CASPaR: Congestion Avoidance Shortest Path Routing for Delay Tolerant Networks

Michael F. Stewart, Rajgopal Kannan†
Department of Computer Science
Louisiana State University, USA

Amit Dvir
Cyber Technology Innovation Center
Department of Computer Science
Ariel University, Israel

Bhaskar Krishnamachari
Ming Hsieh Dept. of EE
University of Southern California, USA

***Abstract*—Unlike traditional TCP/IP-based networks, Delay and Disruption Tolerant Networks (DTNs) may experience connectivity disruptions and guarantee no end-to-end connectivity between source and destination. As the popularity of DTNs continues to rise, so does the need for a robust and low latency routing protocol. This paper describes a novel DTN routing algorithm referred to as Congestion Avoidance Shortest Path Routing (CASPaR), which seeks to maximize packet delivery probability while minimizing latency. CASPaR attempts this without any direct knowledge of node connectivity outside of its own neighborhood. Our simulation results show that CASPaR outperforms well-known protocols in terms of packet delivery probability.**

***Index Terms*—DTN, Routing, Congestion Control**

I. Introduction

Delay Tolerant Networks (DTNs) attempt to facilitate communication where connected paths do not always exist. Attempts to use conventional Mobile Ad-Hoc routing (MANET) protocols such as reactive [1], proactive [2], and hybrid [3] approaches have resulted in failure. Successful DTN protocols must adopt a single or multi-copy "store and forward" approach. This is an important but challenging problem [4] especially considering that DTN devices are increasingly being integrated into our everyday lives.

The development of a one-copy DTN protocol that addresses congestion avoidance, shortest path routing and is capable of operating efficiently in a high-load network is the motivation behind the Congestion Avoidance Shortest Path Routing protocol (*CASPaR*). The algorithm is defined by developmental guidelines: 1) no packet duplication, 2) route packets *to* or *closer to* their destinations and hold onto packets when appropriate and 3) integrate congestion avoidance into the design. *CASPaR*'s goals are: 1) *learn* direct routes to destinations, 2) avoid congestion, 3) dynamically correct routes as the topology changes and 4) minimize latency by moving packets over newly discovered routes. *CASPaR* must negotiate node queue differentials between neighbors similar to backpressure and map shortest paths without explicitly discovering them. Here we present preliminary results showing that *CASPaR* accomplishes these goals.

†This work was performed under the support of NSF grant # CNS-1018273 and LEQSF-EPS(2014)-PFUND-352 while the author was on sabbatical at the Ming Hsieh Dept. of EE, University of Southern California.

The rest of the paper is organized as follows: Section II defines the *CASPaR* protocol and describes its algorithm and parameters. Section III describes the simulation environment and parameters and lists each of the protocols simulated. Section IV presents the experimental results. Section V provides some concluding remarks and a look into future work.

II. CASPaR Protocol

CASPaR is a one-copy routing protocol consisting of two interdependent mechanisms: 1) direct routing and 2) congestion avoidance. The algorithm routes packets over connected paths and employs a routing-protocol-dependent, proactive congestion-avoidance mechanism [5] that uses an open loop congestion control scheme based on buffer availability and historical connectivity knowledge. This allows for alternate route discovery to avoid congestion buildup. Ultimately, congestion avoidance takes precedence over routing forcing a direct-delivery-like mode of operation during heavy traffic. Except for their 1-hop neighbors, nodes have no knowledge of the network.

A. Principle of Operation

All nodes maintain an estimate, $C_n^c(t)$, of the cost to deliver packets to each destination node c. This cost attempts to track the least congested and shortest paths to the destination based on historical knowledge of connectivity to node c and the waiting time of packets to c in node n's queue. All nodes in the network broadcast a Request For Costs (RFC) to its neighbors when one of three things occurs: 1) a packet has just been received from a neighbor, 2) a packet has just been created or 3) the RFC periodic timer expires. Neighboring nodes, upon receiving the RFC, respond with their destination cost table which contains a list of all destinations and the cost to send a packet to that destination. If node n's estimate of delivery costs to c is the lowest amongst its neighbors, then n holds onto these packets in its buffer until it either meets a neighbor with a better (lower) estimate or is connected to c (we use a preference factor of 0.9 to give a slight preference to node n holding onto these packets).

B. Model

Path congestion and route connectivity are modeled by minimizing the delivery costs along some path from source to

destination and is characterized by two convoluted parameters: The first is *Proximity Measure*:

$$\Theta_n^c(t) = \frac{Q_n^c(t)}{T_n^c(t)} \tag{1}$$

$\Theta_n^c(t)$ is a value between 0 and 1 where 1 indicates n and c are connected and 0 means they were never connected. $T_{n,t}^c$ is incremented at every time step and $Q_{n,t}^c$ is set equal to $T_{n,t}^c$ while c and n remain connected thus forcing $\Theta_n^c(t)$ equal to 1. Once disconnected, $\Theta_n^c(t)$ begins decreasing linearly each time step. Periodically both Q and T are reset to some initial values that represent a default measure of connectivity. The second parameter is the *Net Destination Queue Waiting Time*:

$$W_n^c(t) = \sum_{i=0}^{N} (\mathcal{T} - a_{n,i}^c) \tag{2}$$

where $\mathcal{T}$ is the current time and $a_{n,i}^c$ is the arrival time of packet i at n destined for c. The queue waiting times of packets are used as a proxy for congestion as opposed to backpressure which uses queue size differentials. Hence, we model delivery costs as an exponentially increasing function of net waiting times of packets with an increasing discount factor based on connectivity probability. The estimated delivery costs to c via n are calculated as:

$$C_n^c(t) = W_n^c(t)\dot{(}1 - \Theta_n^c(t)) + C_n^c(t-1) \tag{3}$$

CASPaR's delivery costs are calculated and transmission costs between 1-hop neighbors set to 0. Setting 1-hop transmission costs to 0 has the effects: a) If a connection from source n to destination c exists then the delivery cost will be 0 everywhere along that path regardless of path length sinking packets directly to destination c (see line 24 from Alg. 1) and b) If a connection from source n to destination c does not exist then packets to c will be spread over the network based on congestion, radiating outwards towards the destination. When one of these nodes becomes connected, a direct path to c is created and packets are quickly routed to their destinations.

In addition to $\Theta_n^c(t)$ being set to 1, the historical cost, $C_n^c(t-1)$, is reset to 0 when n and c become connected at time t. From the definition of W earlier, the marginal increase in net waiting times at each time step are a function of queue size to c. Thus as can be seen from the expression above, lightly congested nodes along short paths to the destination are favored (the more recently a node is in contact with the destination and the smaller its queue size, the lower the transmission cost) and therefore the net effect of the algorithm is to reinforce delivery on short, less congested paths.

Proximity Measure and *Net Destination Queue Waiting Time*, parameterize not only the shortest but least congested paths. The *Proximity Measure* attempts to minimize the path length while *Net Destination Queue Waiting Time* pulls packets towards neighbors with the smallest queues (similar to a backpressure mechanism [6]) minimizing routing across congested paths. This technique develops routes that *chase* the destination.

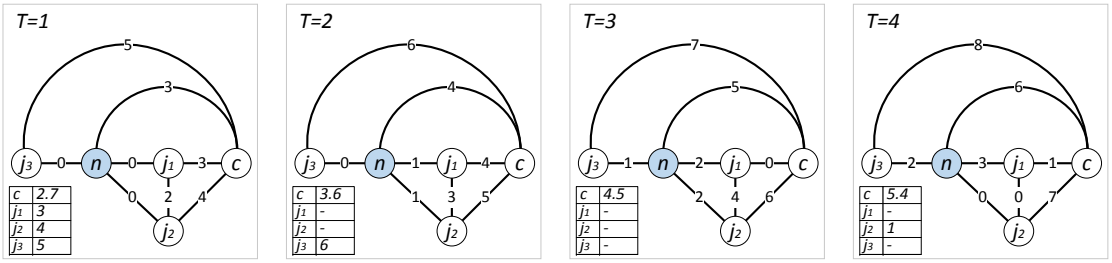
Packets are transmitted in a lowest-cost first, longest-queue-waiting-time second, priority order; the *cheapest*, oldest packets are transmitted first. A minimum node loop count (MLC) is integrated into the algorithm so packets avoid repeatedly traversing the same nodes. It is the minimum number of consecutive unique nodes that must exist in a path before a packet is allowed to revisit a node.

Algorithm 1 The CASPaR Algorithm

1: **function** UPDATE RANGE STATUS
2: **for all** destinations **do**
3: **if** destination c is within 1-hop of node n **then**
4: $R_{n,t}^c$ = true
5: **else**
6: $R_{n,t}^c$ = false
7: **function** UPDATE MEASURE OF PROXIMITY
8: **for all** destinations **do**
9: $T_n^c(t) = T_n^c(t) + 1$
10: **if** $R_{n,t}^c$ **then** $Q_n^c(t) = T_n^c(t)$
11: $\Theta_n^c(t) = \frac{Q_n^c(t)}{T_n^c(t)}$
12: **function** UPDATE QUEUE WAITING TIME
13: **for all** destinations **do**
14: $W_n^c(t) = \sum_{i=0}^{N} (\mathcal{T} - a_{n,i}^c)$
15: **function** UPDATE DELIVERY COST
16: **for all** destinations **do**
17: **if** $R_{n,t}^c$ **then** $C_n^c(t-1) = 0$
18: $C_n^c(t) = W_n^c(t)\dot{(}1 - \Theta_n^c(t)) + C_n^c(t-1)$
19: **function** REQUEST FOR COSTS
20: Update Range Status ()
21: Update Measure of Proximity ()
22: Update Queue Waiting Time ()
23: Update Delivery Cost ()
24: $C_n^c(t) = \frac{9}{10}\dot{C}_n^c(t)$ ▷ Calculate Self-Delivery Estimate
25: **for all** nodes j in 1-hop range of node n, for all destinations c **do**
26: Select $C_m^c = \min(C_j^c(t), C_n^c(t))$ and relay r accordingly
27: Update $C_n^c(t) = C_m^c$
28: **if** $r = n$ **then** Relay packet to node j

C. Example

The weighted graph in Figure 1 represents a small network whose vertices are nodes $n, j_1, j_2, j_3,$, c and weighted edges are the transmission costs between nodes (a weighted-edge of 0 represents neighboring nodes). In this scenario, node n is to deliver a packet to c (at the bottom left-hand corner of each panel is the *destination cost table*). At $T = 1$, node n broadcasts a RFC. Nodes j_1, j_2 and j_3 respond with their *destination cost tables* and n compares them against its self delivery cost, shown in the destination cost table to be $c = 2.7$, $j_1 = 3$, $j_2 = 4$ and $j_3 = 5$. Node n is unaware of the state of the network beyond its own neighborhood. After the RFC responses are received, node n learns its neighbors: j_1, j_2 and j_3 and that they can deliver a packet to c for 3, 4 and 5 respectively. Node n deduces that a direct route to node c doesn't exist since a 0-cost wasn't received. Node n selects itself as the relay since its delivery cost is the minimum. At $T = 4$, node n and j_2 are neighbors. Node j_2 responds with the minimum cost of 1. Node n transmits the packet to j_2 who transmits the packet to j_1; provided that j_1 and j_2 are still neighbors. Node j_1 would hold onto the packet if the topology were identical at the time j_1 received it. However,



(a) Time T=1 (b) Time T=2 (c) Time T=3 (d) Time T=4

Fig. 1. A *CASPaR* example iterating through 4 time periods and the transactions between a group of nodes in a small network.

the network changes quickly. The receipt of the packet from n causes j_2 to initiate its own RFC and other nodes potentially in the path might update their *destination cost tables* forcing more route changes.

III. Simulation

The ONE simulator [7] is used to compare 8 DTN protocols: *Direct Delivery (DD)* [8], *Epidemic (EPI)* [9], *Prophet with Estimation (PRO)* [10], *MaxProp (MP)* [11], *Backpressure LaB (LaB)*, *Spray and Wait (SaW)* [12], *Shortest Path (SP)*, *CASPaR (CASPaR)*. *Shortest Path* is based upon Dijkstra's algorithm where the shortest routes are calculated with each update. It represents the upper-bound on the delivery performance and is the standard by which all protocols are measured. The simulation settings are: World size = 1km x 1km; Node count = 100; Run time = 3,600 seconds; Transmit speed = 10 Mbps; Transmit range = 100 meters; Buffer-sizes tested = 0.2, 0.5, 1, 3, 5, 10 and 30 MB; Node speed = 0.5 - 1.5 meters per second; Message TTL = 5 hours; Message period = 1 second; Message size = 100 KB; Node movement = Random walk; Map = Open map; Minimum Loop Count = 5; $3,600$ messages are created during the 1 hour simulation; Source and destination nodes are chosen randomly therefore each node is just as likely as any other to source or sink messages; The message time-to-live (TTL) is 300 minutes, explicitly set to be greater then the total simulation time so that TTL doesn't play a role in dropped messages; Messages are only dropped due to queue overflow or protocol-based metrics.

IV. Simulation Results

Here we present results from Random Scenario simulations focusing on performance metrics: Delivery Probability, Overhead Ratio, Hop Count and Latency. Also reported are results from two additional investigations: 1) latency frequency and 2) route distribution. All performance metric plots, for all protocols except *MaxProp*, show 1-sigma uncertainties representing deviation between the 17 simulation runs. *MaxProp*'s simulation times prohibited multiple runs.

A. Delivery Probability

Figure 2 shows as buffer size increases so does delivery rate until a bounded maxima is reached. The maximum delivery rate for all protocols except *MaxProp* is reached at < 10 MB buffer allowing for a good comparison between tested protocols. Results show four distinct protocol behaviors: 1) the *SP* group includes both *CASPaR* and *Shortest Path* routing protocols and is characterized by its steep rise in delivery probability settling close to or above 80 percent; 2) the *Direct* group includes *PRoPHET*, *Direct Delivery* and *LaB*. This group also has a relatively steep rise in delivery probability but settles at < 50 percent; 3) *Spray and Wait* is in a group by itself and can be identified by its slow rise, reaching a maximum at > 10 MB buffer; 4) *MaxProp*, also in a group by itself, is unique. Its delivery probability maintains a shallow but constant rise reaching a maximum of 90 percent at a 30 MB buffer and still increasing. *Shortest Path* sets upper-

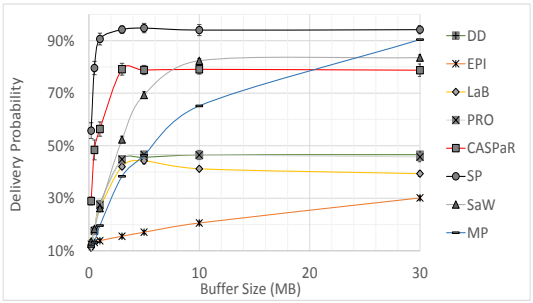


Fig. 2. The delivery probability as a function of queue size. Most $1-sigma$ error bars can not be seen indicating small deviations between simulations.

bound on delivery rate (95 percent). *Direct Delivery* sets the *effective* lower bound at 45 percent. It is considered the lower bound due to its simplistic routing scheme; all protocols should perform as well in this scenario. This result reveals that any two nodes come in contact with each other 45 percent of the time. *CASPaR* delivers > 55 percent of its packets with a 1 MB buffer; twice the number of packets delivered by the next best algorithm. This reveals that *CASPaR* delivers packets more quickly or they are being more evenly distributed across the network or both. Alternatively, *Spray and Wait* performs poorly until its queue size reaches > 10 MB and then barely outperforms *CASPaR* while *MaxProp* starts poorly but outperforms all but *Shortest Path* at a > 25 MB buffer.

B. Latency

Average latency, defined by ONE as the average time for all delivered packets to travel from source to destination, may be the most meaningful metric. It provides both packet

delivery rate and an indirect performance metric for delivery probability. However, average latency can be falsely biased since only packets that reach their destinations contribute to the reported average and it is these undelivered packets that, if delivered, would raise the average latency. The buffer size must be large enough so that packet drop is not a factor. Figure 3 shows this to be at > 10 MB for all protocols except *MaxProp* and *Epidemic*. Regardless, median latency, as shown in Figure 4, provides a better approximation of true latency since the average can be biased by extremely large latencies. To

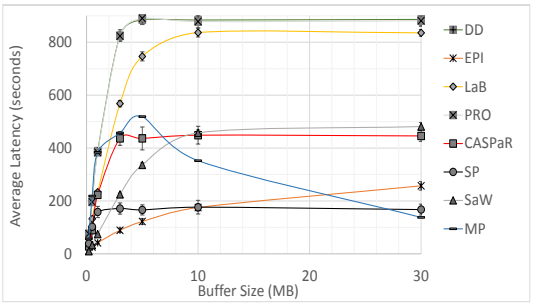


Fig. 3. The average end-to-end packet latency as a function of queue size.

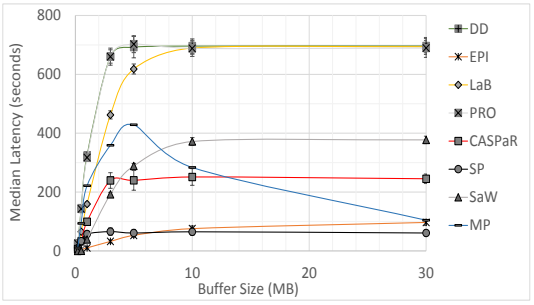


Fig. 4. The median latency as function of queue size.

illustrate, notice the *CASPaR* median latency is nearly half its average and it is nearly a third for *Shortest Path*, an indication that there are low-latency measurements skewing the average. The median and average latencies of *MaxProp*, *Spray and Wait* and the protocols in the *Direct* group are similar in value indicating more balanced measurements. Figure 4 shows that *CASPaR* performs quite well with a median latency of about 250 seconds. *MaxProp* exhibits unique behaviour as it rises above 400 seconds at a 5 MB buffer but drops to < 100 seconds at a 30 MB buffer. *Epidemic* isn't included due to its extremely low delivery probability and high latency.

Figure 5 presents a more in-depth protocol latency analysis for 10 MB buffers. Frequencies have been normalized so a direct comparison can be made. *MaxProp*'s count is about $2,300$ compared with approximately $50,000$ for the others. A closer look uncovers protocol and simulation behavior. For example, all protocols, with the exception of *MaxProp* deliver a proportionately high number of packets in the 0.125 second bin indicating the likelihood that source and destination nodes are within a 2-hop range at the time of packet creation. The

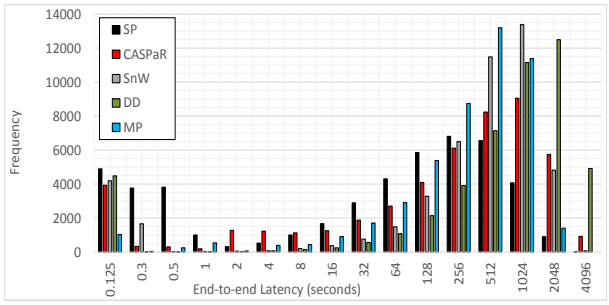


Fig. 5. The frequency of latency distributions show in which time bin packets are delivered for the compared protocols. 17 runs for each protocol (except for *MaxProp*) is included in the analysis. The x-axis bin size is in log base 2 format to accentuate low latencies.

frequency of delivered messages in the $0.125 - 1$ second bins drops quickly for all protocols except *Shortest Path* possibly revealing the existence of multi-hop connected paths packet creation time and provide a multi-hop latency measurement of < 1 second. > 1 second bins are most likely a convoluted measure of the average length of time routes remain disconnected as well as protocols ability to move packets closer to their destination across disconnected paths. If so then *Shortest Path* provides a good performance indicator and comparison tool.

C. Overhead Ratio and Hop Count

Overhead ratio is proportional to energy expenditure and defined by ONE to be $O_r(t) = (P_r(t) - P_d(t))/P_d(t)$ where P_r is the total number of packets relayed by time t and P_d is the total number of packets delivered by time t. Not shown in Figure 6 are: 1) *MaxProp* - overhead > 1700, 2) *Direct Delivery* - overhead always 0, 3) *PRoPHET* - overhead about 6 but has high variation (about $+/-12$) and 4) *Epidemic* - overhead > 4500. Since *CASPaR* and *Shortest Path* are one-

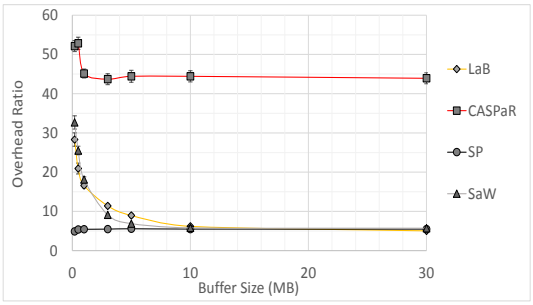


Fig. 6. The overhead ratio required to transfer a packet from source to destination as a function of buffer size.

copy protocols, the overhead is proportional to packet hops. Figure 6 shows this to be about 40 while *Shortest Path*, *LaB*, and *Spray and Wait* all maintain overhead ratios of about 5. Figure 7 shows that *CASPaR's* overhead ratio is reduced at the expense of latency and delivery probability and vice versa. Hop count is the number of nodes packets traverse from source to destination. The final transfer to destination isn't considered a hop and therefore *Direct Delivery*'s hop count is always 0.

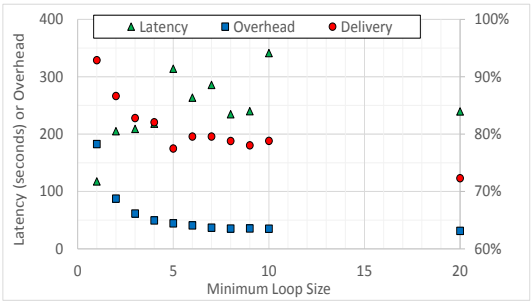


Fig. 7. As the minimum loop count decreases, the overhead increases but median and average latencies decrease while delivery probability increases.

Figure 8 shows the average number of packet hops in the protocols tested. *Shortest Path* indicates the optimum average hop count to be about 6.

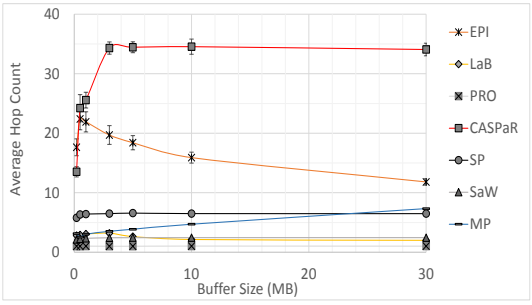


Fig. 8. The average number of nodes a packet traverses from source to destination as a function of buffer size.

D. Route Distribution

Presumably, given a homogeneous set of packet destinations, the more equally packets are distributed across a network, the more efficiently it will function at high loads. Figure 9 shows average queue deviation where *CASPaR* looks

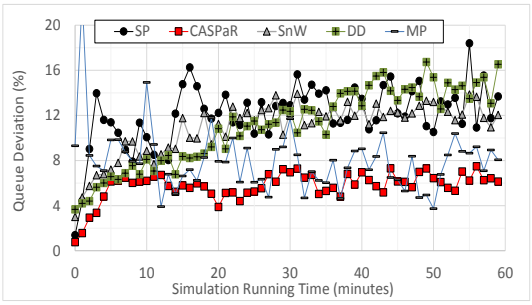


Fig. 9. Queue deviation integrated over 1 minute periods for all 60 minutes of the simulation.

to more evenly distribute packets than compared protocols. *CASPaR*'s variation is half that of *Direct Delivery* and *Spray and Wait* and a bit lower than *MaxProp*.

However, *Shortest Path* experiences the largest variation and yet by every metric, it out-performs all protocols. This indicates that either high-performance does not depend upon an even distribution of packets across queues or that the network load applied during testing wasn't heavy enough to highlight the property. From results reported here, the stated goal of minimizing latency and maximizing delivery can not be met without compromise. *Direct Delivery* has 0 overhead but performs poorly in regards to latency and delivery probability. *MaxProp* delivers many packets at low latencies but requires a larger buffer and high overhead. *CASPaR* is capable of out-performing tested DTN protocols while maintaining relatively low overhead.

V. Conclusion

We have developed an extensible one-copy protocol that can handle a heavy network-load using small network resources. We have shown that *CASPaR* improves network performance where *Spray and Wait* and *MaxProp* also perform well under the same experimental conditions but require larger buffers and in the case of *MaxProp*, a much larger overhead. Future *CASPaR* studies: 1) Various simulation parameters should be applied to more extensively test *CASPaR* 2) A multi-path *CASPaR* variant that out-performs its single-path sibling was developed during the this study and should be further explored. 3) An investigation into a hybrid protocol; a combination of *CASPaR*, *MaxProp* and *Spray and Wait* could yield interesting results. 4) Determine the performance of *CASPaR* and *Shortest Path* at higher network loads. Finally, an analytical model capable of closely approximating the experimental algorithm and its results must be developed.

References

[1] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[2] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *INMIC*, pages 62–68, 2001.

[3] Z. J. Haas, M. R. Pearlman, and P. Samar. The zone routing protocol (zrp) for ad hoc networks. IETF Draft, 2002.

[4] R. J. D'Souza and J. Jose. Routing Approaches in Delay Tolerant Networks: A Survey. *International Journal of Computer Applications*, 1(17):8–14, February 2010.

[5] A. P. Silva, S. Burleigh, C. M. Hirata, and K. Obraczka. A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Network*, 25(1):480–494, Aug. 2015.

[6] M. Alresaini, M. Sathiamoorthy, B. Krishnamachari, and M. J. Neely. Backpressure with Adaptive Redundancy (BWAR). In *INFOCOM*, pages 2300–2308, FL, USA, March 2012.

[7] A. Keränen, J. Ott, and T. Kärkkäinen. The one simulator for dtn protocol evaluation. In *International Conference on Simulation Tools and Techniques*, pages 1–10, 2009.

[8] R. S. Mangrulkar and M. Atique. Performance evaluation of flooding based delay tolerant routing protocols. *International Journal of Computer Applications*, pages 35–40, February 2012.

[9] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke Tech Report CS-2000-06, 2000.

[10] A. Lindgren, A. Doria, and O. Scheln. Probabilistic routing in intermittently connected networks. In *ACM SIGMOBILE Mobile Computing and Communications Review*, pages 19–20, 2003.

[11] J. Burgess, B. Gallagher, and D. Jensen. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *IEEE Infocom*, 2006.

[12] 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*, pages 252–259, 2005.