WONS 2017 1570315160

The More You Speak, the Less You Hear: On Dynamic Announcement Intervals in Wireless On-demand Networks

Lars Baumgärtner*, Pablo Graubner*, Jonas Höchst*, Anja Klein†, Bernd Freisleben*†

*Department of Mathematics & Computer Science, Philipps-Universität Marburg, D-35032 Marburg, Germany
E-mail: {lbaumgaertner, graubner, hoechst, freisleb}@informatik.uni-marburg.de
†Department of Electrical Engineering and Information Technology, TU Darmstadt, D-64283 Darmstadt, Germany
E-mail: a.klein@nt.tu-darmstadt.de; bernd.freisleben@maki.tu-darmstadt.de

***Abstract*—Several protocols used in wireless networks rely on nodes announcing information to other nodes. This can be illustrated by service announcements sent in *ZeroConf*, routing announcements used in *OLSR*, and peer announcements in wireless peer-to-peer or delay-tolerant networking systems such as *Forban* and *Serval*. The main problem is that these protocols use fixed time intervals between subsequent broadcast announcements. Fixed intervals can either lead to high network load (if the announcement interval is too short) or delay the distribution of information between peers (if the announcement interval is too long). Repeatedly broadcasting announcements after fixed intervals also has a impact on the energy consumption of mobile devices operating in wireless networks. In this paper, we present several approaches to realize dynamic announcement intervals that facilitate fast reception from at least one other node while trying to keep the overall communication overhead as low as possible. Experimental results in terms of performance properties and energy consumption are presented to illustrate the benefits of dynamic announcement intervals in wireless on-demand networks.**

I. Introduction

Several network protocols rely on nodes broadcasting announcements to other nodes. Examples include service discovery (*Bonjour/ZeroConf, Samba*), routing algorithms (*RIP, OLSR*), and peer-to-peer or delay-tolerant networking systems (*Forban*[1], *Serval*[2]). While the traffic generated by periodically sending announcements might be negligible in wired networks with high-speed links, bandwidth in wireless networks, such as 802.11, Bluetooth or various mobile ad hoc networks (MANETs), is precious and limited. For example, spontaneous smartphone networks become more and more important not only by providing pervasive wireless Internet access during large human crowd gatherings, but also during emergency situations or post-disaster recovery [2].

In an emergency communication scenario, the main goal is to spread messages and files produced at a disaster site fast among reachable nodes. Therefore, data is passed around in an epidemic fashion to as many neighboring peers as possible. Typically, some nodes are more static, such as devices of people trapped in their houses or small emergency camp sites forming islands, while other nodes are on the move (by bike, car, foot) that by passing through these islands act as carrier-pigeons to distribute information further (see Fig. 1). To make optimal use of the short time in case of a drive-by, it is important to find a peer for data exchange very fast. Depending on the used wireless technology, a mobile phone might have an effective range of 14-80 meters to communicate with others. Thus, if we assume a WiFi radius of 40 meters and a static node being 10 meters away from a street, a car driving on the street would be in the WiFi range for about 77 meters. The car passing by, assuming it moves at about 50 km/h, would have just under 6 seconds for node discovery and exchange of data (see Fig. 2). Therefore, for the fast moving node one of the more static peers is sufficient to start a data transfer. Since all information gets replicated in this scenario, the fast moving node does not need not know all possible neighbors. The static node can distribute the data further among its neighbors. Discovering *all* direct peers as fast as possible is neither necessary nor beneficial for the static nodes. Under these assumptions, using dynamic announcement intervals instead of the typically used static announcement intervals is reasonable. Furthermore, dynamic announcement intervals require not only less network resources, but also potentially save more battery capacity than static announcement intervals.

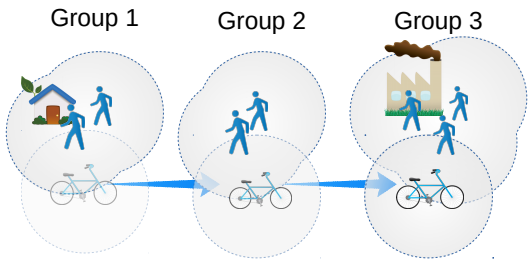


Fig. 1: Drive-by store-and-forward data exchange between three groups.

[1]http://www.foo.be/forban/
[2]http://www.servalproject.org/

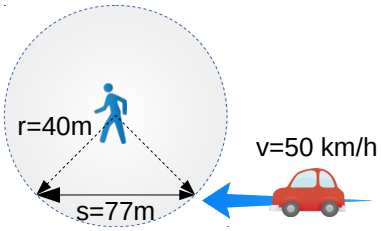


Fig. 2: Drive-by window of opportunity example.

In this paper, we present several approaches to realize dynamic announcement strategies that facilitate fast reception from at least one other node while trying to keep the overall communication overhead as low as possible. Experimental results in terms of performance properties and energy consumption are presented to illustrate the benefits of dynamic announcement intervals in wireless on-demand networks. In particular, the paper makes the following contributions:

- Various strategies for realizing dynamic announcement intervals optimized for different network setups are presented.
- An experimental evaluation of all proposed strategies, including static and random announcement strategies, with respect to bandwidth usage, announcement distribution and energy consumption is presented.
- Test environments suited for various topologies, such as large stable networks, islands merging and networks splitting, are investigated.
- The results are directly applicable for local peer-to-peer content distribution systems in emergency scenarios, such as *Forban* and *Serval*.

The paper is organized as follows. Section II discusses related work. Section III presents the design of our announcement strategies. Implementation details are described in Section IV. The setup of our experimental evaluation is described in Section V, and experimental results are presented in Section VI. Section VII concludes the paper and outlines areas for future research.

II. Related Work

There are several publications that investigated problems associated with static announcement intervals in various protocols and application scenarios.

Natsheh et al. [9] proposed a solution based on fuzzy logic to optimize hello messages in dynamic ad-hoc routing. Their work focused on the mesh routing use case, and experiments with a maximum of 35 simulated nodes were presented. Furthermore, Khalaf et al. [6] investigated the broadcast storm problem in mobile ad hoc networks. The authors presented a probabilistic approach to improve the situation in a mesh routing scenario.

Ahmed et al. [1] addressed the problem of beaconing in vehicular ad hoc networks (VANETs). Combinations of controlling a beacons transmission power, transmission rate, and contention window at the MAC layer were proposed to achieve efficient beacon communication in VANETs. Another approach devoted to improve the problems related to static beacon intervals in ad hoc networks was presented by Tahar et al. [12].

Peng [11] proposed an adaptive mobility-aware MAC protocol for wireless sensor networks. Apart from optimizing the number of messages, the energy consumption was investigated. Lim et al. [7] presented an approach called RandomCast to improve the energy efficiency of 802.11 ad hoc networks. In this approach, the sender can specify the desired level of overhearing of neighboring traffic, trying to find a balance between energy consumption and routing performance.

Peer-to-peer content distribution is another scenario where announcements are relevant, and a trade-off must be made between central tracker-based peer discovery and distributed peer discovery. Dán et al. [4] presented a hybrid approach that uses individual trackers and a gossip protocol to improve peer discovery. By hopping between swarms and redistributing known peers, the efficiency is increased.

Liu et al. [8] developed a censor-ship resistant delay-tolerant network for message exchange and evaluated it with respect to performance and energy consumption. To avoid energy draining broadcasting with fixed intervals, the authors adopted an approach presented by Zheng et al. [15] based on asynchronous wake-ups for ad hoc networks. Another delay-tolerant networking system designed specifically for data synchronization in emergency situations was presented by Paul et al. [10]. While optimizations are proposed to speed up file transfers and syncing, the actual peer discovery was realized by simple broadcasts with fixed announcement intervals.

During an experimental evaluation of *Serval* as a delay-tolerant emergency communication platform, Baumgärtner et al. [3] found that regular broadcasts used for node discovery or announcements of routing and data storage information especially in networks with many direct peers require high network bandwidth. The study showed that around 2 seconds of announcement delay was the best trade-off between quick peer discovery and conserving energy with the stock implementation made available by the Serval Project.

By exploiting social network characteristics for assisting ad hoc peer discovery, Zhang et al. [14] attempted to find optimal beacon probing rates with constant intervals for each group of users. As stated by Wang et al. [13], peer discovery itself can be as energy consuming as making phone calls.

While most of the above mentioned work is highly specific to the studied use cases or applications, the general picture is that adaptive or dynamic announcement intervals usually outperform static ones, not only with respect to network performance, but also regarding energy consumption.

III. Dynamic Announcement Intervals

In this section, we present several dynamic announcement strategies, the constraints associated with them, and quality properties to evaluate their performance.

A. Announcement Strategies

We have developed several strategies for realizing dynamic announcement intervals. Each strategy has access to the current announcement delay, the global number of announcements seen at the last observation interval and the current number of unique peers. Our strategies are described in the following:

1) Static: The *Static* announcement strategy is the basic announcement approach used by most current broadcast protocols. There is a fixed interval defined for every node in which an announcement is sent. This also means that the generated global traffic is growing linearly with the node count. By default, this interval is set to a 2 second delay in our tests.

2) Random: In the *Random* strategy, every node chooses a random announcement delay. This interval is a random number between a minimum of 0.5 and a maximum of 20 seconds for every observation interval. The distribution of the random numbers, depending on the network size, heavily influences the performance of this strategy, as well as the duration of the observation interval.

3) RandomSweet: In this strategy, the *Random* strategy is extended. The announcement interval is only set if the current global announcement rate is higher than one announcement per second or less than 0.5 announcements per second. Thus, if the network has reached an optimum state, this strategy does not change anything and sticks to the last randomized delay for each node. This stabilizes the network if by chance optimal delay combinations are found, at least until nodes join or leave the network.

4) Step: After every observation interval, the *Step* strategy checks the global announcement count. If the count is higher than one announcement per second, the node's announcement delay is increased by one second. If the count is lower than 0.5 announcements per second, the node's announcement delay is decreased. This leads to gradually narrowing down to an optimal announcement delay over time. The strategy is also confined by a minimum announcement delay of 0.5 seconds and a maximum of 20 seconds, based by the chosen observation interval.

5) StepRand: In this strategy, *Step* is extended by adding randomness to each step. While the conditions remain the same as in the basic *Step* strategy, a random value between 0 and 0.5 seconds is added or subtracted to the announcement delay.

6) MaxFirst: MaxFirst is a rather defensive strategy: whenever a high global announcement rate is detected (more than one announcement per second), the node's announcement interval is set to the observation interval, i.e., the maximum possible announcement delay is tried first, hence the name. Then, if less than 0.5 global announcements per second are present, the strategy decreases the announcement delay by one second per iteration, until the local minimum of 0.5 seconds is reached. Thus, a very low announcement frequency is favored, which should be beneficial in larger or fast growing networks.

7) MinFirst: MinFirst reverses *MaxFirst*, and therefore is an aggressive announcement strategy. Whenever less than 0.5 announcements per second are detected globally, the announcement delay is set to the local minimum of 0.5 seconds. If the above condition is not satisfied, the announcement delay is increased by one second per iteration, until the observation interval is reached. This strategy should support scenarios where most of the time only very few peers are in direct vicinity of each other.

8) Unsteady: In the *Unsteady* strategy, each announcement delay is computed only on the basis of the number of unique peers known by a node and not on the global announcement rate like in the other algorithms. The goal is to reach a global rate of one announcement per second. Looking at the current peer count, an announcement interval is computed to complement the announcement intervals of the other nodes. Using this method, the strategy should be able to adapt to new situations as fast as defined by the observation interval.

B. Constraints

To guarantee that a node can be discovered, we define an *observation delay*, with the same value for all nodes. This is the time between re-evaluation and before another change in the announcement frequency can happen. Each node has to announce itself at least once per observation interval. All nodes must set the announcement delay after the observation delay is over. This enables a better comparability between the announcement strategies.

The *observation delay* is set to 20 seconds in all our experiments, since the baseline for static announcements is 2 seconds. Therefore, it seems reasonable to re-evaluate the situation after 10 standard announcements. The higher the delay, the longer it takes for the whole network to adapt to new situations. A very short delay in conjunction with the premise that each node should at least send one announcement per interval leads to higher loads, especially with higher node numbers. Thus, a delay of 20 seconds guarantees that within this interval *all* peers in the direct neighborhood are discovered.

C. Quality Properties

To evaluate and compare different strategies for dynamic announcement intervals, universally applicable quality properties must be defined. Our main goal is to globally have one announcement per second at any given time, not less, but also not much more to conserve resources.

1) Global Announcement Rate: The *Global Announcement Rate* is measured by counting the announcements per second. This parameter is the main optimization goal for our algorithms, since it is directly correlated to the bandwidth used for peer discovery.

2) Global Announcement Gaps: The *Global Announcement Gaps* are measured by the time periods between two announcements. The *Global Announcement Gaps* are important to observe, since they reveal how long a new peer needs until it receives an announcement from the rest of the network.

3) Adaptation Rate: The *Adaptation Rate* represents the time needed for an announcement strategy to adapt to a new situation. It describes the situation that all nodes are started at the same time, and defines the moment, when no significant change in the number of announcements is recognizable.

IV. Implementation

In this section, details about our implementation of the different announcement strategies and the network using them are presented.

A. Mesher

To investigate the behavior of dynamic announcement intervals, we extended a simple broadcast service to provide easily exchangeable announcement algorithms for peer discovery. *Mesher*[3] is a simple local chat written in Google's *Go* language by one of the authors, and therefore is easily extensible. It utilizes broadcast packets for neighbor discovery and for exchanging public chat messages. *Mesher* uses a static announcement interval of 2 seconds in its default configuration, and thus the network traffic is growing linearly with the node count. Each announcement contains the the public key of the sending node, the services provided by the node, 512 bytes random data in order to simulate database states and a cryptographic signature, resulting in 642 bytes per broadcast packet. Other protocols might use larger or smaller announcement packets, depending on the type of state that is broadcasted.

B. Dynamic Interval Computation

To evaluate various interval computation methods including dynamic changes, the corresponding algorithms needed to be easily exchangeable. Therefore, we decided to implement the algorithms using an embedded *JavaScript* engine, and defined an interface to hand over useful information to access it in the main *Go* binary:

- `get_announce_count()`
- `get_and_reset_announce_count()`
- `get_peer_count()`
- `get_announce_delay()`

After analyzing the provided values, the algorithms are able to set a new announcement interval using `set_announce_delay(Int)`.

C. Announcement Strategies in Mesher

For all announcement strategies, we used the same template (see Listing 1) in *JavaScript* where one specific function is responsible for computing any changes. Each strategy gets the current announcement delay and the global number of announcements seen in the last observation interval. This setup proved to be perfect for rapid prototyping of new algorithms without recompilation or modifications of the main binary.

[3]*Mesher*, available online: https://github.com/gh0st42/mesher

Listing 1: Basic layout of the announcement strategies

```
var observation_interval = 20000;
var total_count = 0;
var min_delay = 500;

set_announce_delay(2000);
for(;;) {
  sleep(observation_interval);
  var cur_count = get_and_reset_announce_count();
  var cur_delay = get_announce_delay();

  // call scheduler and set new delay there
  scheduler(cur_count, cur_delay);
}
```

V. Evaluation Setup

To evaluate the different announcement strategies, several setups were used, including emulations with many nodes as well as physical machines connected over various network links.

A. Network Emulation

For network emulation, we selected the Common Open Research Emulator[4] (*CORE*), which is scriptable using *Python* and in this way allows versatile creation of experimental configurations. This system uses Linux and lightweight virtualization to provide a networking testbed for unmodified, regular Linux binaries. All announcement strategies are evaluated under four different network scenarios described in the following:

1) Centralized Network: In the *Centralized Network* configuration, all nodes are connected centrally and hence are located in the same collision domain. This setup is similar to a classic network hub or a local ad hoc wireless network in the sense that each node can directly communicate with all of its adjacent peers. As long as the network is not oversaturated, every node gets the announcements of every other node.

2) Growing Network: In the *Growing Network* configuration, nodes are added periodically to the network. Ideally, the annoucement strategies should adapt to the new situation fast and down-regulate their announcement counts. Each second, a new node joins the network, and adaptation is required to maintain optimal resource usage.

3) Merging Network: In the *Merging Network* configuration, two equally sized *Central Networks* merge at a fixed point in time, doubling their size instantaneously. Using this configuration, adaptation rates for abruptly changing network configurations can be observed.

4) Splitting Network: In the *Splitting Network* configuration, the network is split in two halves at a fixed point in time. By creating two independent networks, the announcement strategies need to react fast to satisfy the defined quality properties and avoid prolonged periods of silence between announcements.

[4]CORE: http://www.nrl.navy.mil/itd/ncs/products/core

B. Physical Testbed

To evaluate the proposed announcement strategies under realistic conditions, a physical testbed was created. It consists of several Raspberry Pi 3 Model B[5] single-board computers, running under the vendor-provided Debian-Linux-based Raspbian[6] operating system. This allows us to obtain realistic energy and power consumption measurements when evaluating the announcement strategies.

We set up eight Raspberry Pis as network participants, as well as an additional Raspberry Pi as a system under test (SUT). The energy consumption of the SUT was measured using an *Odroid Smart Power* measurement device, an external power meter. The data points were logged at 5 Hz to another device, in order to prevent disruption of the measurement.

VI. Experimental Evaluation

In this section, the announcement strategies described in Section III are evaluated using the network configurations of Subsection V-A. Based on the quality properties of Subsection III-C, the strategies are compared to each other.

To test our strategies under various circumstances, the centralized network configuration was evaluated with different node counts. For each of the eight announcement strategies, the tests were performed using 2, 5, 10, 25, 50, 100 and 200 nodes, resulting in 56 configurations. These configurations were each executed using two node starting mechanisms: a) the batch node start, in which all nodes were started randomly in the observation interval window; b) the delayed node start, where a node was added every second, resulting in a linearly growing network.

In addition, two dynamic network configurations were used: *Split*, where the central network was split in two halves, and *Merge*, where two equally sized networks were joined. Summing up the different configurations, 224 independent experiments were performed.

A. Basic Capabilities

In Figure 3, the announcement rate for all strategies in a 25 nodes static network is visualized. The strategies share the same observation interval, and therefore the first 20 seconds are the same, since they also start with the same announce interval of 2 seconds. The *Static* strategy preserves this announcement interval, and the globally generated traffic remains the same for the whole experiment.

Unsteady and *MaxFirst* show similar, very low announcement rates in this network configuration. *Unsteady* uses the node count as described in Sec. III-A8 and computes its maximum announcement delay, which in this case is greater than the observation delay and sets this maximum. *MaxFirst* jumps to the maximum possible announcement delay, since the observed announcement count is high. Since the situation does not change, both algorithms stick with their decision in future observations. This similarity changes for lower node counts.

[5]https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[6]https://www.raspbian.org

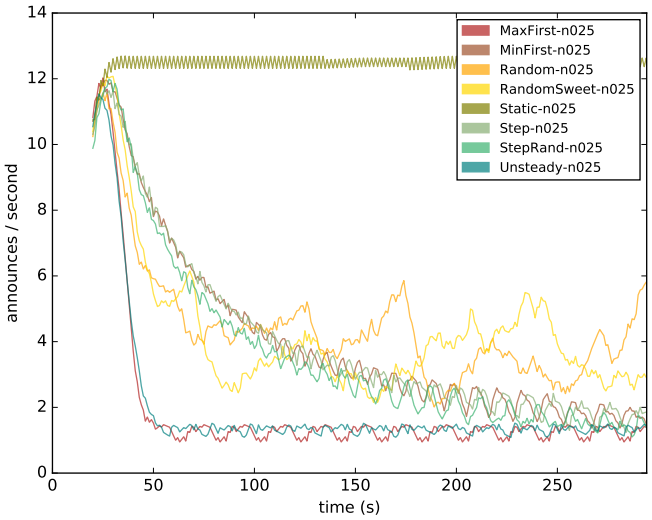


Fig. 3: Announcements per second for the proposed strategies in a 25 node static network.

Considering Figure 4a, *MaxFirst* sets the same very low announcement rates in the beginning, which leads to low global announcement rates and finally to big gaps between each two announcements. *Unsteady* (Fig. 4h) is able to compensate this problem and starts with higher announcement rates in smaller networks.

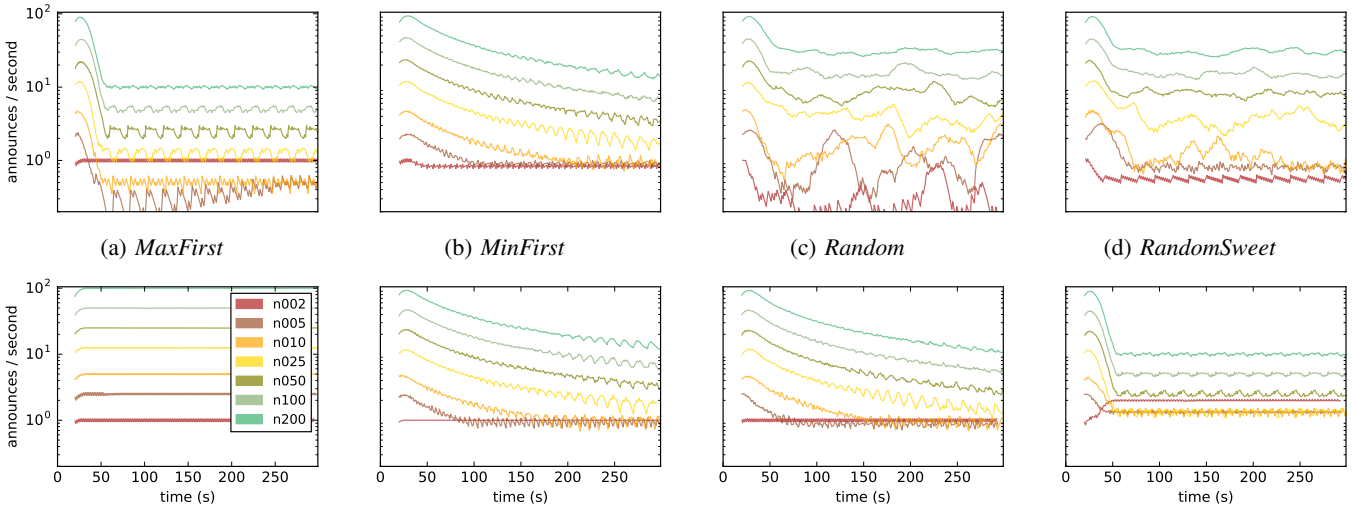
MinFirst and *Step* also behave similarly, since the down steps are implemented the same way. Both algorithms extend their announcement delay by 1 second, starting at a delay of 2 seconds. *StepAndRand* also is in the same group and only differs from *Step* by adding a random value with a maximum of 0.5 seconds. All three algorithms achieve the goal of a less saturated network and also approach the same minimum as *MaxFirst* and *Unsteady*.

In this network configuration, *RandomSweet* as well as *Random* show a similar behavior. The announcement rate drops directly after the initial observation, but stays higher than for the other strategies that achieve a low announce rate after around 200 seconds. To have similar results as, for example, *MaxFirst*, all nodes would need to pick a pretty high delay by chance, and the more nodes in the network, the more unlikely it is that all nodes do this in the same observation interval.

B. Bandwidth Savings

A major goal for using dynamic announce intervals is the reduction of bandwidth in such protocols. Table I shows the the announcement rates of the proposed strategies compared to the static announcement strategy. For this table, the announcements sent count of one node in the batch node start is used. This number also includes the observation delay in which all strategies follow the static behavior.

A major goal for using dynamic announce intervals is the reduction of bandwidth in such protocols. Table I shows the the announcement rates of the proposed strategies compared to the static announcement strategy. For this table, the total announcement count in the batch node start is used. This



(a) *MaxFirst* (b) *MinFirst* (c) *Random* (d) *RandomSweet*

(e) *Static* (f) *Step* (g) *StepRand* (h) *Unsteady*

Fig. 4: Comparison: Announcements produced by the proposed strategies in different static network configurations.

TABLE I: Announcements of the proposed strategies compared.

Name \ # Nodes	2	5	10	25	50
Static	291	732	1460	3658	7296
Random	34,4%	47,0%	37,0%	37,9%	37,3%
RandSweet	58,1%	41,7%	29,0%	35,6%	37,7%
Step	101,7%	45,4%	35,2%	33,2%	33,4%
StepRand	99,7%	42,5%	32,5%	30,1%	30,2%
MaxFirst	99,0%	21,2%	17,1%	17,0%	17,1%
MinFirst	84,9%	44,3%	34,7%	33,3%	33,5%
Unsteady	188,7%	56,8%	32,5%	17,7%	17,1%

number also includes the observation delay in which all strategies follow the static behavior.

All non-static strategies converge for growing node counts. *Step*, *StepRand* and *MinFirst* use around a third of the number of announcements compared to *Static*. *MaxFirst* and *Unsteady* take advantage of their fast adaptation rate and are able to save around 80% of the announcements. This means that only one fifth of the bandwidth is used without sacrificing any comfort or usability of the protocol.

Table I also shows that the proposed strategies benefit the most from their dynamic behavior for networks with 2 to 10 nodes. After that, only minor improvements can be achieved. The announcement rate of *Static* can be altered easily by hand and could therefore also reach the goal of a lower global announcement rate for big networks, but would then lose the ability to perform good in small networks without manual interaction on each node.

Unsteady uses more bandwidth than *Static* for a minimal network. This allows a fast discovery of new peers in an existing network and vice versa and addresses the real-world problems described in Figure 2. *Random* and *RandomSweet* have a lower total announcement count in small networks. This shows that these strategies are inferior in terms of discovery times. The remaining *Step*-based strategies show satisfactory results in small and bigger networks in terms of bandwidth usage, but take a longer time to reach an optimal resource usage.

C. Adaptation Rate

As mentioned in the previous section, *Unsteady* and *MaxFirst* have a very high adaptation rate, since they set their final announcement delay after the first observation interval for all static network configurations, as presented in Figure 4h. *MaxFirst* is able to achieve fast adaptation rates for big networks, while *MinFirst* is able to achieve this in small networks, as a result of their designs. A disadvantage of *MaxFirst* is shown in Figure 4a: For small networks, the announcement rate also drops to the minimum in the first place, so discovery may be worsened.

The adaption rate of the *Step*-based algorithms depend on the number of nodes. As outlined in Figure 4f, in a 5 nodes network around 70 seconds and in a 10 nodes network around 150 seconds are needed to fully adapt.

In Figure 5, a splitting network configuration with 10 nodes is presented. The *Step*-based strategies reach their target announcement rate immediately. In *RandomSweet* and *Unsteady*, new announcement rates are visible after about 30 seconds. Both strategies reach announcement rates as in the united, central network. This understanding only slightly differs in the merging network: The *Step* based algorithms need longer, while *Unsteady* and *MaxFirst* adapt in the observation interval.

The observed adaptation rates are also valid for the merging network configuration: *MaxFirst* and *Unsteady* adapt in a 30-seconds window, while the *Step* strategies take a longer time. For the 5 node network, the *Step* strategies also achieve an adaptation rate of around 40 seconds. Especially in small

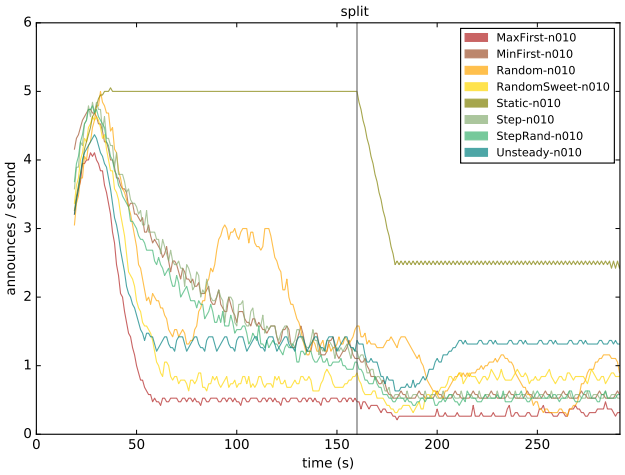


Fig. 5: Splitting network configuration with 10 nodes.

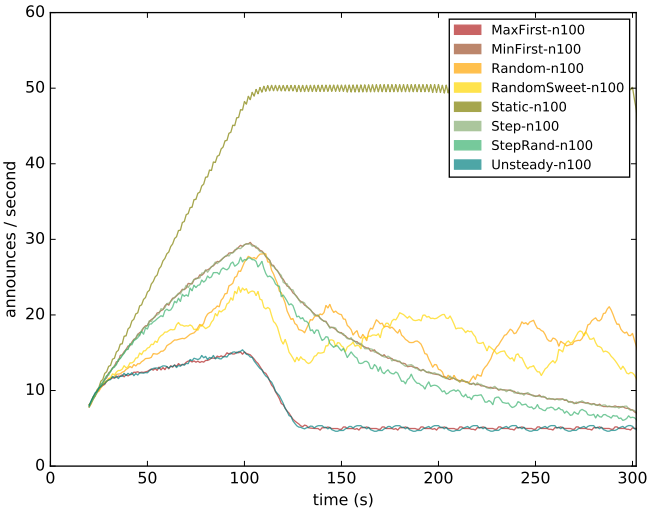


Fig. 6: Announcement computation strategies in a 100 node growing scenario.

networks, this rate is important, since the announcement gaps are compensated quickly.

Figure 6 shows a delayed start of 100 nodes, with one node starting per second. Compared to *Static*, the proposed algorithms are able to keep the announcement rates low. Since every node announces with the default interval for the first 20 seconds the announce rate grows even in the very agile *Unsteady* and *MaxFirst* Scheduler. Immediately after all nodes are spawned, the algorithms are able to adapt to the situation as described previously.

D. Announcement Gaps

Figure 7 shows a violin plot of the global announcement gaps for a 10 node static network. The mean gap correlates with the global announcement rate for obvious reasons, and so does the variance. Having this in mind, the perceptions of Subsection VI-A are backed by this plot. Although *MaxFirst* does not have the highest announcement gap, it produces

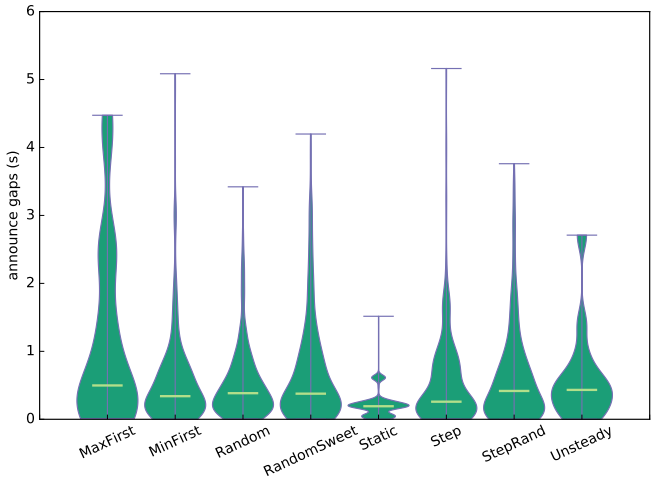


Fig. 7: *Announcement Gaps* resulting from the proposed strategies in a 10 node static network.

a relatively high percentage of longer gaps, while all other strategies only have a low number of outliers in this area. This is also the case for a 5 node network. Yet larger network configurations do not show the same characteristics. This behavior can be ascribed to the observations made in the previous section.

What stands out is that compared to *Static*, all strategies perform worse with respect to the maximum announcement gap. This can be put in perspective by examining the upper quartile: For all algorithms except for *MaxFirst*, the announcement gap upper quartiles are below 2 seconds.

E. Energy Consumption

Our initial assumption was that a reduced number of announcements would reduce the consumed energy proportionally. We evaluated this assumption in a wireless network of 9 ARM-based nodes as described in Section V-B. In these experiments, each node acted as sender and receiver simultaneously. One node (system under test - SUT) was connected to an external power meter (ODROID SmartPower), which logged the power and energy consumption of the node at a 5 Hz rate. Additionally, we ran every experiment with two different network interface configurations, with a different idle power consumption each: ad hoc mode (1.37 W) and managed mode (1.45 W).

To measure the higher end of the power consumption, two additional announcement strategies sending announcements at a high rate are introduced: *Static05* and *Static01*, with 2 and 10 announcements per second, respectively.

To compute the energy consumed by our software, the average idle power is subtracted from the measured power:

$$E(t) := \int P_{measured}(t) - P_{idle}(t)\,\mathrm{d}t. \qquad (1)$$

In the 9 node physical testbed, the default *Static* strategy uses 1.99 mWh. *Static05* and *Static01* use 11.97 mWh and 32.52 mWh for their announcements, respectively. Based on

TABLE II: Correlation of energy consumption and announcements in a 9 node physical test.

Name	# Ann.	E (mWh)	rel. Ann.	rel. E
Static	1323	1.99	1.00	1.00
Static05	5404	11.97	4.08	6,00
Static01	29342	32.52	22.18	16.31
MaxFirst	256	1.17	0.19	0.59
MinFirst	473	1.26	0.36	0.63
Random	434	1.34	0.33	0.67
RandomSweet	342	0.73	0.26	0,37
Step	495	1.20	0.37	0.60
StepRand	460	1.12	0.35	0.56
Unsteady	514	1.38	0.39	0.69

these numbers, a correlation between the number of announcements (sent and received) and the consumed energy is found and presented in Table II.

While the correlation between the number of announcements and the energy consumption is reasonable for large numbers of announcements, this correlation is not substantial with lower numbers of annoucements. The general trend seems to be correct, since all proposed strategies need less energy than *Static*. In contrast, there are examples in which this correlation seems to be vice versa, e.g., when comparing *MaxFirst* and *RandomSweet*.

To summarize, the energy measurements of our experiments show that for high numbers of announcements the energy consumption is increased. Side-effects of the programming language, as well as the relatively low energy impact of the announcements of *Mesher* disturb the energy measurements. Nevertheless, a general trend is clearly evident.

VII. Conclusion

In this paper, we have shown that without relying on application-specific properties, optimizations for network protocols relying on announcements can be achieved. We have compared eight dynamic announcement strategies, including a standard static announcement strategy and a random announcement strategy. While a random announcement strategy might preserve more bandwidth than a static annoucement strategy, it has negative side-effects compared to the other proposed announcement strategies. By dynamically changing the announcement interval and depending on the number of nodes involved, we were able to reduce the bandwidth required for announcements by more than 80% compared to a static announcement strategy. Nevertheless, our requirements of fast discovery of at least one node are still met. The evaluation of the proposed announcement strategies in terms of energy consumption show that announcements do effect battery lifetimes and are thus worth to be reduced.

There are several areas for future work. For example, so far the algorithms only have access to information like the number of announcements received in the last observation interval or the number of currently known peers. By giving the schedulers more information, further optimizations might be possible. Furthermore, a dynamic observation interval could be implemented, to allow even faster adaptation to new situations. In addition, the proposed announcement strategies should be tested in real-world applications where more computations are needed to generate the announcements, e.g., sending database state by using hashing functions, or transmitting routing tables in a mesh network. Finally, the proposed announcement strategies should be implemented in existing software platforms such as *Serval* where it could make a difference in real world emergency scenarios.

Acknowledgments

This work has been funded by the LOEWE initiative (Hessen, Germany) within the NICER project and the Deutsche Forschungsgemeinschaft (DFG, SFB 1053 - MAKI).

References

[1] S. H. Ahmed, S. H. Bouk, and D. Kim. Adaptive beaconing schemes in vanets: Hybrid approach. In *2015 International Conference on Information Networking (ICOIN)*, pages 340–345. IEEE, 2015.

[2] G. Aloi, M. Di Felice, V. Loscrì, P. Pace, and G. Ruggeri. Spontaneous smartphone networks as a user-centric solution for the future internet. *IEEE Communications Magazine*, 52(12):26–33, 2014.

[3] L. Baumgärtner, P. Gardner-Stephen, P. Graubner, J. Lakeman, J. Höchst, P. Lampe, N. Schmidt, S. Schulz, A. Sterz, and B. Freisleben. An experimental evaluation of Delay-Tolerant networking with serval. In *2016 IEEE Global Humanitarian Technology Conference (GHTC) (GHTC 2016)*, Seattle, USA, Oct. 2016.

[4] G. Dán, N. Carlsson, and I. Chatzidrossos. Efficient and highly available peer discovery: A case for independent trackers and gossiping. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 290–299. IEEE, 2011.

[5] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. *IEEE/ACM transactions on networking*, 2(2):122–136, 1994.

[6] M. B. Khalaf, A. Y. Al-Dubai, and W. Buchanan. A new adaptive broadcasting approach for mobile ad hoc networks. In *Wireless Advanced (WiAD), 2010 6th Conference on*, pages 1–6. IEEE, 2010.

[7] S. Lim, C. Yu, and C. R. Das. Randomcast: An energy-efficient communication scheme for mobile ad hoc networks. *IEEE Transactions on mobile computing*, 8(8):1039–1051, 2009.

[8] Y. Liu, D. R. Bild, D. Adrian, G. Singh, R. P. Dick, D. S. Wallach, and Z. M. Mao. Performance and energy consumption analysis of a delay-tolerant network for censorship-resistant communication. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 257–266. ACM, 2015.

[9] E. Natsheh, A. B. Jantan, S. Khatun, and S. Shamala. Adaptive optimizing of hello messages in wireless ad-hoc networks. *Int. Arab J. Inf. Technol.*, 4(3):191–200, 2007.

[10] P. S. Paul, B. C. Ghosh, K. De, S. Saha, S. Nandi, S. Saha, I. Bhattacharya, and S. Chakraborty. On design and implementation of a scalable and reliable sync system for delay tolerant challenged networks. In *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–8, Jan 2016.

[11] F. Peng. A novel adaptive mobility-aware mac protocol in wireless sensor networks. *Wireless Personal Communications*, 81(2):489–501, 2015.

[12] R. Tahar, A. Dhraief, A. Belghith, H. Mathkour, and R. Braham. Autonomous and adaptive beaconing strategy for multi-interfaced wireless mobile nodes. *Wireless Communications and Mobile Computing*, 2015.

[13] W. Wang, V. Srinivasan, and M. Motani. Adaptive contact probing mechanisms for delay tolerant applications. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 230–241. ACM, 2007.

[14] B. Zhang, Y. Li, D. Jin, P. Hui, and Z. Han. Social-aware peer discovery for d2d communications underlaying cellular networks. *IEEE Transactions on Wireless Communications*, 14(5):2426–2439, May 2015.

[15] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 35–45. ACM, 2003.