Using Buffer Space Advertisements to Avoid Congestion in Mobile Opportunistic DTNs

Jani Lakkakorpi, Mikko Pitkänen, Jörg Ott

Using Buffer Space Advertisements to Avoid Congestion in Mobile Opportunistic DTNs

Jani Lakkakorpi, Mikko Pitkänen, Jörg Ott
Email: {jani.lakkakorpi, mikko.pitkanen, jorg.ott}@aalto.fi

Aalto University, Department of Communications and Networking[†]

Abstract. This paper investigates congestion control in opportunistic networks that use delay-tolerant networking (DTN) as a basis for communication. We propose a mechanism that advertises buffer occupancy information to adjacent nodes and avoids forwarding through nodes with high buffer occupancy. The nodes then achieve global congestion avoidance simply based on locally available information. The proposed mechanism works independent of the routing protocol and is thus applicable to wide array of scenarios. Extensive simulations, with different node mobility models and radio modeling, indicate that the proposed mechanism improves message delivery ratio and decreases end-to-end delay.

Keywords: DTN, congestion control, ns-2, opportunistic network

1 Introduction

Opportunistic wireless networks dynamically created between mobile nodes are able to function without supporting infrastructure (such as base stations and the core network). While traditional Mobile Ad-hoc Networks (MANETs) use packet-based routing to establish end-to-end paths between communicating nodes, opportunistic networks allow for store-(carry-)and-forward operation based upon hop-by-hop message exchanges. Relaxing the need for an end-to-end path following the ideas of Delay-tolerant Networking (DTN) allows nodes to communicate in challenged networking environments, characterized, e.g., by sparse node population, high mobility, high delays, or unstable (short-lived) links. However, this advantage comes at a cost: Lacking an end-to-end path makes it difficult for the originator of a message to get feedback on the transmission progress of a message and thus to protect against congestion in the nodes that the messages traverse on their way towards the destination. In this paper we explore a simple local scheme for congestion control for mobile opportunistic networks. While congestion control in connected packet networks usually provides feedback to the sender to (instantaneously) throttle communication, we cannot rely on such feedback and hence seek to a) distribute and b) limit resource consumption within the network.

DTN routing protocols often attempt to protect communication against unreliable network conditions by creating multiple copies of message to deliver at least a single instance to its destination [1, 2]. Multiple copies can lead to congestion in networks that

[†] P.O. Box 13000, FI-00076 AALTO, FINLAND

are typically characterized by scarce resources. To avoid harmful effects of congestion, such as message drops and retransmissions, message copies can be removed once a copy has reached its destination [3]. Without feedback mechanisms it is difficult to adjust the number of messages copies while the message is still in its way towards the destination.

This paper introduces a simple mechanism to avoid congestion in opportunistic networks that operates proactively before congestion-induced message drops occur. The proposed mechanism locally advertises a node's buffer occupancy to adjacent nodes based upon which the latter can take local decisions and avoid sending messages to nodes whose buffers are nearly full. The proposed congestion control mechanism enables to maximize resource utilization when resources are available. When resources are scarce, however, the mechanism prevents congestion from occurring by postponing message transfers until sufficient resources are available. As our congestion control mechanism performs load reallocation inside the network, there is no need to return explicit feedback to the message source. Implicitly, any source will learn the degree of congestion inside a (region of a) network based upon the buffer state notifications it receives from its neighbors.

To evaluate the congestion control mechanisms we conduct extensive simulations. The model is novel in compared to different previous DTN routing algorithms since it models also the effect of congestion in physical radio layer, which is often omitted in DTN simulations. The simulations model the proposed usage scenarios closely and provide results with both synthetic and real life mobility traces.

This paper is organized as follows. Section 2 begins with an overview of congestion control (starting with IP networks), explains the IRTF DTN architecture and protocol, and covers related work. Section 3 introduces the proposed congestion control mechanism for mobile DTNs. Section 4 describes the models and parameters used in our simulations. Section 5 presents variety of simulation results. Finally, Section 6 concludes the paper with a summary and a brief discussion.

2 Background and Related Work

The foundations of congestion control in the Internet date back to 1988 when suitable mechanisms were devised for TCP [4]. In TCP, sender controls rate based on acknowledgments from the receiver that can be used to deduce that some packets are lost or delayed too much, and rate can be accordingly decreased at sender. To avoid synchronized buffer overflows in the network, Floyd and Jacobson [5] introduced random early detection (RED) in 1993. RED is applied to router queues and drops IP packets in a randomized fashion already before the buffers are full, thereby implicitly providing early feedback to senders. RED can lead to lower packet delays and increased TCP goodput; yet, despite being well-known and widely implemented in routers, it is not broadly deployed in the Internet. In another approach, called explicit congestion notification (ECN) [6], routers mark packets when congestion is imminent and thereby signal to the endpoints that they reduce their sending rate.

2.1 DTNs and Congestion Control

The DTN [7] architecture introduces a *bundle* protocol [8], which offers transport services for applications. The bundles are often significantly larger than IP packets in order to allow creation of self-contained messages that enable complete application interactions with a single message exchange. The architecture is designed for networks where an end-to-end path may not exist, thus the bundles are transmitted based only on hop-by-hop reliability. Any suitable convergence layer (such as TCP, UDP or LTP [9]) can be used for transferring a bundle over a single hop. Bundle spreading is limited by its lifetime, also known as time-to-live (TTL), after which the bundle expires and is removed from the network. Bundle retransmission can take place either at the originating node or at an intermediate node.

Numerous routing protocols have been proposed for DTNs. Most of them create multiple copies of the bundle in order to increase the probability of reaching the destination. *Epidemic* routing [1] creates unlimited number of messages by copying the message to all nodes that do not yet have a copy. The large number of message copies created by epidemic routing has been shown to cause congestion, which decreases message delivery probability. *Spray and wait routing* [2] explicitly limits the number of bundle copies to a fixed value. *MaxProp* uses explicit acknowledgments to remove delivered messages from intermediaries [10] .

Independent of the DTN routing protocol, the destination node may generate a return receipt to the source node when the bundle is received. The return receipts can also act as antipackets like in VACCINE antipacket mechanism [3], which deletes the bundles and gives immunity to nodes upon seeing a return receipt. Whatever the bundle routing scheme, antipackets are always sent using epidemic routing [11]. Antipackets will eventually expire, just like the bundles. Seligman *et al.* [12, 13] propose a solution to congestion for DTN custody transfer, in which a node is not allowed to drop a message to make room if it has accepted custody for this message. They suggest resolving congestion in individual nodes by moving bundles temporarily to other non-congested nodes and taking them back again later (somewhat similar to swapping in operating systems). However, their approach relies on nodes being able to reach non-congested nodes predictably (for both "swapping" messages out and back in) and cannot be applied to an opportunistic network with unpredictable contact patterns.

Burleigh *et al.* [14] implement DTN congestion control by propagating buffer utilization stress back to the bundle sources. This is accomplished by declining to take custody of bundles, forcing the source to retain the bundles and thereby increasing source node's demand for buffer space, forcing it in turn to refuse custody of bundles. A financial model of buffer space management is suggested. Zhang *et al.* [15] propose a rather similar approach.

2.2 Congestion Avoidance in Opportunistic Networks

Krifa *et al.* [16] present different bundle buffer management policies for DTNs showing that traditional buffer management policies, such as drop tail, are sub-optimal. An optimal buffer management policy, based on global knowledge about the network, is proposed using the theory of encounter-based message dissemination. The proposed

policy can be tuned either to minimize the average delay or to maximize the average delivery ratio. Moreover, they propose a distributed algorithm that uses statistical learning to approximate the global knowledge.

Radenkovic and Grundy [17] propose a congestion approach for social opportunistic networks. They suggest a combination of routing and congestion avoidance that uses heuristics to infer shorter paths to destinations from social information and use buffer information to avoid areas of the network that are congested. Pujol *et al.* [18] introduce a routing algorithm for delay tolerant networks, where messages are preferably forwarded to users that have a stronger social relation with the target of the message. The aim of the approach is to balance the message load in the network.

Ryu *et al.* [19] present a back-pressure routing and rate control algorithm for intermittently connected networks. In back-pressure routing, per-destination queues are maintained at each node. A message for destination d can be forwarded to a neighbor node only if the backlog for destination d is smaller at the neighbor node. In the modified back-pressure routing algorithm, the queue lengths are advertised to other nodes and advertised gateway node queue lengths are scaled down first. They consider applying the approach so that periodic and predictable mobile movement patterns are exploited in order to obtain throughput-optimal performance. However, this is not applicable in many opportunistic scenarios where replication-based algorithms are used to secure against unpredictable events in the network.

Thompson *et al.* [20] propose a congestion control algorithm for intermittently connected networks (later used for performance comparison in Section 5). In their scheme, message replication is dynamically limited based on local information. In order to determine congestion level in the network, the nodes exchange information about message drops and message replications upon encountering other nodes. This information is then used in calculating the replication limit. Similarly to our approach, they use local information in order to prevent congestion. However, their approach seems more prone to message drops, since the individual reception capacity of the next hop node is not known.

3 Using Buffer Space Advertisements to Avoid Congestion

With long message lifetime in DTNs, an effective congestion control mechanism should not be based on network conditions at the message creation time, but it should instead make congestion control decisions adaptively during the message lifetime in the traversed nodes. Thus, we propose a congestion control mechanism, where a message can be transferred only to such intermediate nodes who advertise sufficient available buffer capacity at the time of message forwarding. The proposed mechanism spreads buffer occupancy information in *Hello* packets that nodes periodically broadcast to each other (nodes use the same packets to discover each other). Nodes forwarding messages use the advertise buffer availability to infer how much data the the next hop node accepts for forwarding.

Our congestion control mechanism works with heterogeneous buffer and message sizes. The mechanism can be used with any DTN routing scheme and does not rely on specific routing information, support for custody transfer [14, 15], or antipackets [3].

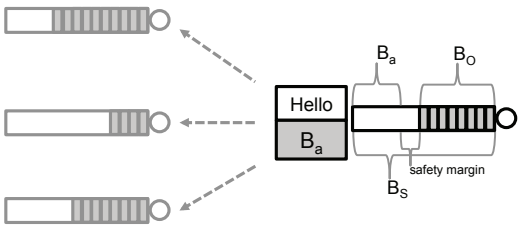

Fig. 1: Adaptive node model for congestion avoidance.

Figure 1 illustrates the variables in our model. The advertised buffer size B_a is simply calculated from total buffer size B_s by multiplying it by a congestion threshold T_C (safety margin) and subtracting the current buffer occupancy B_o as follows:

$$B_a = T_C \times B_s - B_o. \tag{1}$$

The key benefit of using the above formula to advertise the available buffer space is that each node can locally define safety margin for amount of data they want to receive. When congestion is likely to occur, e.g., the node has larger probability to experience message drop, the safety margin can be increased to accommodate larger number of unexpected message arrivals. In sparse network conditions, and when message drops have not occurred recently, the node can advertise available capacity with small safety margin. Without safety margin any concurrent transmissions from adjacent nodes could lead to message drops.

Algorithm 1 Adapt buffer space advertisement B_a

if $D = 0$ AND $T_C < T_{Cmax}$ **then**
  set $T_C \Leftarrow T_C + ai$
else
  if $D > 0$ AND $T_C > T_{Cmin}$ **then**
    set $T_C \Leftarrow T_C \times md$
  end if
  set $D \Leftarrow 0$
end if
set $B_a \Leftarrow T_C \times B_s - B_o$

To adapt the advertised value to varying network conditions, we propose to adjust the advertised buffer capacity based on message drops D according to Algorithm 1. It uses the additive increase and multiplicative decrease (AIMD) approach to adapt the congestion threshold T_C based on message drops. If there are message drops between two consecutive Hello messages (interval of 100 ms), the congestion threshold is reduced. Otherwise, the congestion threshold is increased. This leads to an adaptive approach that backs off exponentially when congestion occurs, and conservatively

increases when more capacity appears available. We apply the following default parameters that we determined experimentally: $ai = 0.01$ and $md = 0.8$. Variable T_C is initialized with a value of 0.8. The minimum and maximum values for T_C are 0.5 and 0.9, correspondingly. After having calculated B_a, node appends the value to Hello messages that it sends to its immediate neighbors as shown in Figure 1.

Algorithm 2 Forward message to intermediate node

```
select message from message storage
if routing protocol allows then
   get B_a(i)
   if B_a(i) > M_s then
      forward message
      set B_a(i) ⇐ B_a(i) − M_S
   end if
end if
```

Upon sending messages to another intermediate node for forwarding, each node executes Algorithm 2. If the advertised buffer space B_a at the neighbor node (or message creator node) is less than message size (M_s), the message cannot be forwarded to this particular neighbor (or created). This approach maintains two properties that are typical to DTN routing protocols. First, if a node receives a message that is destined for it, full buffers do not matter. Second, the message is not sent to a node that already holds a copy of it.

The congestion control mechanism is executed immediately after node encounter occurs, i.e., prior to forwarding to next hop node. Thus, the approach is interoperable with any mechanism used by the known DTN routing protocol making it applicable to wide variety networks. For example, with spray and wait, when the node runs out of message tokens, the approach allows forwarding the message only to final destination.

Our congestion control algorithm requires only minimal state because it uses only information about the neighbors per current contact. The approach is somewhat similar to back-pressure mechanisms, since information about full buffers is propagated, albeit not explicitly beyond a single hop into the direction of the message originator. The mechanism operates clearly using only a local scope, with the information it requires and the interactions it performs restricted to nodes in its local proximity. However, in Section 5 we show that it is effective across the network.

4 Simulation Model

Our simulations use several extensions to the *ns2* simulator that allow us to study DTN routing together with accurate models for radio links. Our simulations build on top of synthetically generated random waypoint (RWP) model, more realistic yet synthetic pedestrian mobility model, and real-world vehicular traces.

4.1 Traffic and Mobility Models

We apply a relatively simple traffic model in which each node sends a (fixed or variable size) message at a random time with 200 second intervals $[t, t + 200s]$ to another, randomly selected, node. Before sending to MAC layer, the bundles are fragmented into 1500-byte datagrams. A retransmission mechanism, providing reliable delivery of datagrams is implemented, so that messages will not be lost due to transmission errors.

Random waypoint is our first mobility model since this model is well understood and it is easy to generate scenarios with different network densities and node velocities. We generate random waypoint node mobility using the setdest program, which is part of the ns-allinone package [21]. We have 40 mobile nodes that select a random direction and a random (uniformly distributed) speed at random times. Maximum speed is 20 m/s and pause length is two seconds. We always pause before choosing a new direction and a new speed. In our RWP scenarios, area size ranges from 10 m times 10 m to 2000 m times 2000 m, and the simulation time is always 5000 seconds.

In both of our trace-based mobility models the number of mobile nodes is 116 and the simulation time is 3600 seconds. In the first mobility trace, the San Francisco taxi cab trace [22], the area size is 5700 m times 6600 m. The whole data set contains GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay area. We choose to use this trace due to its high resolution, node positions are recorded frequently enough to provide location information for the used radio models.

The second mobility trace was obtained in a synthetic fashion: the map of Helsinki city center (area dimensions: 4500 m times 3400 m) is used as input for the ONE simulator [23] and the nodes are configured to move between selected points of interest. Node velocity is uniformly distributed between 0.7 m/s and 1.4 m/s and pause length between 0 and 120 s. This model provides a more dense network scenario in comparison to the aforementioned taxi cab scenario, which is relatively sparse.

4.2 DTN Model

In the simulator, DTN nodes advertise their buffer content to each other every 100 ms by sending Hello messages. In our simulations, this message has enough room for the identifiers of buffered bundles and return receipts. A bundle can be generated only if the node has sufficient buffer space available. When the bundle lifetime expires, all copies of that bundle are deleted. If the sender does not receive a return receipt within retransmission timeout, it will retransmit the bundle. Return receipts may also serve as antipackets; their lifetime is the minimum of retransmission timeout (1000 seconds) less bundle forwarding time and bundle lifetime (750 seconds). Antipackets and Hello messages are small in size. The selected routing protocol is either epidemic routing or binary spray and wait, the latter if uses 16 message copies.

Return receipts are forwarded first. When the head-of-line receipt has been forwarded to all current neighbors, one by one, we put that receipt to the tail of the receipt queue and dequeue the next receipt. Then, we forward regular bundles to their destinations, in a similar manner as return receipts. After this, the bundles are re-ordered so that the least forwarded bundles are put to the head of the queue. Finally, regular bundles are forwarded to neighboring, non-destination nodes.

4.3 Wireless Channel Model

We use a realistic wireless channel model from the dei80211mr library [24], which is now a part of the *ns-allinone* package. There is support for different transmission rates, modulation methods and coding schemes that are defined in the IEEE802.11b/g standards.

A signal-to-interference-and-noise ratio (SINR)[1] based packet level error model is introduced that is calculated using pre-determined curves of packet error rate (PER) vs. SINR and packet size. The reception threshold (RXThresh_) variable, which has been used in the default *ns2* 802.11 implementation, has been removed. SINR, in turn, is calculated using received signal strength, noise, and interference. Interference is calculated using a Gaussian model to account for all transmissions that happen simultaneously to the one which is considered for reception. Noise strength is fixed in all simulations. The IEEE 802.11g simulation parameters are the same as in [25].

The capture model, that is, the determination of whether a packet can be received when there are other concurrent transmissions, is embedded in the interference model.

5 Simulation Results

5.1 Random Waypoint Mobility

Figure 2 compares routing performance in the random waypoint scenario, showing plain epidemic, spray and wait, and epidemic routing enhanced with congestion control (CC). For each protocol, performance figures are also provided with the antipacket (AP) mechanism. Node buffer sizes are randomly chosen to be 250 KB or 2 MB so that both occur with equal probability. The size of the individual bundles is uniformly distributed between 1 Byte and 20 KB, and the (static) congestion threshold (T_C) is 0.8.

An initial (expected) observation is that reducing congestion by means of antipackets improves message delivery probability with all routing protocols. A significant improvement is also gained by applying congestion control or explicitly limiting the number of copies per message. Combining both we find that antipackets dominate the performance gain, irrespective of congestion control. But for sparse scenarios (1000 m and above), congestion controlled epidemic routing does as well as antipackets.

The message delivery delay plot shows that the antipacket mechanism is very efficient in reducing the message delivery delay. This is expected behavior since removing copies of delivered messages from buffers prevents them from blocking the undelivered ones. Explicitly limiting number of message copies (spray and wait) yields the smallest delivery delays, since the other approaches do not minimize queuing in intermediaries, but rather only avoid increasing it too much. With antipackets, congestion control adds a marginal improvement and only for sparse scenarios. Without antipackets there is notable gain but the main advantage is clearly on the delivery ratio—which is expected because our congestion control algorithm delays forwarding.

[1] Noise is set according to $Pn = kTB$, where k = 1.38e-23 J/K, T = 290 K, and B =2.437 GHz. With the selected parameters, the node transmission range will be 66-130 m, depending on the modulation and coding scheme (MCS). The better the MCS, the higher the transmission rate (and smaller the transmission range).

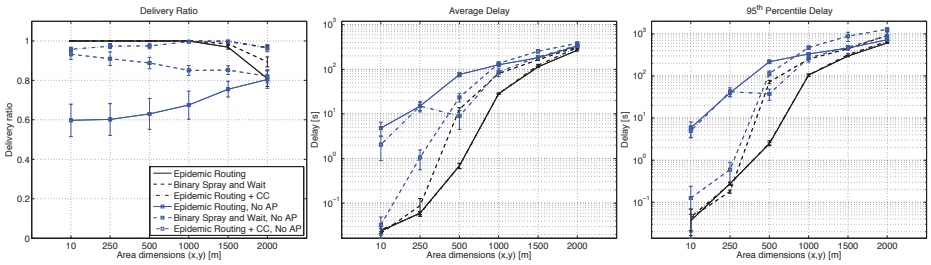

Fig. 2: Heterogeneous nodes, RWP mobility.

Further simulations (graphs omitted) with homogeneous buffer size (400 KB) and uniform bundle size (10 KB) show largely similar results.

5.2 Trace-based Mobility

Figure 3 illustrates the performance evaluation with more realistic conditions. The top row shows performance with vehicular mobility derived from the San Francisco cab trace and the bottom row with pedestrian mobility created using the Helsinki City Scenario. In both cases, the message size is uniformly distributed and varies between 10 KB and 100 KB with 10 KB granularity. In the San Francisco case, the node buffer size is either 1.375 MB or 11 MB (both occur with 50% probability) whereas in the Helsinki case the node buffer size is always 6 MB.

The results of the two simulation scenarios are quite similar. With both mobility models, using congestion control leads to better delivery ratios. However, in the (more dense) Helsinki city scenario, this gain is marginal and it seems that bandwidth is a bottleneck, too, so that spray and wait outperforms epidemic routing. Larger hop counts indicate that using congestion control allows (partly forces) messages to travel further through the network. This is a positive observation, since using congestion control prevents the message replication (also towards destination) in the intermediary nodes under heavy traffic.

The delay plot indicates largely similar behavior across all routing approaches. With congestion control, on the one hand, queues are kept a shorter by means of the threshold value, as also indicated by the maximum queue sizes. On the other hand, messages may take longer routes (leading to larger delay) because well-connected (“central”) nodes, which would enable shorter paths, may have already full buffers and thus cannot be chosen as next hops. A reduction of the (maximum) delivery delay is achievable in some scenarios (as visible for the taxi trace), but not in others. In the RWP scenario above and in the HCS scenario, node mobility and frequent but shorter contacts lead to higher buffer fill ratio and thus more queuing. Overall, the performance gain for delay is scenario-dependent and improved delivery rate may or may not come with reduced delay.

The right side of figure 3 shows the buffer occupancy of the nodes. To some extent, this reflects the operation of the congestion control algorithm. For the sparser taxi sce-

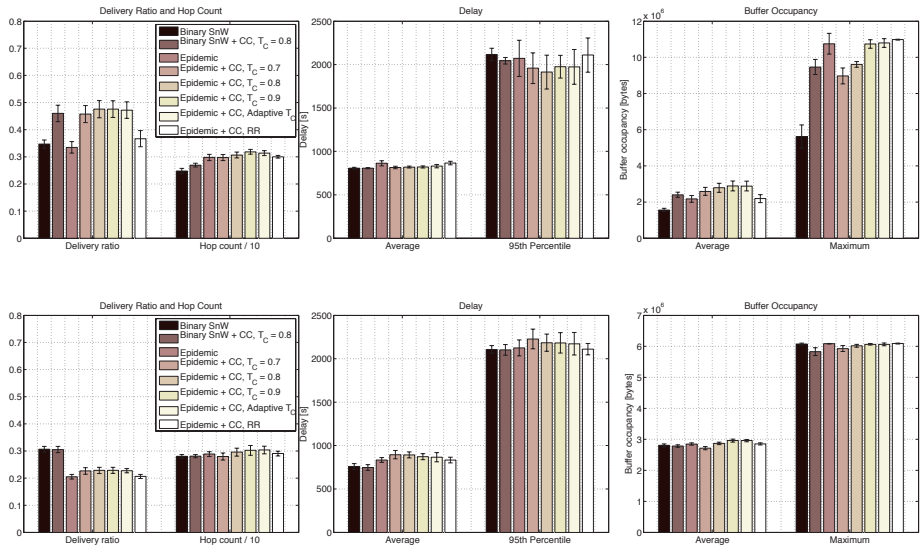

Fig. 3: Real (top) and synthetic mobility trace (bottom).

nario, the maximum and mean buffer levels are correlated with the respectively chosen threshold (see below) whereas, for the denser HCS scenario, the fill levels are roughly equal and above the threshold. We attribute this to the dense nature: frequent contacts and parallel transfers lead to the congestion control algorithm making active use of its safety margin. Overall, this shows that, despite explicitly limiting message replication, our congestion control is able to make use of forwarding capacity in the network.

Figure 3 also illustrates a performance comparison of different congestion control schemes. Congestion threshold (T_C) is either static (0.7, 0.8, 0.9) or adaptive. We can see that a congestion threshold of 0.9 leads to best message delivery ratio. However, we believe that static congestion threshold may not be suitable for all traffic and mobility scenarios and, therefore, we prefer an adaptive congestion threshold. The first adaptive case is our own algorithm (adaptive increase, multiplicative decrease of T_C) while the second one (RR) is from Thompson *et al.* [20]. As described earlier, our minimum and maximum values for T_C are 0.5 and 0.9.

The results show that our adaptive algorithm (congestion threshold fluctuates between 0.5 and 0.9) gives as good results as the best static case. However, the algorithm from Thompson *et al.* does not seem to work as expected. This may be the result of different mobility and wireless models compared to [20]. We implemented the latter algorithm exactly as described in [20] with the exception that congestion value CV was updated periodically (every ten seconds) and not every time a given node meets another node. The latter alternative would (according to [20]) lead to very low delivery ratios; the time between two contacts could be either very short or very long and thus there could be no or tens of received/dropped messages between two contacts.

Summarizing we find that the proposed congestion control variants improve delivery performance with both mobility scenarios while not negatively impacting any of the other key performance metrics. This suggests that even a very simple congestion control approach with only minimal knowledge can be effective in opportunistic networks.

6 Conclusion

We have presented a simple congestion control mechanism for mobile opportunistic networks that operates using only instantly available local information from itself and its current contacts. The algorithm is independent of the chosen routing protocol. Our simulation results using the random waypoint model as well as real-life and synthetic mobility traces show that our simple congestion control enhancement (implemented for epidemic routing and spray and wait) performs well across our diverse scenarios: applying this congestion control scheme generally leads to higher message ratios and may additionally yield lower delivery delays. We did not come across cases in which our algorithm harms performance. Exploring additional mobility scenarios and studying the interaction with other routing protocols (especially such using utility functions) are subject to our ongoing work.

In contrast to well-connected networks, our algorithm focuses on distributing the load inside the network and provides only indirect backpressure to the sender: a sender refuses to create new messages if there is not enough buffer space locally available; this allows the congestion control algorithm to pace message generation. This decision is taken as a function of the buffer occupancy at the sender (and thus implicitly the immediate surroundings) rather than of the path towards the destination; this assumes that transitive propagation of congestion information about a region occurs. Moreover, the feedback may—naturally—arrive with quite some delay and relies on sufficient interactions between the nodes. One interesting avenue of future research is understanding how congestion signals spread in time and space, how congestion regions can be identified, and how such knowledge can be exploited for refining congestion control mechanisms.

Finally, the way how feedback is provided causes all applications on a node (in fact all nodes in a region) being treated equally: if some of them generate a disproportionally high load all others might suffer, too, so that additional mechanisms for segregating traffic inside the buffers (e.g., as discussed in [26]) may need to be applied.

Acknowledgments

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

References

1. A. Vahdat and D. Becker, “Epidemic routing for partially connected ad hoc networks,” tech. rep., Duke University, April 2000.

2. T. Spyropoulos, K. Psounis, and C. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *ACM SIGCOMM Workshop on Delay-Tolerant Networking*, (Philadelphia, PA, USA), pp. 252–259, August 2005.
3. Z. Haas and T. Small, "A new networking model for biological applications of ad hoc sensor networks," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 27–40, February 2006.
4. V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, (Stanford, CA, USA), pp. 314–329, August 1988.
5. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.
6. K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC Standards Track 3168, IETF, September 2001.
7. "Delay Tolerant Networking Research Group." `http://www.dtnrg.org`.
8. K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC Experimental 5050, IETF, November 2007.
9. M. Ramadas, S. Burleigh, and S. Farrell, "Licklider Transmission Protocol - Specification," RFC Experimental 5326, IETF, September 2008.
10. J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *IEEE INFOCOM*, (Barcelona, Spain), pp. 1–11, April 2006.
11. B. Walker, J. Glenn, and T. Clancy, "Analysis of simple counting protocols for delay-tolerant networks," in *CHANTS'07*, (Montréal, Québec, Canada), pp. 19–26, September 2007.
12. M. Seligman, K. Fall, and P. Mundur, "Alternative custodians for congestion control in delay tolerant networks," in *CHANTS'06*, (Pisa, Italy), pp. 229–236, September 2006.
13. M. Seligman, K. Fall, and P. Mundur, "Storage routing for DTN congestion control," *Wirel. Commun. Mob. Comput.*, vol. 7, no. 10, pp. 1183–1196, 2007.
14. S. Burleigh, E. Jennings, and J. Schoolcraft, "Autonomous congestion control in delay-tolerant networks," in *SpaceOps*, 2006.
15. G. Zhang, J. Wang, and Y. Liu, "Congestion management in delay tolerant networks," in *WICON'08*, (Maui, Hawaii, USA), pp. 1–9, November 2008.
16. A. Krifa, C. Barakat, and T. Spyropoulos, "Optimal buffer management policies for delay tolerant networks," in *SECON*, (San Francisco, CA, USA), pp. 260–268, July 2008.
17. M. Randkovic and A. Grundy, "Congestion aware forwarding in delay tolerant and social opportunistic networks," in *WONS*, (Bardonecchia, Italy), pp. 60–67, January 2011.
18. J. Pujol, A. Toledo, and P. Rodriguez, "Fair routing in delay tolerant networks," in *IEEE INFOCOM*, (Rio de Janeiro, Brazil), pp. 837–845, April 2009.
19. J. Ryu, V. Bhargava, N. Paine, and S. Shakkottai, "Back-pressure routing and rate control for icns," in *ACM MobiCom*, (Chicago, Illinois, USA), pp. 365–376, September 2010.
20. N. Thompson, S. Nelson, M. Bakht, T. Abdelzaher, and R. Kravets, "Retiring replicants: congestion control for intermittently-connected networks," in *IEEE INFOCOM*, (San Diego, California, USA), pp. 1118–1126, March 2010.
21. UCB/LBNL/VINT, "Network Simulator - ns (version 2)." `http://www.isi.edu/nsnam/ns`.
22. "Cabspotting." `http://cabspotting.org`.
23. A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09*, (Rome, Italy), March 2009.
24. U. of Padova, "dei80211mr: a new 802.11 implementation for NS-2." `http://www.dei.unipd.it/wdyn/?IDsezione=5090`.
25. J. Lakkakorpi, M. Pitkänen, and J. Ott, "Adaptive routing in mobile opportunistic networks," in *ACM MSWIM'10*, (Bodrum, Turkey), pp. 101–109, October 2010.
26. J. Solis, N. Asokan, K. Kostiainen, P. Ginzboorg, and J. Ott, "Controlling resource hogs in mobile delay-tolerant networks," *Comput. Commun.*, vol. 33, no. 1, pp. 2–10, 2010.