Messages in a Bottle

Ari Keränen
Ericsson Research Finland
ari.keranen@ericsson.com

Jörg Ott
Aalto University Comnet
jo@comnet.tkk.fi

ABSTRACT

Drift bottles and drift buoys have been used to explore currents and, more recently equipped with sensors, also to record other data about oceans, lakes, and other waters—and occasionally to carry messages or small items in a rather unpredictable fashion. (Moored) buoys may also be equipped with network interfaces to transmit their results or serve as network infrastructure to relay data from, e.g., underwater nodes. In this paper, we explore a low-cost variant for surface-to-surface communication on the water using messaging bottles, characterize their performance in different setups, and provide an implementation in the ONE simulator based upon our findings and a simple drift model for a river.

1. INTRODUCTION

Wireless connectivity has been expanding into remote areas of the planet. The main use cases have been in providing Internet access (in remote villages, on aircrafts, on ships, in vehicles) or collecting measurement data (in the savanna [8], in lakes [13], in the ocean [14], in igloos and snowfields [1, 5], in mines [6]). Commercial Internet access is mostly provided using networks that feature instant end-to-end connectivity, e.g., using satellites, cellular networks, or directed wireless links, but a few exceptions have seen some degree of deployment, e.g., in Lapland [12] or India [7]. While Internet access is usually driven by the user expectations of virtually instant information access, other applications such as environmental sensing do not impose such constraints. Especially for "challenged environments", delay-tolerant or opportunistic networking technologies come into play to reduce the demand on network connectivity and the required infrastructure in order to enable communication where there is no commercial case to be made and hence would not have any network otherwise.

One such area is networking on water. While ships and large-enough buoys can use satellites or long-range radio for communication,[1] smaller devices with limited energy resources may not be able to fit and/or power the necessary network equipment. Some coastal waters (such as the Finnish archipelago) and lakes may be covered by cellular networks so that also smaller, low-power devices could communicate, but this is not the general case. Providing connectivity between stationary, drifting, or controlled moving objects under water and on the surface has been studied extensively in the past. Underwater links use acoustic modems whereas above surface communication usually relies on dedicated radios designed for much larger systems, both specialized and quite expensive equipment.

In this paper, we explore a simple message bottle design using off-the-shelf components to understand its feasibility for simple, time-limited communication support tasks such establishing a temporary capillary network [2] in an area with limited drift or carrying messages downstream a canal, river, or modest rapids. Such inexpensive message bottles may be used to collect sensor data from otherwise hard to access sources, to assist in information sharing in otherwise insufficiently connected environments, and even to provide covert channels for censorship resistant communication.[2] The main point is understanding the networking aspects, so we leave the choice of rugged parts required by some of these environments for future work. We describe our component choices and design in section 2 and carry out initial experiments to characterize the networking performance of the message bottles in section 3. In section 4, we report on extensions to the ONE simulator [10] we developed based upon our insights, to which we also add an initial movement model for objects drifting in a section of a river to understand the emerging connectivity patterns. Section 5 concludes this paper with a brief summary, points out implications for opportunistic system design, and briefly discusses some engineering aspects as well as future work in experimentation and validation.

2. MESSAGE BOTTLE DESIGN

2.1 Hardware

As noted above, our initial design targets feasibility studies, not robustness, and hence we focus on inexpensive off-the-shelf components. We choose a Raspberry Pi with an 8 GB SD card that is powered by an EasyAcc 12,000 mAh external battery pack. We experiment with two different WLAN adapters: the Netwjork PICO 150Mbps USB WLAN adapter due to its fit small form factor (referred to as "small") and the Conrad WLAN USB antenna stick N150 ("large").[3] These components are shown in figure 1 (left).

The enclosure, our *Message Bottle*, is a simple glass jar with a sufficiently wide opening for to fit all components in; a final design

[1] See, e.g., http://www.jcommops.org/dbcp/platforms/types.html

[2] We note that to avoid environmental pollution, we are interested in scenarios wheres the message bottle can be recovered in the end, but not necessarily during their operational life.

[3] The latter of which even allows connecting an external antenna, but we do not make use of this feature.

ExtremeCom '14, August 11-15, 2014, Galapagos Islands, Ecuador.

would use plastic storage jars to avoid the risk of breaking. To ensure protection of our electronics, we place the equipment in a zip-lock bag and buffer it inside the glass jar with bubble foil. The resulting Message Bottle is depicted in figure 1 (right). We do not yet attempt to stabilize the bottle using extra weights to ensure that the antenna would usually be above the water surface.

Figure 1: Raspberry Pi hardware in a waterproof container

2.2 Messaging Software

We use the Java-based SCAMPI router implementation [15] that is compliant with the delay-tolerant networking architecture developed in the DTNRG [3] and implements the Bundle Protocol [17] and the TCP convergence layer [4]. In addition, the SCAMPI router uses a number of mechanisms for neighbor discovery and provides a key-value structure for extensible application layer messaging on top of DTN bundles, suitable for implementing content-centric addressing, publish/subscribe operation, and other mechanisms for content sharing. The SCAMPI router also runs on Android devices and can communicate over arbitrary IP networks. Thus, mobile phones can post messages to or retrieve them from the bottles and could provide an uplink to the Internet via a cellular network, if available, to support simple capillary network scenarios.

Message Bottles can operate in three modes: 1) As drifting liberouters [9], i.e., WLAN access points with supplementary opportunistic networking software, and allow mobile devices to connect to them when they come in range; 2) as WLAN clients looking for access points to connect to; and 3) in ad-hoc mode so that they can actually connect to each other, which isn't possible for modes 1) and 2). However, ad-hoc mode severely limits interaction with other mobile devices such as mobile phones and WLAN access points. If we want to enable bottle-to-bottle communication, we can equip each Message Bottle with two wireless interfaces and run one as access point and the other as station adapter (as we have done in our mining system [6]). However, this increases energy consumption and cost and may increase the form factor too. Another alternative is running, e.g., WLAN-Opp to dynamically switch between access point and station adapter mode [18]. For our experiments in this paper, we follow the latter idea but use static mode assignments.

The SCAMPI router relies on on UDP for neighbor discovery and on TCP for exchanging control information and messages. To simplify instrumentation, we carry out the following experiments using existing tools for measuring TCP, UDP, and ICMP performance. Separate experiments showed the impact of the overhead of the bundle protocol and the control plane implemented in the SCAMPI router: (1) The control channel overhead for two nodes to determine which messages to exchange is clearly visible and (naturally) a function of the number of messages to be exchanged per time interval and thus the message size: for example, using 10 KB messages instead of 1 MB messages reduces the effective message transfer rate by a factor of ten, measured for continuous message exchange for 60 s. (2) The actual transfer of each individual message is not substantially affected, i.e., the (constant) overhead of the bundle protocol, the TCP convergence layer, and the router implementation do not have much impact the achievable transmission rate over the wireless interface. Since the overheads are independent of using message bottles, we choose a simplified setup for our initial evaluation. However, we realize that wireless channel conditions impacting the control channel handshakes could slow down message exchanges. We are working on controlled experiments for a complete system evaluation (including bottle-to-bottle and bottle-to-shore messaging).

3. EXPERIMENTS

We are interested in the communication performance of our message bottles when using WLAN in terms of communication range, UDP performance and Round-trip Time (RTT) as a measure for node discoverability, and achievable TCP throughput representing messaging over the TCP convergence layer.

3.1 Setup

We carried out a series of experiments with two message bottles in July 2013 in two different locations:

1. Land: As initial feasibility evaluation, we performed "dry runs" in a residential neighborhood in Espoo, Finland. We place the bottles a) in about 25 cm height (using plastic buckets) and b) on the fairly plain sandy ground. We place the bottles with the antennae facing each other in increasing distances: for 1–10 m in increments of 1 m and then for 10–30 m in increments of 5 m. This outdoor environment sees interference from numerous (about 10) other WLAN access points in the neighboring homes, but as we do not aim at measurement precision but rather validate the setup for the experiments in the water, their presence is not an issue. We do not report quantitative details on these runs.

2. Sea: We deploy the message bottles in the shallow waters of a beach (Mellsten, Espoo, Finland) in the Baltic Sea about 5–6 m from the shore. We chose the open Baltic Sea so that we won't have any interference from other WLANs (we validated the absence of other networks) and that there won't be any reflections (e.g., from walls or roofs of swimming halls) that could assist communications. We manually keep the bottles "roughly" in their respective location, allowing them to drift and turn while floating. In this setting, we carry out measurements in distances of 3–20 m; we stop at 20 m as this appears to be limit to get connectivity for our message bottles. At the time of our measurements in the sea, the sea was very calm with very small waves (rather ripples; amplitude $<$10 cm). We also perform reference measurements on the (not quite plain) sandy beach with the bottles in 1 m and 20 m distance.

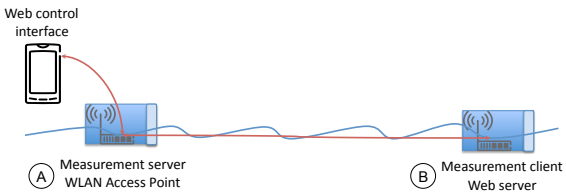

Figure 2: Measurement setup

We use one of the standard Raspian distributions (3.6.11+, 474) on the Raspberry Pis and configure one of them (A) to act as WLAN

Test	Parameters	Duration
ping	packet sizes 64, 512, 1024, 1500 bytes	30 packets in 6 s
TCP SEQ	B→A, A→B sequentially	30 s each direction
TCP SIM	A↔B simultaneously	30 s total
UDP SEQ	B→A, A→B, 1 Mbit/s sequentially	30 s each direction
UDP SIM	A↔B, 1 Mbit/s simultaneously	30 s total

Table 1: Measurements series per experiment.

access point and also to run a DHCP server. This device serves as the measurement server. The other Raspberry Pi (B), the measurement client, is configured to automatically connect to the access point based upon a well-known SSID. We assign static IP addresses for both devices and run an Apache web server (2.2.22) on the measurement client, which uses a PHP script to run the measurements. The index web page displayed allows configuring a measurement context and features buttons to label measurements with the respective distances and start the measurements. We use a waterproof mobile phone to connect via the access point to the web server and invoke the measurements/

We use the tools *ping(1)* and *iperf(1)* for carrying out the actual measurements and log their output. The measurement server runs iperf daemons for TCP and UDP; the corresponding iperf clients and ping are invoked on the measurement client. The set of measurements we run for each configuration is shown in table 1. Before each measurement, a script validates that the client is still connected to the access point and waits until it reconnects if necessary.

3.2 Measurement Results

We first look at basic connectivity using simple ping messages of different sizes. In our first set of experiments, we use the small USB WLAN adapters, which are interesting because of their form factor. We measure the performance when placing the message bottles on a flat ground (sand, grass) and when fixing them about 25 cm above ground. We find that connectivity works well only up to 10 m and may exhibit substantial variation in round-trip time. While performance benefits benefits substantially from placing the nodes at a small elevation above ground, doing so isn't feasible for small message bottles. Also, a range of 10 m appears fairly sufficient for our purposes. To validate the limitations, we performed the experiments with the small antenna configuration in the sea, but this setup did not yield connectivity across more than 2 m distance and is thus confirms that this is not suitable for our bottle-based messaging. In the remainder of this paper, we therefore focus on the large antenna configuration.

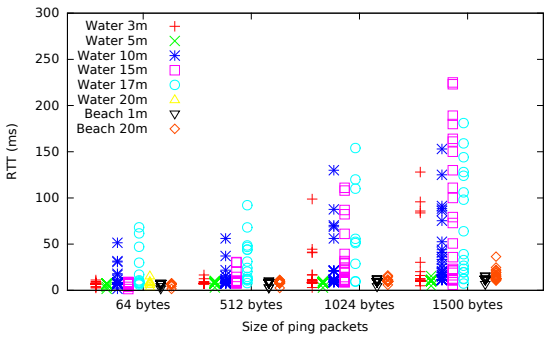

Figure 3: RTTs of individual ping packets for different packets sizes (64–1500 bytes) and different distances between the message bottles in the Baltic Sea with two reference measurements on the beach.

With the large antenna configuration, we obtain connectivity up to 30 m when placing the bottles on the ground (no bucket needed). This configuration also turns out to be feasible for measurements in the Baltic Sea and we obtain basic connectivity up to 20 m as the ping measurements shown in figure 3 indicate. We observe that smaller packets have lower RTTs (and also lower loss rates on the water) due to fewer L2 retransmissions. This suggests that especially peer discovery packets should be kept small and that it may be advisable to use smaller TCP segments than the typical MTU size of 1500 bytes. Nevertheless, we leave detailed segment size studies for future study and use an MTU size of 1500 bytes for the TCP measurements we report on below.

Since transmitting larger messages requires some degree of reliability and our SCAMPI router platform implements the TCP convergence layer, we now look at TCP performance. Figures 4a) and b) show the spread of instant TCP throughput as reported by iperf in one-second intervals for the TCP SEQ and TCP SIM. The transfer rates in both directions (B→A; A→B) are reported together per distance as and also the mean transmission rate is calculated across both directions.

Expectedly, the obtained transmission rate per direction is larger (roughly by a factor of two) for the case, when A and B do not transmit at the same time (TCP ALT) compared to simultaneous transmission (TCP SEQ). This becomes a bit more pronounced with growing distance. The data rates decline with distance for both TCP SEQ and TCP SIM and, are close to the reference measurement on the beach only for 3 m distance. The connectivity at 15 m distance and above becomes highly unstable so that measuring required often multiple attempts. The instant data rates begin fluctuating at 10 m so that the measured data rates should only be taken to indicate that communication is possible but not suggest a rate. We also clearly see how the spread of data rates grows with distance and connectivity degrades starting around 10 m.

Figure 4c) closer investigates one run of selected measurements in the sea by looking at TCP performance over time for the TCP SEQ case (the results for TCP ALT are qualitatively similar): this figure shows that the broader spread of instant data rate samplings occurs across the entire measurement. The figure also hints that repeated packet losses, which appear random and thus due to channel errors, lead to timeouts and make rate recovery more difficult at larger distances.

Figure 5 shows the data volume that was transmitted in each direction, both simultaneously and sequentially during 30 s (or until the respective connection broke). When transmission happens in sequence (TCP SEQ), we find that the transmission direction appears to have little impact, some time-varying influences (e.g., wave patterns, turning bottles) aside, which manifest primarily for larger distances. For simultaneous transmission, it appears consistently in our experiments that sending data to the AP node (A) performs better than sending data from it, which requires further exploration. Repeated experiments (also on land) show that, for TCP SIM, both TCP connections used by iperf start out similarly, but within some five to ten seconds either direction may gain over the other for a period of time, leading to the asymmetric in transmission rate and data volume. Which connection performs better may change during the experiments, but often the first asymmetry lasts until the end of the 30 s measurement. This appears to happen more frequently (but not always) for the direction B←A. We note that this asymmetry may be related to iperf using two independent TCP connections (which increases the load as ACKs of one direction cannot be piggybacked onto data segments in the other). This requires further investigation.

For UDP, we use the iperf default packet size of 1470 bytes and

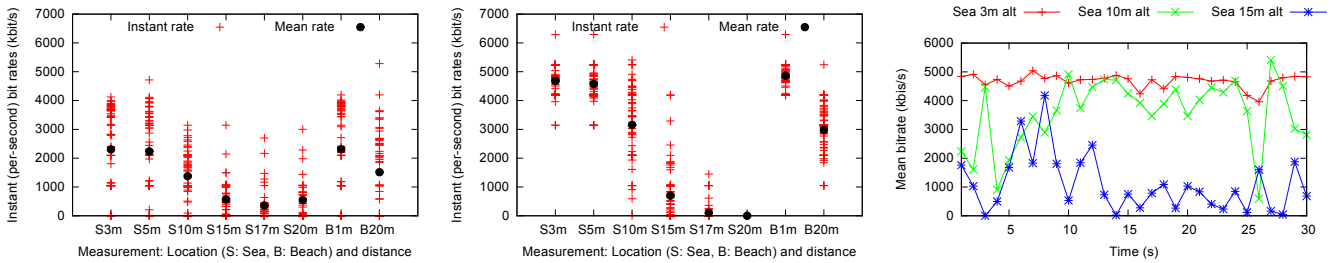

Figure 4: Instant TCP data rate per one-second interval and mean data rate, for both directions: (a) simultaneous data transmission A↔B (TCP SIM, left) and (b) sequential data transmission B→A; A→B (TCP SEQ, middle). (c) Data rate variation over time for TCP SEQ (right).

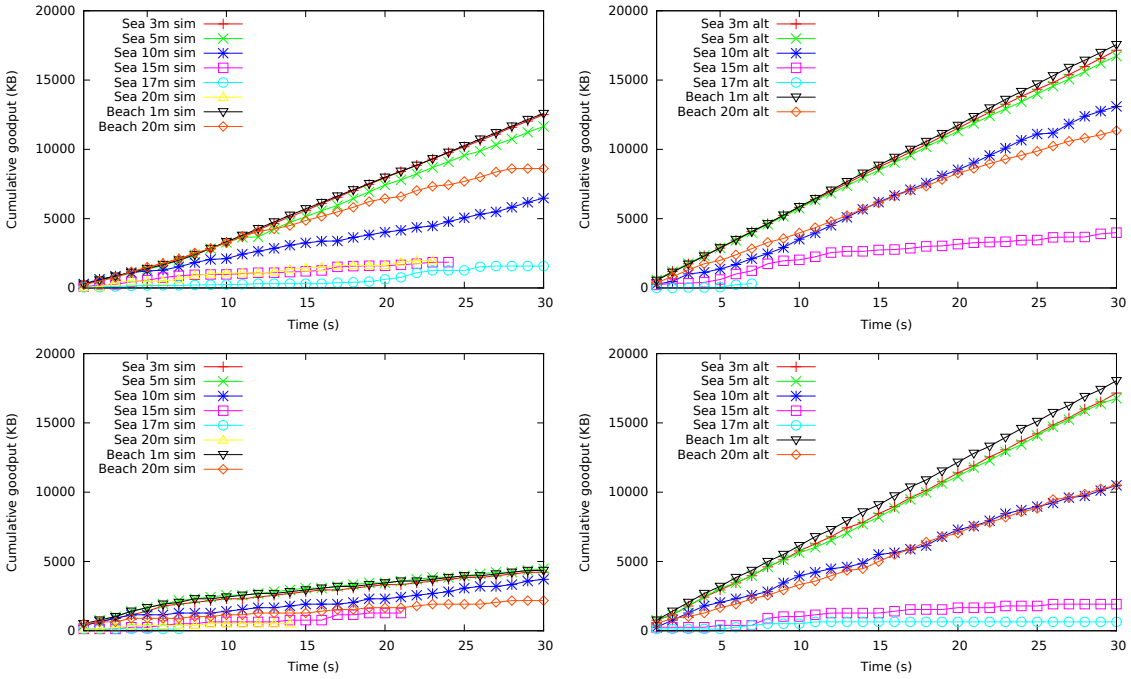

Figure 5: Cumulative TCP data volume transmitted during 30 s for simultaneous (TCP SIM, left column) and sequential transmission (TCP SEQ, right colum). The top row shows the transmission direction A→B and the bottom row B→A. Note that some connections broke during the respective experiment.

fix the data rate at 1 Mbit/s, which is notably lower than we could achieve with TCP, so that the measurements can focus channel- induced packet loss. Figure 6 shows that this target rate is kept for short distances with virtually no packet loss. Again, variations begin showing at 10 m and performance starts to degrade at 15 m, consistent with the ping-based RTT measurements and the TCP measurements. Receive data rates above 1 Mbit/s simply indicate that packets got grouped due to link layer retransmissions; rates lower than 1 Mbit/s stem partly from similar effects, but are also due to packet losses, which become significant for distances of 15+ m as the figure also shows. The resulting cumulative data volume (not shown) is less diverse than for TCP due to the rate limit (which also limits the data volume to 3.75 MB). Again, the connectivity does not last throughout the experiments for 15 m and above.

Overall, in calm waters, our message bottles can communicate as soon as they get within 15 m of each other. Communication performance shows that data rates of close to 1 Mbit/s are achievable at this distance and that several megabytes of data can be transferred between bottles in 30 s. Since drift is usually expected to be slow, this should suffice for two bottles to exchange a substantial number of messages. With this, running the SCAMPI router in message bottle appears promising, but future experiments are needed to undertand the achievable messaging performance of the complete system.

3.3 Limitations

The above measurements provide some insight into the communication performance of our *specific* message bottle design. The measurements were taken on three days on land and on two days in the sea, with especially the latter certainly biased towards (sunny) weather conditions, particularly wind and waves (or lack thereof). The measurements only capture certain data points (TCP and UDP with certain packet sizes) and traffic patterns. To be able to characterize the link performance, we did not use the SCAMPI router but rather dedicated measurement tools. All individual measurements were of limited duration of up to 30 s; using longer durations in

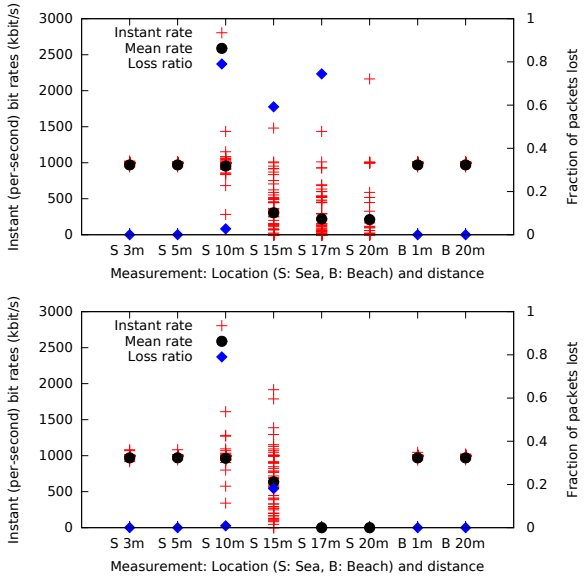

Figure 6: Instant UDP data rate per one-second interval, mean data rate, and fraction of packets lost, averaged over both directions: (a) simultaneous data transmissions (top) and (b) sequentially alternating transmission (bottom). When the lost packets could not be determined reliably (because reports are missing), the corresponding data point is omitted.

early trials yielded 15–20 min per experiment, which we cut back to some 10 min to allow for sufficient measurement diversity within finite time. Since our aim was understanding feasibility and not a representative characterization (we did not have control over the environment in the first place), we only performed repetitions if measurements failed.

In spite of these limitations, our experiments confirm the basic feasibility of our message bottle design and our measurement results appear sufficiently consistent that we can build a simple first model for the ONE simulator.

4. SIMULATING MESSAGE BOTTLES

The movement of nodes floating on water is mainly governed by the surface currents of the water. While measurements of surface current and drifting behavior and modeling such is done, e.g., in oceanography, the corresponding models can get quite sophisticated and are (naturally) highly specific to the respective environment. To keep our initial simulation setup simple, we start out with a specific trace set covering trajectories in a river, as has been studied studied in [11]. The authors derive a simulation model based on the collected GPS traces of freely floating devices. We implemented the simulation model to the ONE simulator in order to perform initial evaluations of scenarios with drifting nodes.

The simulation model is based on sampling simulation parameters from parameter maps. The authors of [11] collected 45 trajectories of floating devices with GPS modules that recorded their way in a river segment at Hsinchu, Taiwan. The collected location, direction, and speed data is transformed into three different parameter maps: 1) point map for relative probability of finding a floating device at each location 2) velocity map for speed of the surface current at each location 3) direction map for the general direction of the current at each location. The parameter maps are divided in cells with size of $1.85m \times 1.85m$ and for each cell general speed and velocity, and also the probability of finding a node is calculated. These parameter maps were kindly shared by the authors of the study and used as input data for the implementation in the ONE simulator.

We also implemented a new simulated wireless network module that supports different transmission speeds based on the distance between the nodes, which we parameterize based our measurements, but which can use arbitrary connectivity characteristics. Since we are modeling reliable transfer, we use the data from the alternating TCP transfer speed measurement at sea as the input for simulations. A screen capture of a resulting simulation is shown in figure 7.

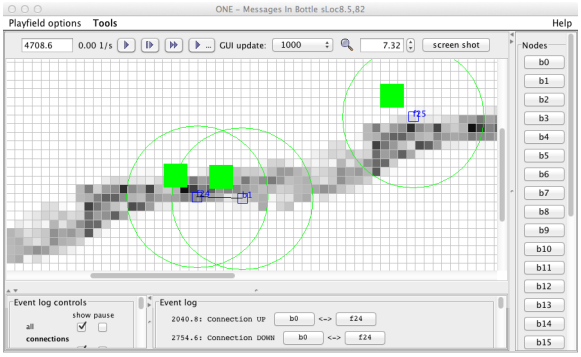

Figure 7: Screenshot of a simulation scenario

These two new extensions to ONE allow us to experiment with a river scenario with relatively realistic mobility and networking behavior. Currently implemented simulation scenarios allow 1) experimenting with message exchange with bottles drifting in a group 2) drifting bottles collecting data from stationary nodes on the river (as shown in figure 7) 3) both previous scenarios but with stationary or drifting nodes capable of connecting both to the bottles and to the Internet (i.e., capillary network scenarios).

Unfortunately the river segment for which we have the parameter map data is fairly short (only roughly 100 meters), and therefore limits our simulation scenarios. In order to explore scenarios with longer river segments we duplicated and concatenated the river segment 5 times and added factors of randomness to the speed ($\pm 5\%$) and direction (± 20 degrees).

We placed 3 stationary nodes (representing e.g., environmental sensing devices) in the river, roughly 50 meters apart, and simulated releasing drifting nodes, once every 30 minutes, from a location about 100 meters upstream. We measured the attainable connection volume, i.e. the potential amount of data that the stationary nodes could transfer to the drifting nodes, during a 12 hour simulation. If the drifting nodes have cellular connectivity and form a temporary capillary network for the stationary nodes, the connection volume would correspond to the amount of data they could directly transfer to or from the Internet. In order to evaluate the impact of release location to connectivity, we run the simulation with 10 different start locations, each 2 meters apart. At those locations the nodes were released in a randomly selected spot within a 2×2 m area.

Additionally we placed one collector node about 100 meters downstream after the last stationary node. If the drifting nodes have only short-range radio connectivity, the collector node would take care of communication to the Internet. For this scenario the fixed nodes generate one small (1kB, e.g., an environmental measurement data point) message every 10 minutes, except for the first and last hour of the simulation. For message routing, we used SprayAndWait router in binary mode with (maximum) 4 copies. The fixed nodes were configured to not accept messages from other nodes. There-

fore the first drifting node can potentially also send the message to another drifting node and the following one would deliver it only to the final destination. The messages are then delivered by the drifting nodes to the collector node.

Our initial simulation experiments show that the drifting nodes are in the radio range of the fixed nodes usually either around 90-150 seconds (37% of connections) or 250-400 seconds (57%). In a few occasions (5% of the connections) the drifting nodes get stuck close to a fixed node and in these cases the connections lasted 650-900 seconds. The connections resulted in connection volumes of approximately 45-75 MB, 100-160 MB, and 270-380 MB per connection respectively. This level of connection volumes can easily accommodate for various kind of scenarios, e.g., environmental sensing with high resolution image data.

Average message delivery latency from the fixed nodes to the collector node was in most of the scenarios 1650-1800 seconds. This includes the time a message is waiting at a fixed node for a drifting node to arrive and for the drifting node to reach the collector node. In three scenarios the average latency increased to 2100-2350 seconds due to drifting nodes getting occasionally stuck on the way. These three scenarios were the ones where the drifting nodes were released closer to the east shore of the river and ended up moving a large fraction of the time close to the shore. This seems to imply that also the node release location can have a substantial impact on the message delivery.

5. DISCUSSION AND CONCLUSION

In this paper, we have presented a simple, low-cost prototype for opportunistic communication on the water surface. We have explored the communication capabilities of our message bottle system in calm open waters, finding that data exchange is feasible for distances of up to 15 m at that we can obtain net TCP data rates of close to 1 Mbit/s at 15 m distance. We have incorporated these initial insights into an suitable connection model for the ONE simulator, complemented by a simple version of a drifting movement model for a river, so that we can perform initial simulation-based evaluation of bottle messaging scenarios. In our future research we will expand our simulations to cover various other drifting models and scenarios to further assess the impact of different parameters on the feasibility of a floating and drifting opportunistic network.

In principle, we can run our SCAMPI router unchanged inside message bottles. Yet, our experiments suggest (minor) modifications to maximize the communication performance: (1) To extend reach, the discovery mechanisms and the convergence layers could consider using smaller packets. (2) The frequent disconnections require robustness mechanisms to (aggressively) rediscover peers, suggest minimal connection establishment overhead, and make support for reactive fragmentation advisable. (3) It may be worthwhile exploring different reliable transport protocols (such as LTP over UDP) as an alternative to TCP. Exploring the performance impact of these mechanisms on a SCAMPI router implementation is subject to future work. Further experiments are required to explore TCP-based throughput with the SCAMPI router to understand if they should limit their data exchange to a single TCP connection and/or take turns and in sending messages rather then transmitting simultaneously (over two TCP connections). Sequential operation appears feasible as bottle usually drift slowly.

From an engineering perspective, our message bottle design requires numerous improvements to make it practically more suitable. This includes using more robust bottles (such as plastic ones); fitting the equipment (and possibly adding some weights) in a way so that the antenna remains above the water surface; and fixing and cushioning the equipment to it from impact so that the bottle can be thrown, go down small waterfalls or rapids, and collide with other objects without breaking. To extend the digital life of a bottle, using a (more rugged) device with a smaller power footprint and adding energy harvesting could also be subject of future studies. For example, a Telos [16] board could potentially bring us an order of magnitude more standby time and also a smaller form factor.

6. REFERENCES

[1] J. Arkko, A. Keränen, S. Farrell, K. Hartnett, and E. Davies. Demo: Snowcat5 - networking in snow. In *Proc. ExtremeCom*, 2012.

[2] I. Augé-Blum, K. Boussetta, H. Rivano, R. Stanica, and F. Valois. Capillary networks: a novel networking paradigm for urban environments. In *Proceedings of the first workshop on Urban networking*, pages 25–30. ACM, 2012.

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. RFC 4838, Apr. 2007.

[4] M. Demmer, J. Ott, and S. Perreault. Delay Tolerant Networking TCP Convergence Layer Protocol. Experimental RFC 7242, June 2014.

[5] M. Doering, S. Rottmann, and L. Wolf. Demonstration of a snowpack monitoring system based on inexpensive sensor nodes and solar-powered backhaul links. In *Prof. ExtremeCom*, 2012.

[6] P. Ginzboorg, T. Kärkkäinen, A. Ruotsalainen, M. Andersson, and J. Ott. DTN communication in a Mine. In *Proceedings of the 2nd Extreme Workshop on Communications*, September 2010.

[7] S. Guo, M. Falaki, E. Oliver, S. U. Rahman, A. Seth, M. Zaharia, U. Ismail, and S. Keshav. Design and Implementation of the KioskNet System. In *International Conference on Information Technologies and Development*, 2007.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of the ASPLOS-X conference, San Jose, CA, USA*, October 2002.

[9] T. Kärkkäinen and J. Ott. Liberouter: Towards Autonomous Neighborhood Networking. In *Proc. of IEEE WONS*, 2014.

[10] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[11] H.-C. Lee, C.-Y. Lin, C.-H. Lin, S.-W. Hsu, and C.-T. King. A low-cost method for measuring surface currents and modeling drifting objects. *Instrumentation and Measurement, IEEE Transactions on*, 60(3):980–989, 2011.

[12] A. Lindgren, A. Doria, and M. E. Jan Lindblom. Networking in the land of northern lights: two years of experiences from DTN system deployments. In *Proc. ACM workshop on Wireless networks and systems for developing regions*, 2008.

[13] P. McDonald, D. Geraghty, I. Humphreys, S. Farrell, and V. Cahill. Sensor Network with Delay Tolerance (SeNDT). In *Proc. of IEEE ICCCN*, 2007.

[14] J. Partan, J. Kurose, and B. Levine. A survey of practical issues in underwater networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(4):22–33, October 2007.

[15] M. Pitkänen, T. Kärkkäinen, J. Ott, M. Conti, A. Passarella, S. Giordano, D. Pucinelli, F. Legendre, S. Trifunonic, K. Hummel, M. May, N. Hedge, and A. Spyropoulos. SCAMPI: Service platform for soCial Aware Mobile and Pervasive computIng. In *Proc. of the ACM SIGCOMM workshop on Mobile Cloud Computing (MCC)*, August 2012.

[16] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369. IEEE, 2005.

[17] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050, November 2007.

[18] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre. WiFi-Opp: Ad-Hoc-less Opportunistic Networking. In *Proc. of ACM MobiCom CHANTS workshop*, Sep 2011.