes over time, including those appearing only after the message was spread or declaring interest in a message only very late.

Moreover, since many nodes will be permanently connected and could thus constantly exchange messages, we need to pace interactions to prevent the nodes from running out of battery quickly. Therefore, we consider sleep-wake cycles as suggested for sensor networks or for neighbor discovery but, as noted above, limit our considerations to always active, synchronized, and random operation, thereby providing upper and lower bounds.

Without these dynamics, one could simply implement a one-time flooding of each message throughout the network. If the memory of all nodes sufficed to hold all messages (until they expire), this epidemic-style approach would also support time shifted interest in content, sleeping, and late arrivals. In the following section, we present several algorithms for dealing also with memory constraints. An ideal routing solution achieves acceptable performance in terms of mean acquisition time and coverage, while minimizing the number of transmissions and energy consumption in general at the same time. The challenge is to come up with a simple yet robust algorithm that spreads the information in *waves* repeatedly

4. INFORMATION SHARING

In this section, we will introduce the information sharing algorithms that will be evaluated in the next section.

4.1 Epidemic

As a simple reference, we use epidemic routing [22] as a simple flooding protocol for opportunistic networks. Messages get replicated whenever a node comes into radio range that does not yet hold a copy of the message. When the node buffer gets full, the messages received earlier are deleted in FIFO order.

[1]Geo location provides another cue and a database of all places where BeachNet-style conditions could occur might be available in the Internet and even downloaded with regional maps to mobile devices, but we prefer to avoid such dependencies.

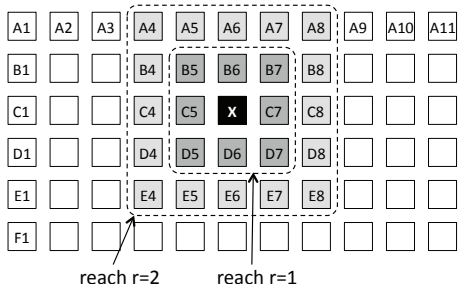

Figure 1: Game of Life overview.

4.2 Conway's Game of Life

The beach setup described above is essentially a mostly static setting with minor changes in the environment (nodes coming and going). In an abstract representation, this may be viewed as a grid of nodes—or cells in the *Game of Life* automaton [6]. Game of Life simulates cells by a set of simple rules that infer the fate of a cell in a given position X from the number of live adjacent cells: if there are too few or too many neighbors, the cell dies; for some range of neighbors a new cell is born; and for another range, a cell survives. Figure 1 shows a sample grid of cells, in which cell X has eight neighbors (according to the original definition of a neighbor): the dark shaded cells $B5 - B7$, $C5$, $C7$, and $D6 - D8$. The rules are applied simultaneously in rounds so that the state of cell X in round $n + 1$ are solely dependent of the states of its neighbors in round n. In the Game of Life, initially, a number of organisms ("live cells") are placed randomly on the grid and their evolution is then observed when repeatedly applying the same set of rules.

For our communication scenario, each field of the grid is a node and the "alive" state of a cell means that a copy of a message M_k is available at this node. The radio range and the distance between nodes (i.e., their density) define the neighborhood of each node and the message, both of which we summarize as *reach*. A simplified version is shown in figure 1: for a reach r, a node in the middle of the area X has $n(r) = (2r + 1)^2 - 1$ neighbors: $n(1) = 8$, $n(2) = 24$, etc.; nodes near the edges have less ($A1$ just has 3 immediate neighbors, i.e., there is no wraparound). Node X senses its neighborhood and determines how many copies of the message are available at its neighbors and then executes a rule set similar to the original Game of Life. The message is either replicated to X (birth), kept at X (survival), or discarded at X (death). This essentially yields four values to fully specify the rules for our Life algorithm: l_s, l_b, u_b, and u_s, where

$$\begin{array}{ll} l_s : \text{lower survival bound} & u_b : \text{upper birth bound} \\ u_s : \text{upper survival bound} & l_b : \text{lower birth bound} \end{array} \quad (1)$$

and $l_s \leq l_b \leq u_b \leq u_s$. Let $N_{i,k}(n)$ be the state of node N_i with respect to message M_k so that $N_{i,k}(n) = 1$ indicates that N_i has a copy of M_k in round n and $N_{i,k} = 0$ otherwise. Let $C_i(n)$ be the number of message copies available at the cell's neighbors. The rules are then defined as follows:

$$N_{i,k}(n+1) \begin{cases} 0 & \text{if } C_i(n) < l_s \vee C_i(n) > u_s \\ 1 & \text{if } C_i(n) > l_b \wedge C_i(n) < u_b \\ N_{i,k}(n) & \text{otherwise} \end{cases} \quad (2)$$

We first seek to find suitable parameters for our specific scenario. One major difference from the original Game of Life is that our starting point will be a single message copy. Hence, $l_b = 0$, or the message would "die out" immediately. We explore the parameter space for different u_b bounds by simulations to find a combination that allows the message to survive while limiting the fraction of nodes that carry a copy of the message.[2] Our target is to find an upper bound that allows a sufficient number of message copies to survive and move around, without flooding the network unnecessarily with message replicas. To keep the combinatorial complexity under control, we choose $l_s = l_b$ and $u_s = u_b$ for this paper. Assuming that beaches will form rather long stretches and square playfields, we choose field sizes of $x \times y$ with $x = 500$ and $y \in \{1, 2, ..., 10\}$.

Our target is to find lower and upper bounds that are suitable for a broad range of environmental conditions.

In our initial exploration, we use a simple custom-written cell automaton (written in C). We assume a perfect grid of nodes as shown in figure 1 with neighborhoods defined by the reach; error-free and instant communication, i.e., messages are always fully transmitted; and just a single message initially placed at a random point inside the grid, uniformly distributed. We then observe the message spreading behavior over time. For all the simulations in this section, we use 1000 iterations[3] and 30 random seeds. We report on the mean number of message copies around in the end of the simulation as an indicator for the suitability of the chosen bounds.

Initially, we assume a fully static setup and explore the impact of different radio ranges. We model different densities by choosing to populate only a fraction $p_{\text{init}} \in \{0.5, 0.8, 0.9, 1.0\}$ of all the grid fields with nodes, randomly spread. Figure 2 shows the mean fraction of present nodes holding a copy of the message after 1000 rounds across all values of y for three different reaches. In addition (not shown), we investigate the minimum and maximum number of copies. We find that the fraction of nodes holding a copy grows quickly for $u_b > 2$. However, for $u_b = 2$, we find quite many (10 out of 12) setups, in which the message disappears. Only $u_b > 4$ yields no message losses in our runs for the dense case (reach=3), but causes a lot of overhead in sparser scenarios. Hence, $u_b \in \{3, 4\}$ appears to work generally fine for the sparse and medium and are equally at risk of message loss in the dense scenario. Since $u_b = 3$ causes less overhead, this appears preferable.

We then add "openness" to the system: starting with an initial node population, measured as a fraction p_{init} of the total spots in the grid, we allow nodes to enter and leave the beach according to a Poisson process with mean times T_a and T_d between arrivals and departures, respectively. Nodes leave by vacating a spot in the grid; they take the message copy they hold away. Nodes may only arrive to vacant spots; arriving nodes naturally do not carry a copy of the message. We investigate $T_a = T_d \in \{5, 10, 20, 50\}$ rounds for reach=1. We find that these—fairly modest—rates of arrivals and departures do not impact the above results qualitatively and also the quantitative variation is low.

Finally, since we model communication between mobile devices, we also need to take into account their energy consumption. After all, they need to last for a day on the beach. We also conduct simulations with nodes alternating between sleep state and active state. In contrast to nodes leaving, sleeping nodes maintain their message copy. We choose sleep and active times, T_s and T_c, evenly distributed from $[0; T_{\max}]$ and choose $T_{\max} \in \{5, 10, 20, 50\}$ rounds independently for T_s and T_c. We vary $p_{\text{init}} \in \{0.5, 0.8, 0.9, 1.0\}$ and as above set reach=1.

When introducing sleep and active times with the same $T_{\max}$, i.e., when nodes have equally long asleep and awake phases (on average), messages get lost occasionally across all scenarios. Nevertheless, for $p_{\text{init}} \geq 0.8$, messages generally survive and the mean fraction of nodes holding a copy remains in the same order of mag-

[2] We don't yet optimize to minimize the number of transmissions.

[3] We found in preliminary studies that messages will have spread throughout the $500 \times y$ area by then and reach a steady state.

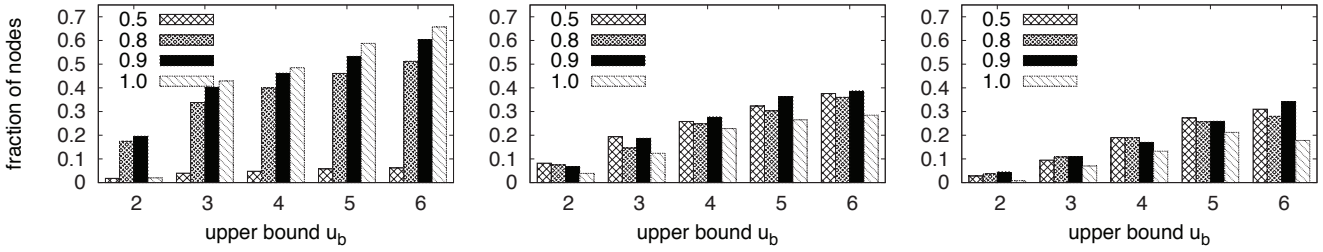

Figure 2: Static Life simulation results: mean fraction of nodes carrying a copy of the message after the simulation period of 1000 rounds. Left: sparse (reach = 1), center: medium (reach = 2), right: dense (reach = 3).

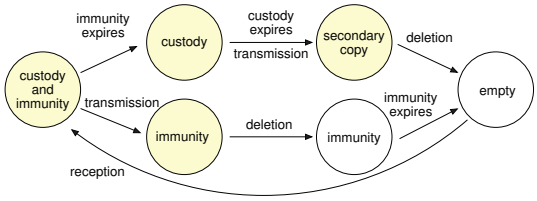

Figure 3: State diagram of a message in a node with Wave router.

nitude as for the constantly awake nodes. For $p_{\text{init}} = 0.5$, however, the mean availability shrinks to some 2%. These findings hold across all values for u_b. Increasing the reach, however, improves the case for $E[T_s] > E[T_c]$ noticeably.

Extending the mean active cycle length beyond the mean sleep cycle has a minor impact on the availability: in most cases, the number of message copies in the system grows. In the opposite case, message availability collapses quickly. For $2 \times E[T_c] \geq E[T_s]$, the availability remains in the same order of magnitude; for $4 \times E[T_c] \leq E[T_s]$, mean availability goes down to 2–3% and messages continuously disappear for $10 \times E[T_c] = E[T_s]$.

Based upon these findings, we choose $l_b = l_s = 0$ and $u_b = u_s = 3$ for our simulations below. In spite of the negative results for reach=1, we experiment with non-favorable sleep/active cycles to obtain a lower bound. Because of their limited impact, we do not elaborate further on nodes joining and leaving.

4.3 Wave

Wave router is a specially designed content-driven routing scheme to realize BeachNet scenario. In Wave, each node keeps track of the messages it has recently received. In particular, a node does not accept a message if it is found on the tracking list. The messages are kept in the list for a certain fixed time, which we refer to as the *immunity time* (node is "immune" to given message for a certain time period). The message deletion is executed independently of the tracking list, i.e., the node may be immune to a message even though the message is no longer in its possession. Moreover, upon accepting a message, the node agrees to take over the *custody* of the message, which means that it will not delete the message until it has been passed further (along with the custody) or the message's time-to-live expires; see figure 3.

Hence, the routing decision is a local operation suppose to ensure that messages find their ways to areas they have not recently visited, in accordance with the philosophy behind the BeachNet. Formally, the Wave router is described in algorithm 1.

Algorithm 1 Pseudo-code of Wave router.

Upon receiving message $\mathbf{m}$**:**
- **if** $m \in$ immunity table $\mathcal{I}$, or $m \in$ buffer $\mathcal{B}$ **then**
  - ignore the message
- **else**
  - **if** message buffer $\mathcal{B}$ is full **then**
    - **if** buffer $\mathcal{B}$ has message(s) without custody flag **then**
      - delete the oldest such entry (FIFO)
    - **else**
      - ignore the message
    - **end if**
  - **end if**
  - store m to $\mathcal{B}$ and set a custody flag with timer $2 \cdot t$ for m
  - add m also to immunity table $\mathcal{I}$ with expiration time t
- **end if**

Regular update event:
- clear expired custody and immunity flags from $\mathcal{B}$ and $\mathcal{I}$
- **if** buffer $\mathcal{B}$ is non-empty **then**
  - broadcast one message $m \in \mathcal{B}$ chosen in random
  - **if** reception of m is acknowledged **then**
    - clear the custody flag in m (local copy)
  - **end if**
- **end if**

5. EVALUATION

For our simulation-based evaluation, we use the ONE simulator [11] for which we have implemented special "mobility" model reflecting the beach scenario as well as and the aforementioned Life and Wave routing protocols supporting the BeachNet application. We also use the Epidemic router included in ONE.

We choose two different grid sizes: 100×10 and 50×5 nodes both occupying the same area, with 10 m and 20 m spacing between grid points. In our basic scenario, nodes are exactly located at the respective grid points; in a variation, they are offset from this point for x and y uniformly distributed between 0 and 20 m. We use two different radio ranges, 20 m and 80 m, and three types of sleep/active cycles: 1) no sleeping; 10 s wake and 100 s sleep cycles 2) synchronized within 10 s across nodes and 3) chosen at random without node synchronization. This yields a total of 72 setups. Nodes generate messages at a rate of 0.1 msg/node/hour and each node can store up to 10 messages. All simulation parameters are summarized in table 1.

We run each simulation for six hours with a time step of 1 s: this means that connectivity and neighbor message content is evaluated once per second and, for Life routing, a new round starts every second. We do not use warmup times to reach steady state in order to reflect the dynamics of the system with the number of messages growing during the day, but we consider the different periods that messages are available in our evaluation.

Figure 4 shows our initial *coverage* metric accounting for the spread of messages across nodes. We choose 40 nodes at random and count the number of nodes reached by each message at least once (ignoring subsequent deletion and repeated delivery). Both scenarios, 5×50 and 10×100, perform roughly similarly, with slightly better coverage achieved for the former simply because the

Hardware:	10 Mbps WLAN interface, 10 MB buffer size
Messages:	1 MB each, creation 1 msg / node / 10 hour
Radio ranges:	1) 20 m, 2) 80 m
Network topology:	1) 10×100 grid, inter-node distance of 10 m 2) 5×50 grid, inter-node distance of 20 m optionally a random offset $\mathrm{U}(0, 20\,\mathrm{m})$ added independently to (x, y) coordinates per node
Routers:	1) Epidemic 2) Wave: 5 min immunity 10 min custody 3) Life: $l_b - l_s = 0, u_b = u_s = 3$
Awake-sleep cycles:	1) none 2) 10 s / 100 s, max 10 s random offset 3) 10 s / 100 s, without synchronization

Table 1: Parameters for ONE simulator.

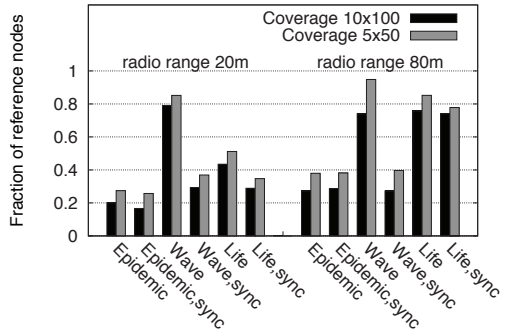

Figure 4: Coverage of messages without grid offset.

sampling nodes are denser. Expectedly, basic connectivity dominates message spread: 80m radio range performs equally well without and with grid offset, whereas a 20m range quickly leads to isolation with offsets (not shown). For the sufficiently connected scenarios, Life and Wave achieve better coverage than Epidemic (up to factor 2–3, reaching 75%+); in some cases, Life outperformed Wave, in others Wave reaches more nodes than Life. However, this initial metric does not cover all nodes, nor does it capture (repeated) coverage over time; a more comprehensive characterization in these respects is subject to ongoing work.

Table 2 shows several performance indicators for the 50×5 nodes scenario: the number of different messages (*mean, max #msg*), the number of copies per message (*median, max # copies*) in the system, and the mean buffer utilization. We sample these values every 60 s, and report *mean values* across five simulation runs with different random seeds (e.g., median # copies is the mean of the medians observed in the five simulation runs, etc.).

In this scenario, 149 messages are spread to the network. Expectedly, we find that random sleep times do not work well for the short radio range, across all routing algorithms, simply due to lack of connectivity; this is confirmed when investigating the overhead for relaying messages. Yet, they prove quite workable for the long one. We also observe that, with the short radio range, the random offset in the node locations easily leads to disconnected networks: most messages survive, but only with one or a few copies each, meaning that no spreading takes place.

Comparing the three routing mechanisms, we find that Life manages to keep the largest number of different messages in the system simultaneously and occupies the least amount of buffer space, by keeping a modest number of copies per message. We also notice that the difference between the median and the maximum number of messages is fairly small, indicating a balance ("fairness") among messages. Wave and Epidemic make full use of the buffers but lose more messages over time, with Epidemic performing better with

Routing	Grid offset	Radio range	Sleep cycle	mean #msg	max #msg	median # copies	max # copies	buf util.
Epid.	no	20m	none	27.6	30	90.8	111	0.96
Epid.	no	20m	sync	55.6	62	31.4	33	0.98
Epid.	no	20m	random	149.0	149	1.8	2	0.13
Epid.	no	80m	none	31.8	33	73.0	79	0.95
Epid.	no	80m	sync	47.2	52	53.0	58	0.98
Epid.	no	80m	random	59.8	64	35.2	42	0.96
Epid.	20m	20m	none	135.6	142	9.0	11	0.59
Epid.	20m	20m	sync	142.6	148	8.4	9	0.59
Epid.	20m	20m	random	149.0	149	1.0	1	0.09
Epid.	20m	80m	none	30.6	32	82.8	93	0.95
Epid.	20m	80m	sync	46.6	50	50.2	57	0.98
Epid.	20m	80m	random	58.8	69	35.6	44	0.97
Life	no	20m	none	54.0	58	41.8	43	0.84
Life	no	20m	sync	84.8	89	21.0	23	0.87
Life	no	20m	random	149.0	149	1.8	2	0.13
Life	no	80m	none	105.4	109	18.0	19	0.74
Life	no	80m	sync	114.8	117	16.6	18	0.73
Life	no	80m	random	83.4	92	22.2	27	0.85
Life	20m	20m	none	144.2	147	7.4	9	0.49
Life	20m	20m	sync	146.0	149	8.0	9	0.52
Life	20m	20m	random	149.0	149	1.0	1	0.09
Life	20m	80m	none	106.2	107	18.2	19	0.74
Life	20m	80m	sync	114.4	118	16.4	17	0.73
Life	20m	80m	random	81.0	87	25.2	27	0.85
Wave	no	20m	none	42.0	46	66.4	69	0.99
Wave	no	20m	sync	46.6	51	57.0	75	0.98
Wave	no	20m	random	149.0	149	1.8	2	0.13
Wave	no	80m	none	35.8	39	72.6	80	0.99
Wave	no	80m	sync	34.2	38	82.6	92	0.99
Wave	no	80m	random	39.4	44	77.2	82	0.97
Wave	20m	20m	none	143.2	146	9.6	11	0.59
Wave	20m	20m	sync	144.2	147	9.2	11	0.59
Wave	20m	20m	random	149.0	149	1.0	1	0.09
Wave	20m	80m	none	36.0	38	70.8	75	0.99
Wave	20m	80m	sync	33.8	36	83.2	89	0.99
Wave	20m	80m	random	42.4	48	74.2	81	0.97

Table 2: Results for the 50×5 node topology.

and Wave better without synchronized sleep cycles.

The good performance of Life routing comes at the expense of messaging overhead (not shown), especially when nodes do not sleep. Life is in the same order of magnitude as Epidemic routing, requiring more replications for 20 m radio range and less for 80 m. Both their overheads are an order of magnitude higher than Wave routing for the non-sleep case and 20-30% for the setups with workable sleep cycles. This not surprising due to very nature of the algorithm where copies are quickly deleted and regained—which helps low buffer occupancy and message circulation. The overhead for the Wave algorithm stays within a factor of three across all scenarios (from some 100K through some 300K message replications over six hours, i.e., 65–200 messages sent per node per hour); compared to up to 2.2M messages for Epidemic and Life. However, the Life would allow for less frequent rounds and thus could be tailored to reduce the overhead. Doing so is subject for further study, as is understanding the impact of non-synchronized rounds and improving operation with long random sleep cycles.

We omit result details for the 100×10 scenario due to space constraints. In brief, in this four times denser scenario, Life clearly outperforms Epidemic and Wave routing across all settings, in most cases by a factor of two or more. We also find performance to be less uniform. Epidemic performs generally better than Wave routing. The relative overhead of the algorithms is similar as above.

6. CONCLUSION

We have applied DTN-style information sharing using only local state to a specific type of often (yet not necessarily) dense, largely connected, and mostly static networking environments for which we pick beaches as one example. We have devised two dedicated routing algorithms: Life, based on the well-known Conway's Game

of Life cell automaton, and Wave, and performed an evaluation of their performance under varying conditions, comparing them to Epidemic routing as a reference.

We find that the algorithms succeed in spreading and keeping messages across a wide range of scenarios unless the network is (constantly) too disconnected; obviously, if static nodes are just out of reach in space or out of sync in time, not much can be done. Our future work will add node dynamics such as arrivals and departures.

Life, with parameters chosen in a very idealized simple simulation setup, performs best across virtually all scenarios that appear workable at all. This comes at the cost of significant messaging overhead, especially in permanently connected scenarios due to messages moving constantly back and forth between nodes—as can easily be observed by inspecting the GUI of the ONE simulator. Wave, on the other hand, manages to keep the number of message transmissions down with its custody and immunity schemes, but it does not circulate messages as well as Life. We are investigating how to integrate suitable transmission damping elements into the Life algorithm, possibly borrowing from Wave. We also seek to better understand why the performance of Life and Wave vary a lot across different settings and improve them to achieve more predictable results. To this end, future work includes more detailed performance investigation, especially, analyzing the *the mean acquisition time* for a node to get a message, a more comprehensive *coverage* metric, and the *routing efficiency*.

Our beach model represents an "open" DTN system in which nodes come and go, which has not received that much attention yet (except implicitly when using mobility traces). Along with subtly more mobility, we intend to study this aspect of opportunistic networks further and we are interested in other instances of this class of environments, e.g., a "strip" surrounding a lake or a stadium.

Acknowledgment

This work was partly funded by the Academy of Finland in the RESMAN project (grant no. 134363).

7. REFERENCES

[1] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, August 2002.

[2] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proc. of ACM SenSys*, 2008.

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

[4] Chih fan Hsin and Mingyan Liu. Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithms. In *Proc. of ACM IPSN*, 2004.

[5] J. J. Garcia-Luna-Aceves and Ewerton L. Madruga. A multicast routing protocol for ad-hoc networks. In *Proc. IEEE INFOCOM*, pages 784–792, 1999.

[6] Martin Gardner. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, pages 120–123, October 1970.

[7] Olafur Helgason, Emre Yavuz, Sylvia Kouyoumdjieva, Ljubica Pajevic, and Gunnar Karlsson. A mobile peer-to-peer system for opportunistic content-centric networking. In *Proc. of ACM MobiHeld*, 2010.

[8] Esa Hyytiä, Jorma Virtamo, Pasi Lassila, Jussi Kangasharju, and Jörg Ott. When does content float? characterizing availability of anchored information in opportunistic content sharing. In *IEEE INFOCOM*, Shanghai, China, April 2011.

[9] A. Kandhalu, K. Lakshmanan, and R.R. Rajkumar. U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proc. of IPSN*, 2010.

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

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

[12] Santosh Kumar, Ten H. Lai, and Jozsef Balogh. On k-Coverage in a Mostly Sleeping Sensor Networka. In *Proc. of ACM MobiCom*, 2004.

[13] V. Lenders, M. May, G. Karlsson, and C. Wacha. Wireless ad hoc podcasting. *ACM MC^{2}R*, Jan. 2008.

[14] Jo Agila Bitsch Link, Christoph Wollgarten, Stefan Schupp, and Klaus Wehrle. Perfect Difference Sets for Neighbor Discovery Energy Efficient and Fair. In *Proc. of ACM ExtremeCom*, 2011.

[15] Jörg Ott, Esa Hyytiä, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. Floating content: Information sharing in urban areas. In *Proc. of IEEE PerCom*, 2011.

[16] Jörg Ott, Dirk Kutscher, and Christoph Dwertmann. Integrating DTN and MANET Routing. In *Proc. of ACM CHANTS workshop*, 2006.

[17] C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001.

[18] Charles Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequences Distance-Vector Routing (DSDV) for Mobile Computers. In *Proc. of ACM SIGCOMM*, 1994.

[19] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Experimental RFC 3561, July 2003.

[20] Marina Petrova, Lili Wu, Matthias Wellens, and Petri Mähönen. Hop of No Return: Practical Limitations of Wireless Multi-Hop Networking. In *Proc. of RealMAN workshop*, 2005.

[21] Thomas Repantis and Vana Kalogeraki. Data dissemination in mobile peer-to-peer networks. In *Proc. of MDM*, 2005.

[22] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.

[23] A. Villalba Castro, G. Di Marzo Serugendo, and D. Konstantas. Hovering information: Self-organizing information that finds its own storage. Technical Report BBKCS707, School of Computer Science and Information Systems, Birkbeck College, London, UK, November 2007.

[24] Wei Wang, Vikram Srinivasan, and Mehul Motani. Adaptive contact probing mechanisms for delay tolerant applications. In *Proc. of ACM MobiCom*, September 2007.

[25] John Whitbeck and Vania Conan. HYMAD: Hybrid DTN-MANET Routing for Dense and Highly Dynamic Wireless Networks. In *Proc. of IEEE AOC workshop*, 2009.

[26] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[27] Eiko Yoneki, Pan Hui, S. Chan, and Jon Crowcroft. A Socio-Aware Overlay for publish/subscribe communication in delay tolerant networks. In *Proc. of ACM MSWiM*, 2007.

[28] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. Multicasting in Delay Tolerant Networks: Semantic Models and Routing Algorithms. In *ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, 2005.