Reality-Check for DTN Routing Algorithms

Arshad Islam Marcel Waldvogel
University of Konstanz, Konstanz, Germany
<first>.<last>@uni-konstanz.de

Abstract

Many applications of ad-hoc networks include intermittent connectivity. Anyone wishing to implement routing into her delay-tolerant network can select from a wide variation of options, but the choice is hard, as there is no strong comparative evidence to the relative performance of the algorithms. Every paper uses a different setting, mostly far from realistic. In our desire to improve the basis for decisions, we simulated a promising selection of DTN routing algorithms in three vastly different scenarios, all based on publicly available real-world traces. Using our open-source DTN simulator, we compare and analyse 11 routing techniques, then provide explanations for the behaviour and give advice for choosing a suitable mechanism. To our own surprise, the results challenge the conventional wisdom gained from synthetic simulations and poses the question whether the world is ready for DTNs.

1 Introduction

The presence of an increasing number of programmable mobile devices has caught on the desire to use them to that effect, and first ad-hoc routing protocols were born, allowing to find an active path to a destination, which can then be used for real-time data exchange. Delay/disruption tolerant networks (DTNs) go one step further and allow messages to be passed even when the destination and/or intermediate nodes are not currently online. For many data exchanges, this additional delay is acceptable; it may even be an advantage, as the possibility to use nodes carrying the data ("data mule" or just "mule") to a destination can provide higher bandwidth than having the data continuously hop from one mobile node to the next.[1]

Several proposals for efficient routing mechanisms have been devised, who claim superiority based on differing experimental or simulated data. Various protocols assume different levels of knowledge available to network nodes, ranging from only the set of nodes currently directly reachable over connectivity history of the node in question and, maybe, other nodes it has seen up to accurate prediction of the future (oracles [6]). Most of the evaluations make over-simplistic assumptions for either network topology or information available to network nodes, or provide access to unlimited information resources.

In this paper, we aim to inspect how the proposed mechanisms compare under a variety of real-world traces, each representing one typical connectivity paradigm. For each of the traces, we simulate how varying the ratio between message size and available bandwidth affects the result. All the results include analysis and explanations, to our own surprise showing that, even under assumptions favourable to high-information protocols, some low-information protocols such as flooding outperform many algorithms under a variety of constellations.

In the next section, we discuss several issues involved in DTNs and provide brief description of several techniques for routing in DTNs that we have analysed in our comparative study. We also mention the implementation adjustments made for each of them in this section. In section 3, characteristics and properties of the traces on which the routing algorithms are applies are presented. Section 4 gives a brief overview of the simulation environment. We attempt to justify the results obtained through our simulations in section 5 and section 6 contains the concluding remarks.

2 Related Work

Usefulness of DTNs greatly depends on the routing efficiency of the network. We may classify the routing schemes into two categories, i.e., with or without replication. Generally speaking, strategies that do not employ redundancy (cf. Figure 1 (b)) use computationally intensive procedures to determine the path for the message [2,6,8]. This path may be calculated at the source and then regularly updated on each hop depending on the network topology and congestion. Mechanism that

[1] A similar idea lies behind the saying "Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway," which is often attributed to Andrew S. Tanenbaum.

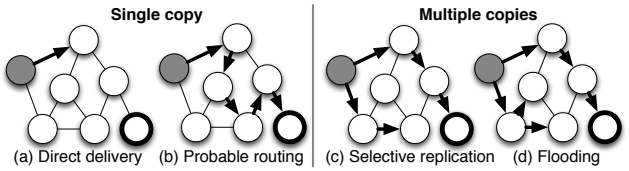


Figure 1. Routing options

involve redundancy try to exploit multiple paths for replicas to enhance the delivery of message through the network [3, 5, 6, 10]. Some algorithms are assisted by oracles that are capable of predicting the network behaviour [6]. Obviously, the quality of prediction depends on the accuracy of the oracle. One can safely assume that the higher the accuracy, the less likely it is to actually find such an oracle.

Figure 1 shows a few routing examples, both for single and multiple message copies. An extreme case of multiple copy also known as flooding, where a source and intermediary nodes deliver a copy of the message to every node which not already has the message. While single-copy algorithms typically require large amounts of bandwidth for information collection, multiple-copy algorithms do so for message delivery.

2.1 Simulated Algorithms

Before presenting the descriptive summary of all the simulated algorithms, qualitative summary in Table 1 shows, Flooding has high performance on the cost of storage as well as communication in contrast to Perfect Oracle, that achieves high performance on the cost of processing and communication overhead. MaxProp on the other hand, has an impressive performance just with processing cost. Estimated Erasure Encoding is successful in reducing storage and communication overhead due to inherent advantages of erasure encoding. Please find a quick description of the algorithms taken from the literature that we simulated. For more details, we refer to the actual publications.

Direct Delivery. The source holds the data until it comes in contact with the destination. Direct Delivery uses minimal resources since each message is transmitted at most once. However, it may incur long delays [10] and frequently shows poor performance (Table 1).

Flooding/Epidemic Routing. Each node forwards all the non-duplicated messages (including messages received on behalf of other nodes) to any other node that it encounters. Flooding has the potential to deliver messages with the minimum delay if there are no resource constraints, such as link bandwidth or node storage [9, 10]. In our implementation, flooding avoids transmitting a message to a devices which already has a copy using the ihave/sendme model [**?**].

First Contact Routing. Messages in this scheme follow a seemingly random path determined by a hot-potato algorithm. The next hop is chosen randomly from the available neighbours, if any. Otherwise, it is handed off to the node coming into proximity first [6]. This phenomenon can cause the message to hop among a group of two or more peers for a long time until the one having the message leaves the group. To reduce this overhead, our simulation prevents returning the message to one of the previous 10% of the hops the message has travelled. The choice of a next hop does not try to make progress towards the destination; therefore, messages may aimlessly propagate through the network.

Simple Replication. This is a simple replication strategy in which identical copies of the message are sent over the first r contacts, with r known as the replication factor. Only the source of the message sends multiple copies, the relay nodes are allowed to send only to the destination; they cannot forward it to another relay. This makes it a mixture between direct delivery and flooding [10]. This algorithm has medium consumption of bandwidth and storage.

History-Based Simple Replication. In this technique, the source creates r identical copies of a message, who are then delivered to the "best" r nodes, where quality is determined by history. The intermediate nodes will then each perform Direct Delivery. Our simulation follows the ZebraNet model of relying on the frequency at which a node has encountered the destination [5, 10].

History-Based Erasure Coding. This mechanism works very similar to history-based simple replication, but kr fragments totalling r times the message size are generated and sent to the best kr intermediate nodes. The intermediate nodes will deliver only to the final destination, where any k fragments can reconstruct the message. This has the same performance as Simple Replication when the path failure model is Bernoulli and the contact volume is sufficient for an entire message [10].

Estimation-Based Erasure Coding (EBEC). History-Based Erasure Coding is an all-or-nothing function: The nodes with highest probability get all the data and path length is limited to two hops. EBEC [7] is more adaptive, as the two communicating intermediate nodes exchange data until the number of fragments for a given destination is proportional to the nodes' probability of meeting the destination. To accelerate the simulation, history is calculated at intervals of 5 minutes and history oracle is used.

Algorithm	Knw	Proc	Stor	Com	Perf
Direct Delivery	+	+	+	+	+
Flooding	+	+	+++	+++	+++
First Contact	+	+	+	+++	+
Simple Replic.	+	+	++	+	++
History-based R.	++	++	++	+	++
History-based EC	++	+	++	++	++
EBEC	++	+++	++	++	+++
Mobile Vehicle	++	++	++	+	+
MaxProp	++	+++	+	+	+++
Perfect Oracle	+++	+++	+	+++	+++
Imperfect Oracle	++	+++	+	++	+

Comparing required knowledge, processing amount, storage, communication needs, and performance

Table 1. Algorithm Characterization

Mobile Vehicle Routing. The routing decision is based on finding a peer that has the highest probability of visiting the region of the destination. In our simulation, we allocate each peer to a home access point or cell tower, depending on the time they have spent with different access points or cell towers. Then we try to find out the peer that is most probable to visit the home region of destination [3]. Both the source and the selected node try to perform Direct Delivery to the destination, which results in a slightly higher resource consumption than Direct Delivery alone.

MaxProp Routing. MaxProp attempts to forward the message to any device that has the greater probability to deliver the message to destination. MaxProp involves calculating the path for each message at each transfer opportunity using a modified Dijkstra algorithm with history as pivotal criterion. MaxProp defines its own way of computing history to dictate the path computation but it is assumed that topology information does not consume bandwidth. It also incorporates a fancy mechanism of message queuing at peer level that prefers the newly born messages and degrades the priority of messages based on the number of hops they have travelled and the delivery probability [2]. Even without the computational complexity of erasure coding, MaxProp is hungry for processing resources as the maintenance of the local queue is expensive for mobile devices under high message counts.

Earliest Delivery a.k.a. Perfect Oracle. The path of a message is computed using a modified Dijkstra algorithm [6], where the link costs represent the waiting time for the next contact between the vertices. It assumes a contact oracle which has perfect foresight of future node encounters, equivalent to knowing the time-varying DTN multi-graph. This algorithm is bound to perform better than all of the others because it has the unrealistic knowledge of the future. A message may still fail to reach the destination due to complete lack of a path to destination or congestion.

3 Trace Description

We have considered three different kinds of data sets, all of which have been obtained from CRAWDAD.

Bluetooth ("MITBT"). The MIT Bluetooth trace spans around 16 months, i.e., from February 2004 to August 2005. Each device scanned every five minutes for active Bluetooth neighbours. We limited ourselves to one month of connectivity trace, where any visible Bluetooth device was considered a candidate connection. We selected the month with the largest activity, November 2004, for our simulation. Even then, the resulting connectivity remains sparse.

Access Points ("IBM"). In the case of IBM Access Point trace, SNMP was used to to poll access points every 5 minutes, from July 20, 2002 through August 17, 2002. A total of 1366 devices have been polled over 172 different access points during approximately 4 weeks. To turn these samples into continuous data, we assume that the snapshot data will remain constant for the next 5 minutes. In the rare cases where this would cause an overlap with another snapshot from another access point, we assume that the transition happens halfway between the two snapshots.

Cell Towers ("MIT"). This trace again holds exact timings, relieving us from such plays. Due to several lapses in data gathering, mentioned by the creators of the data, only 89 of 100 devices are included, which see 32768 different cell towers. As expected, November 2004 came out to be the month with maximum activity here as well. We observed 81 devices and 12592 different cell towers. Our connectivity model assumes that nodes registered at the same tower (actually, antenna) could communicate with each other. The resulting connectivity is the most dense of the three. Other analyses we performed with the second-most active month, October 2004 with 79 devices and 11784 antennae, resulted in similar performance for the algorithms, supporting our thesis that November is nothing special.

Comparison. From our analysis of interaction patterns, we know that in the Cell Tower case (both for the selected month of November as well as the preceding month), essentially all active nodes could directly communicate with every other node at least once during a month. In the Access Point case, the majority of 928 devices were able to contact 50...100 devices, many of them up to 200.

In the Bluetooth scenario, the 1858 nodes get in touch with at most 100...200 nodes, the majority only with 5...50. Interestingly, some of the highest-ranking devices were not one of 89 participant nodes, but showed up more frequently in the communications range of trace group members than other members. We

Message count	100
Message size	1.6E3...1.6E7 B
Size distribution	Power law
Replication	$r = 4$
Erasure coding	$k = 4$
Bandwidth (low)	100 kiB/s
Bandwidth (med)	1000 kiB/s
Bandwidth (high)	10,000 kiB/s

Table 2. Simulation parameters

have to expect that these non-participants had contact with many other non-members; unfortunately, there is no way to tell given these traces.

We can see that the wider the communication range, we increase the likelihood to communicate among peers and thus make the network more dense, creating an increasing density from Bluetooth to Cell Towers. On the other hand, the data communication rate is mostly lower for long-distance networks (e.g., Access Points and Cell Towers; not so much for Bluetooth vs. Access Points). For current networks, the effects of the density (or lack thereof) seem to dominate the bandwidth effects.

4 Simulation Setup

Basics. The aim of our simulator is to help us find the delays incurred by messages during execution of different routing algorithms. The output is analysed on the basis of both number of messages as well as amount of data delivered. As already mentioned, three different traces have been used that significantly differ in the number of devices involved as well as the number, frequency, and distinctness of meetings that were taking place among the participants. As the span time of Access Point trace is approximately one month whereas for Cell Tower and Bluetooth traces is more than one year, we have chosen one month from Cell Tower and Bluetooth data on the basis of highest activity, so that the results can be compared. We observed that November 2004 had the highest activity[2] among all the month for which Cell Tower data has been recorded. Furthermore to see the correlation with other months, for Cell Tower trace, we decided to include the month that ranked second, i.e., October 2004.

A detailed discussion of the simulation configuration can be found in [4].

Imperfect Oracle. In analogy to the observations of Apostolopoulos et al. [1] for frequent link QoS updates, frequent transmission of connectivity changes or history will severely reduce the network bandwidth, even more so in a mobile environment where dynamics are high and bandwidth scarce. The simulator assumes that these topology exchanges happen out-of-bound, clearly unrealistic, especially for Earliest Delivery. We therefore wanted to examine the impact of small prediction errors on the routing performance. To do this, we created two versions of Earliest Delivery: Perfect Oracle, which corresponds to the scheme presented by Jain et al. [6], and Imperfect Oracle, described below. To bring Perfect Oracle at par with real world scenarios, we have created an imperfect contact oracle that share most of the properties of contact oracle but its accuracy for predicting the future is 1/3 of the contact oracle. We have introduced 2 different types of weak errors in the contact oracle through the following mechanism, each responsible for modifying the predictions of 1/3 of the devices.

Mistiming: Assume that the start and end times of contacts may be off. This is done by randomly moving the start or end point while maintaining the middle of the active period as active and the middle of the idle period as idle. This ensures that two devices will still meet, but maybe somewhat earlier, later, longer, or shorter.

Systematic errors: Exchange the timelines of two similar devices, namely the ones that have seen each other the most number of times. This is comparable to someone changing his habits and, while still a weak modification, stronger than mistiming.

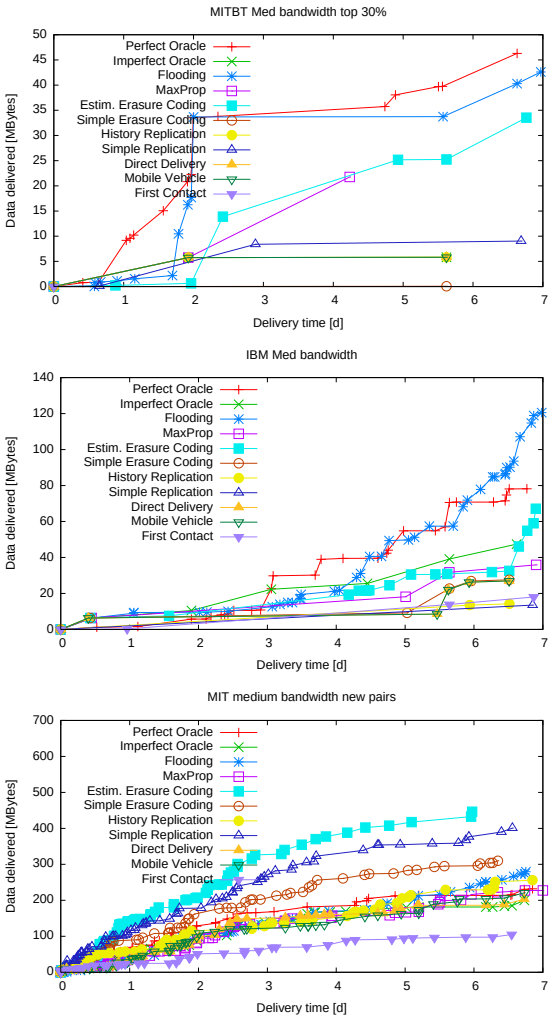
We have, in this way, tried to introduce the alterations, that are in line with the system and may be somewhat closer to what a realistic contact oracle can achieve. Here again, whenever the system realises that a packet has failed to take a hop or the hop was not available at the predicted time, then a new route to destination according to imperfect oracle is computed.

5 Results

Figure 2 shows how well messages are delivered in the medium bandwidth case for the three environments. For example, in the MIT Bluetooth case (top), we can see that Perfect Oracle early on starts delivering messages and after almost six days, it has delivered close to 50 MBytes, about 10% of the total message load. The runner-up, Flooding, starts a little bit slower, but after 2 days, it has caught up and will remain close to the winner.

In the IBM Access Points scenario (Figure 2, middle), however, Flooding, Direct Delivery, Mobile Vehicle, Imperfect Oracle, and EBEC all deliver their first

[2] Activity is defined as time spent "online" by devices, i.e., being connected to either cell towers or other neighbouring devices.



More results can be found in [4].

Figure 2. Performance of DTN data transfer

big message half a day after it was injected into the network; Perfect Oracle requires about four times as long and four smaller messages until it reaches the same volume. In the meantime, its ugly stepbrother Imperfect Oracle has taken the performance lead for a day, but after seven days, Flooding has delivered roughly 25% of the data, collecting the Maillot Jaune. In the MIT Cell Tower scenario (bottom), things look very different again: EBEC leads the pack with a considerable Simple Replication start catching up then, but remain without chances.

How can this be? Why does this happen? Why does the Perfect Oracle behave so poorly despite its omniscience? The following paragraphs will try to answer these questions.

Omniscience ... is not all. First of all, Perfect Oracle is *not* omniscient, it lacks knowledge about concurrent traffic so it cannot avoid bottlenecks. Even worse,

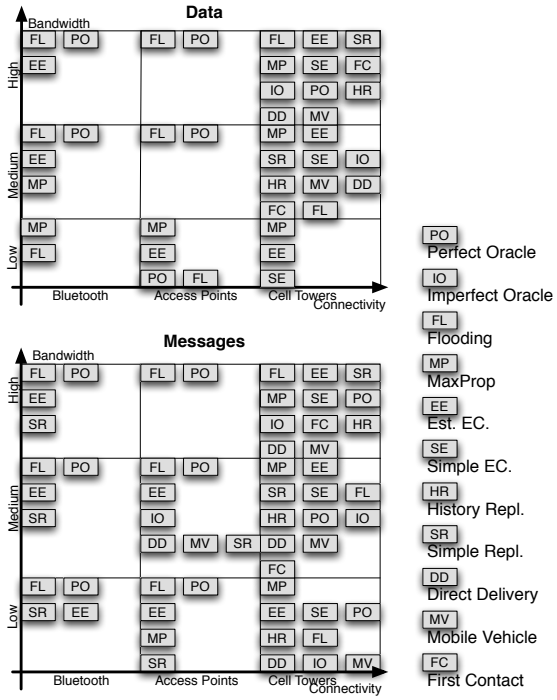


Figure 3. Performance of Routing Algorithms

it does not include message size into the calculation, which can result in the choice of a path which does not provide long enough connection times to transmit the message even in the absence of other messages. The latter could be avoided, but not the former.[3] It also seems that the perfection is bad, as the selection of the "best" path is predictable. Even in the Cell Tower case, connectivity seems to be sparse enough to create a few attractive bottleneck links, through which large portions of the traffic should be funnelled. When this fails, the remaining messages need to be rerouted, probably again along similar routes, creating more bottlenecks.

This is where other algorithms including Flooding take control of the network. In a well-connected environment such as Cell Towers, Direct Delivery and its two-hop cousin Simple Replication perform well, the latter delivers about 80% of the data; even the hot potatoes from First Contact manage to reach 20%.

Comparison. To answer further questions, we compiled a figure of merit, Figure 3 , summarising all the best algorithms in each of the nine bandwidth/ connectivity areas. In each square, the top row corresponds to the top performers, which are roughly on par with each other. The second row describes the second group, and so on. Algorithms not mentioned in a

[3] It would probably require a high-speed ubiquitous wireless network for topology/traffic information exchange. If you have such a network, why not use it for the actual data?!

square perform very poorly.

What we can see is that only 5 of the 11 algorithms make it ever to the top: Flooding and Perfect Oracle dominate under weak connectivity, while MaxProp works well under high connectivity. Flooding, EBEC and Simple Replication can take advantage of "nice" networks (high bandwidth, high connectivity). As Simple Replication gives opportunity to each replica of the message to take at-least one hop, therefore, in a dense network like Cell Tower, its performance in high bandwidth case is very impressive.

MaxProp performs very well when bandwidth is low, but does not seem to be able to take advantage of higher bandwidths: It delivers not significantly more despite 10- or 100-fold increases of bandwidth. Its noteworthy that MaxProp is the only algorithm that defines the queue management mechanism for one device, that apparently gives it an edge. This technique gives priority to messages that are destined for next hop and sorting the remaining messages according to their age, i.e., younger messages are given priority for next transfer opportunity. We believe that this strategy provides quick delivery of fresh messages while being persistent about older messages. Together with the fine-grained proportional message sharing, this ensures that many pieces of the message follow the best path quickly.

MaxProp also employs a unique strategy to compute history information that in our view helps the algorithm to better determine the candidate next hops. We believe that the history normalisation employed by MaxProp helps reducing the chance that too many messages will be loaded onto a very mobile device, which does not stay long enough in the vicinity of the destination to actually deliver all messages.

If you wanted to implement only one algorithm, hoping it would perform reasonable under most regimes, Flooding and EBEC are the candidates to consider (Perfect Oracle can be excluded, as it seems impossible to implement the necessary oracle in real life).

6 Conclusion

To our knowledge, this is not only the first comprehensive comparison and analysis of DTN routing algorithms, but also the first to approach realistic communication models. To our own surprise, many of the simpler algorithms, under the broad leadership of Flooding, perform among the best in all classes.

To be among the best here does not imply to be good. For many scenarios, the performance even of the best is lousy, especially at low bandwidths or low connectivity. Therefore, a huge challenge lies before us, namely, *to make the world ready for DTNs* by creating higher densities and higher bandwidths. Even with a tenfold increase, waiting times of around a day and success rates between 5 and 80% will still be common, which does not make it ready for prime-time yet.

Social algorithms may have to be included in the list, where the device knows more about its human carrier and her friends than just statistics. Coupling with a calendar and other information resources may open new avenues, but retaining privacy under these circumstances will be a daunting task. So far, we have no clear winner, but many good candidates. Each of them probably has a home turf on which it performs extremely well. Even with the DTN community's goal to find the all-encompassing solution, many niches will remain, where specialised algorithms will always excel.

References

[1] G. Apostolopoulos, R. Guérin, S. Kamat, and S. K. Tripathi. Quality of service based routing: a performance perspective. *SIGCOMM Comput. Commun. Rev.*, 28(4):17–28, 1998.

[2] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for Vehicle-Based Disruption-TolerantNetworking. In *Proceedings of IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[3] B. Burns, O. Brock, and B. N. Levine. Mv routing and capacity building in disruption tolerant networks. In *Proceedings of IEEE Infocom*, 2005.

[4] A. Islam. Nuntix-modeling of delay tolerant networks. Technical report, University of Konstanz, 2008.

[5] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in a delay tolerant network. In *Proceedings of ACM SIGCOMM*, pages 109–120, New York, NY, USA, 2005. ACM Press.

[6] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. 2004.

[7] Y. Liao, K. Tan, Z. Zhang, and L. Gao. Estimation based erasure-coding routing in delay tolerant networks. In *Proceedings of IWCMC*, June 2006.

[8] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Proceedings of MobiHoc (Poster)*, 2003.

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

[10] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. *ACM workshop on Delay Tolerant Networking*, 2005.