Routing in Large-Scale Buses Ad Hoc Networks

Michel Sede[†], Xu Li[*], Da Li[*], Min-You Wu[*]

[†] Laboratory for Computer Communications and Applications, EPFL, Switzerland

[*] Department of Computer Science and Engineering, Shanghai Jiao Tong University, China

***Abstract*— A Disruption-Tolerant Network (DTN) attempts to route packets between nodes that are temporarily connected. Difficulty in such networks is that nodes have no information about the network status and contact opportunities. The situation is different in public bus networks because the movement of buses exhibit some regularities so that routing in a deterministic way is possible. Many algorithms use a Contacts Oracle that provides the exact meeting times and durations between all nodes. However, in a real vehicular environment, an oracle is not always accurate, and deterministic routing gives poor results. In this paper, we present BLER, a routing algorithm that achieves effective routing in a buses environment. BLER, compared to other algorithms, performs routing at bus line level instead of bus level; it uses specific bus lines informations to achieve good performances. We evaluate BLER on real traces of the bus network of Shanghai, and compare it to other routing algorithms. Performances provide good results for this kind of DTNs.**

***Keywords*— BUSNET, Disruption-Tolerant Networks, Routing, Vehicular Ad Hoc Networks.**

I. Introduction

Traditional wired networks provide constant connectivity between nodes; these are increasingly being extended to include wireless links, thus allowing to save space, time, and to add a mobility dimension. *Disruption Tolerant Networks* (*DTNs*) enable routing of information in networks where end-to-end paths are unstable and varying over time [3]. Applications of DTNs are various and at different scales: oceanic sensor networks, communications between Low-Earth Orbiting Satellites (LEO), asynchronous Internet access to villages far from large cities [1], or large-scale disaster recovery.

An emerging field for development of DTNs is *vehicular ad hoc networks* (*VANETs*), such as cars or buses networks [10]. In metropolitan zones, public transportation systems cover very large areas, and can thus be used to route information. In this paper, we study the construction of a DTN on the public buses network in Shanghai, that we call ***BUSNET*** (See Fig.1). Delay tolerant applications, such as buses software updates, advertisement dissemination, or other information transmission between buses, can use a DTN rather than location-based services, costly and resource-consuming.

Fig. 1. BUSNET representation

Many previous works provide algorithms in order to find end-to-end paths between nodes [2]. In some cases, having knowledge about the network state allows to perform more efficient routing, that is, to minimize delays and to maximize delivery rates. For BUSNET, we aim to propose a more effective routing than general DTN routing by using some specific knowledge about bus networks, such as bus schedule, bus line information, etc. In ideal case of bus networks, we can make use of an oracle that provides the exact meeting times of mobile nodes, and thus achieve very effective routing [1]. In real environments however, it is very difficult for buses to follow an exact time schedule, and performances are not as good as expected because poor traffic conditions always make buses miss the exact schedule; especially, this is a very common phenomenon in downtown area in most of cities.

In our work, we first show that deterministic routing in a large-scale environment like BUSNET does not perform as good as a non-deterministic algorithm. We then present **BLER** (***B****us* ***L****ine-based* ***E****ffective* ***R****outing*), an algorithm specially designed for BUSNET that considers routing at a *bus-line-level* instead of *bus level*. By comparing BLER with other routing algorithms, we show that it yields good performances and is suitable for a buses network.

We first present some previous works related with DTN routing (section II), we then show an overview of BUSNET (section III). After introducing our new algorithm (section IV), we test and comment the performances (section V).

II. Related Work

In this section we briefly describe some existing routing algorithms and the way they work. A routing algorithm is said deterministic if it has partial or full knowledge about the network state (in the BUSNET case, all buses movements and meetings); a routing algorithm without any knowledge is said stochastic or non-deterministic.

A. Epidemic Routing

In Epidemic Routing [6,7], a node forwards its packets to every node it meets. To avoid network flooding, each packet has an expiration time, after which it is dropped by the node. To stop the epidemic, the VACCINE method can be used: once a packet is received at destination, the latter starts propagating vaccine packets to all its neighbors; a vaccine packet is an acknowledgment that makes nodes drop a corresponding epidemic packet.

This non-deterministic algorithm is optimal if we assume very high bandwidth and infinite buffer size.

B. Oracle-based Dijkstra algorithms

Using a Contacts Oracle, i.e. an oracle that gives the meetings and durations of all pairs of nodes at any time, we can use the Modified Dijkstra algorithm to compute a shortest path from a source to a destination. There are many different algorithms using Dijkstra [1], among which we retain *Earliest Delivery*. The route for a packet is computed once at source and is fixed (source routing). A Contact Oracle is defined in the following way: let Q be the set of nodes (buses), and T the time interval of the simulation:

$$O=(i,j,s,e) \vee i,j \in Q, s,e \in T, i \neq j, s<e \qquad (1)$$

The element *(i,j,s,e)* in Equ. (1) defines a single contact between node i and node j, starting at time s and ending at time e.

Paths computed by Earliest Delivery do not take into account queuing delays. If a contact is said to be available at a certain time, the algorithm assumes that this contact can be used to transmit a packet. Although it greatly improves delivery delays, Earliest Delivery algorithm becomes disastrous if one contact in a path is missed for some reason; the node will have to wait until the contact appears again, which may never happen.

C. Global Oracular Algorithm

The main goal of deterministic Global Oracular algorithm [5] is to build a tree, using a Contacts Oracle: the root of the tree is the source of the path, and every leaf is a destination. The algorithm considers all edges of all nodes to build subtrees; with a complexity of $O(N!)$, N being the number of buses, the resulting structure cannot be handled for large-scale systems like BUSNET. The authors of Global Oracular only provide a simple example (eight nodes network), but no serious performance evaluation of the algorithm.

D. Shortest Path in Space and Time (SPST)

The idea of SPST [4] is to build space-time routing tables for all nodes in a graph. An interesting key of the algorithm is that it considers *colored paths*, i.e. paths designed depending on packets sizes. The authors showed that the algorithm is optimal in the sense that it provides a 100% delivery rate. They use 128 nodes and either a 512-time unit interval, or a periodic (cyclic) nodes meeting schedule. In BUSNET, more than 700 nodes and a 12-hours time interval would produce too massive structures that could not be handled.

E. MaxProp

The *MaxProp* protocol [8] performs routing by considering the priority of packets to be transmitted, and the priority of packets to be dropped. Packet forwarding is made by computing delivery likelihood from a node to all its neighbors. MaxProp has been implemented on a real bus network, and has also shown to perform well in a wide variety of DTN environments. However, MaxProp is a generic algorithm for vehicle-based DTNs; in our work we focus on bus lines-based DTNs, in order to take advantage of organized networks properties (nodes scheduled meetings,...).

F. RAPID

In [9], authors introduce DTN routing as a resource allocation problem. Their protocol, *RAPID*, optimizes an administrator-specified routing metric. Routing is done by packet replication, and by computing per-packet utilities which determine how packets should be replicated. *RAPID* was tested over a vehicular DTN testbed of 40 buses.

III. Overview

In BUSNET, we consider the public buses network of Shanghai, with a set of 30 bus lines, and more than 700 buses. The total covering area of all bus lines is approximately 150 km², as represented in Fig. 3. Each bus line is made of two end-to-end connected routes, that we call upstream and downstream routes, and has a certain number of bus stops; once a bus arrives at the end of one route, it starts running on the second one, ans so on. All buses are equipped with a GPS antenna, and regularly send localization informations to a central server (Fig. 2). These informations mainly consist of the bus longitude and latitude, the bus identifier, the emission time, and the current direction the bus is following (upstream or downstream); other secondary informations such as instant speed, gasoline left in the tank,... are also sent.

In large cities like Shanghai, buses cannot follow a precise time schedule; traffic variations and complexity make it very hard to estimate. On the other hand, the meetings of buses pairs may be regular enough to use a contacts oracle; by knowing in advance when two buses will meet, one can perform very effective routing. In our framework however, we want to demonstrate that the use of an oracle does not improve the routing performances in a real traffic environment, compared to non-deterministic routing.

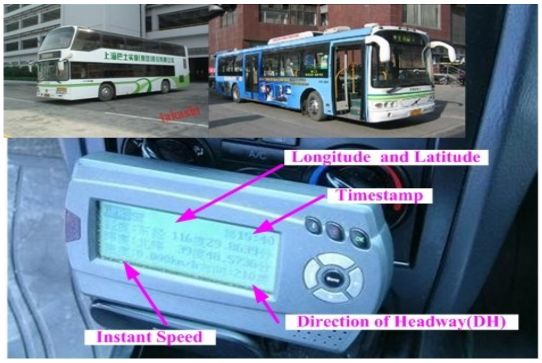

Fig. 2. Public buses GPS equipment

To fulfill this goal, we study routing on two different sets of data:

i. *Real buses data*, that are recorded traces from March 23rd, 2007 to March 26th, 2007 between 08:00:00 and 20:00:00.
ii. *Artificial buses data*, that are computed from the real buses data, and are more regular. In this set, all buses on a particular line run at the same speed, and are equally spaced in time. Buses also wait a constant time at every bus stop.

Fig. 3. Shanghai city map and bus lines

We build our Contacts Oracle thanks to the set of artificial data. As the two datasets are very similar, the oracle is supposed to match both of them. Simulations using these two datasets allow us to prove that exact time schedules for buses is not realistic, so that it is not feasible to apply deterministic routing in real situations.

IV. The BLER Algorithm

In this section, we present our new algorithm, Bus Line-based Effective Routing (BLER). Performances are discussed in Section V.

The main problem we have when using a Contacts Oracle in a vehicular environment is that the buses rarely respect the meetings schedule. Contacts between real buses are, compared to the Oracle's, time-shifted or inexistent for many reasons:

- inconstant speed and waiting times at bus stops
- delays due to car accidents, gasoline refill, detours,...

For all these reasons, an accurate mapping from artificial data to real data is not feasible. Also, computing shortest paths with Modified Dijkstra is not a feasible solution, since exact locations and contacts occurrences are required.

In BLER, we do not consider paths among buses, but paths among bus lines. The nodes are no more individual buses, but buses on a specific bus line. We introduce a new kind of oracle: the *Route Contact Oracle*. This oracle is computed based on the artificial data's Contacts Oracle. It specifies, for two routes, the total duration of the contacts among pairs of buses that belong to these routes. More formally, this oracle models a static graph G such as:

- every node of G is a bus line (30 nodes for BUSNET)
- node a is linked to node b if there exists at least one contact between buses i, j, such that i belongs to bus line a and j belongs to bus line b.
- The weight of an edge between two nodes a and b is equal to the sum of all contacts lengths between route a and route b, as defined in Equ. (2).

$$W(a,b) = \Sigma\, contact(i,j).length \vee i \in a,\, j \in b \quad (2)$$

A large weight means that two bus lines have buses that meet frequently. Now, the problem of finding a shortest path between two buses at a particular time is simplified to finding a shortest path between two bus lines at any time. We can then simply apply original Dijkstra's algorithm on the previous graph, and find paths that maximize the contacts lengths. For a given packet, a path is a list of bus lines identifiers. In Fig. 4, a path from bus line 3 to bus line 145 goes through lines 18 and 9, because this path maximizes the total contacts lengths, hence the probability of having meetings.

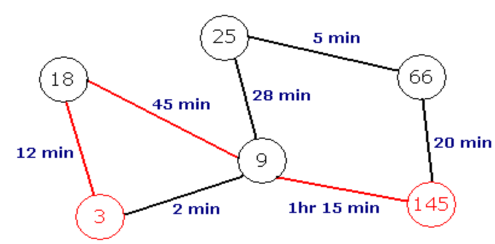

Fig. 4. Bus lines graph example

Routing is performed in two steps. The first step is to route a packet to the destination bus line; to do this, a bus considers only the next bus line ID in the packet path; once it meets a bus belonging to that line, it forwards the packet. Every hop is from one bus line to another, and a packet always stays on the same bus until next bus line hop.

When the packet reaches the destination line, the second step is to route it to the destination bus. This is done by what we call the *zigzag process*: the buses transmit the packet to other buses running on the same line, but in opposite directions. The packet then zigzags in the same area, jumping from one bus to another. Since all buses of a given line run on

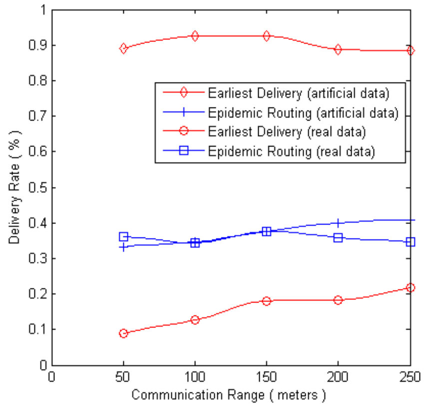

Fig. 5. Delivery rates for routing on real and artificial data

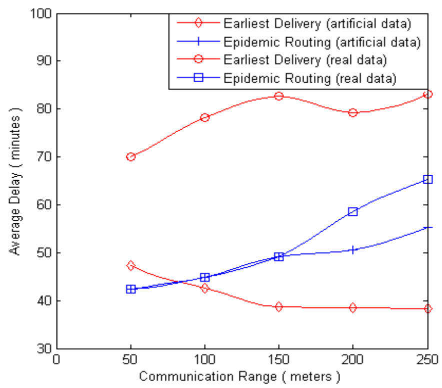

Fig. 6. Average delays for routing on real and artificial data

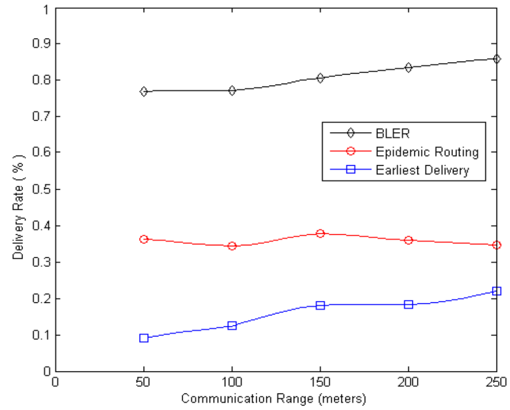

Fig. 7. Delivery rates on real data

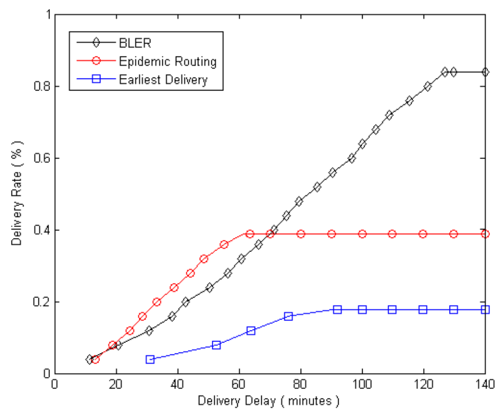

Fig. 8. CDF of packets delivery delays, real data

the same road segments, this guarantees that a packet will inevitably reach the destination bus some time.

V. Performances

Now that we have defined an algorithm for BUSNET, we first evaluate the previous algorithms, Epidemic Routing and Earliest Delivery, on the artificial dataset and the real dataset. Then we test all algorithms performances on the real dataset, and finally we evaluate the BLER algorithm by varying some parameters. We simulate a 12-hours buses traffic, between 8:00 and 20:00. Every bus is a mobile node that can store, carry and forward packets of informations. A node can transmit a packet to a neighbor node by establishing an ad hoc connection, if it is in the communication range; nodes also have a buffer to store intermediate packets, until the correct destination is reachable. If a buffer is full, then every new arriving packet is dropped. The network load is a set of 1000 randomly generated packets; these have random source and destination, a random sending time uniformly distributed over the 12-hours interval, and a fixed size of 128 KB. Buffers size is 1 MB. The communication range varies from 50 to 250 meters. For each algorithm and each set of parameters, we run 10 simulations.

A. Simulations on real and artificial data

As the Contacts Oracle is computed from the set of artificial data, the delivery rate for Earliest Delivery (Fig. 5) is very high because all contacts are scheduled at the right time. However, once the simulations are done on real data, performances are much less better. Earliest Delivery suffers from the fact that a missed contact makes the node keep the packet until the same contact appears, and that may never happen. On the other hand, Epidemic Routing provides good delays, but the network is quickly flooded, which leads to poor delivery rates (Fig. 6).

B. Algorithms comparisons

We clearly see that the delivery rates are much more better on the real data when using BLER algorithm (Fig. 7). Suppressing the contact time constraint of Earliest Delivery offers more flexibility and contacts alternatives to the nodes. Delivery delays are however very high compared to Earliest Delivery and Epidemic Routing; the most time-consuming part of BLER is the *zigzag process* : until it reaches the destination bus, a packet continues to jump from one bus to another; on large bus lines, this process can take more than an hour, since there is no clue of the destination's position. The Table 1 shows, for each algorithm, the average number

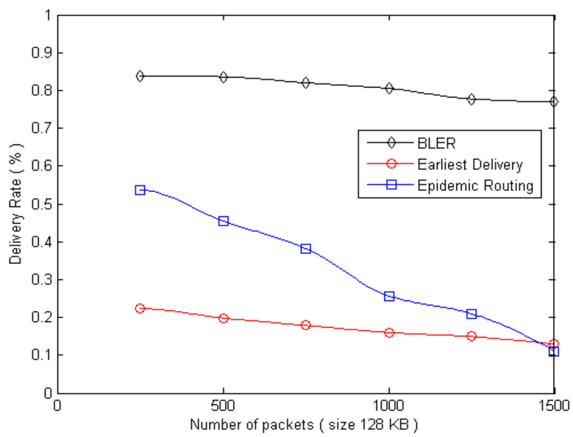

Fig. 10. Delivery Rates for different number of packets

of hops for a packet from source to destination.

	Earliest Delivery	Epidemic	BLER-e	BLER-i
# Hops	5.5	6	1.5	6.5

Table I: Comparisons of average number of hops for packets, from source to destination

With BLER, any pair of bus lines can be connected in a very small number of hops (BLER-e, inter-route forwarding); source bus line and destinations bus line can very often be joined by one or two hops; this is due to the density of the bus lines graph (see Fig. 3), and the fact that a lot of bus lines have many common road segments, hence are connected to each others. But for delivering the packet to the destination, the zigzag process takes much more hops to reach a specific destination bus (BLER-i, intra-route forwarding). Also, the packets that are zigzagging may not reach their destination, because some buses may not be on duty after a certain time, or the simulation ends before the packet gets to the last node. In Figure 8, we plot the Cumulative Distribution Function of the delivery delays, using a 150 meters communication range, and a network load of 500 packets of size equal to 128 KB. Half of the packets of BLER are delivered with large delays (>90 minutes). However, we also have to consider the fact that Earliest Delivery provides very poor delivery rates, whereas BLER produces delays similar as Epidemic Routing and better than Earliest Delivery, but with much better delivery rates (>80%).

C. Examining other parameters

We examine the effect of varying the network load, i.e. the number of packets to be sent in the 12-hours simulation time interval. We also observe the resources consumption (buffer occupancy) on the buses for the BLER algorithm. All packets have a fixed size of 128 KB; buffer size on all buses is 1 MB, and the radio bandwidth is 1 Mbps. Contrary to Epidemic Routing which replicates packets to all possible nodes, BLER and Earliest Delivery perform source routing, so much less nodes are concerned by routing; increasing the

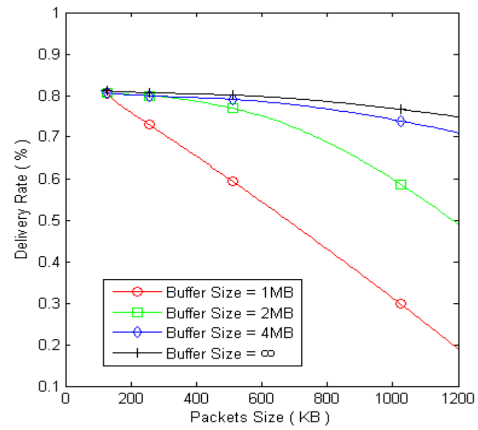

Fig. 11. Varying buffers sizes on BLER

network load in Fig. 10 does not cause delivery rates to decrease dramatically. BLER's routing strategy allows good spreading of packets among the all network's nodes. By observing the buffers and packets sizes variations on Fig. 11, we see that a 4MB-buffer is almost a ceiling for this set of data. Increasing buffer size beyond this value does not influence delivery rates much more; we can then conclude that BLER's performances are not limited by network load, but by intra-route forwarding, as explained in section V.B.

VI. Conclusion

With unpredictable behavior, large-scale buses networks can hardly take advantage of a Contacts Oracle and perform deterministic routing. Exact meetings between buses are difficult to satisfy in real environments. The BLER algorithm suppresses this time constraint by considering paths among bus lines. We have shown in BUSNET that BLER performs better than traditional deterministic routing algorithms, in term of delivery rate. By using a low number of hops (one bus per line hop), network load and resources consumptions are also lower with BLER. Its simplicity make it very effective for routing in vehicular networks on large-scale areas, such as buses running in a big city like Shanghai. With no location-based services or network information, BLER is a well adapted solution for this generation of DTNs.

References

[1] S. Jain, K. Fall, R. Patra, "Routing in a Delay Tolerant Network", SIGCOMM 2004.

[2] E. Jones, P. Ward, "Routing Strategies For Delay-Tolerant Networks", 2006.

[3] Z. Zhang, "Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges", San Diego Research Center, 2006.

[4] S. Merugu, M. Ammar, E. Zegura, "Routing in Space and Time in Networks with Predictable Mobility", Georgia Institute of Technology, 2004.

[5] R. Handorean, C. Gill, G-C Roman, "Accommodating Transient Connectivity in Ad Hoc and Mobile Settings", Washington University in St. Louis, 2003.

[6] A. Vahdat, D. Becker, “Epidemic Routing for Partially-Connected Ad Hoc Networks”, Duke University, 2000.

[7] X. Zhang, G. Neglia, J. Kurose, D. Towsley, “Performance Modeling of Epidemic Routing”, 2006.

[8] J. Burgess, B. Gallagher, D. Jensen, B. Levine, “MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks”, University of Massachusetts, 2006.

[9] A. Balasubramanian, B. Levine, A. Venkataramani, “DTN Routing as a Resource Allocation Problem”, University of Massachusetts, SIGCOMM 2007.

[10] X. Zhang, J. Kurose, B. Levine, D. Towsley, H. Zhang, “Study of a Bus-based Disruption-Tolerant Network: Mobility Modeling and Impact on Routing”, University of Massachusetts, MOBICOM 2007.