Distance Vector Routing in Partitioned Networks

Ammar Farooq* and Murat Yuksel†
Department of Electrical and Computer Engineering
University of Central Florida, Orlando, FL, USA
Email: *farooq@knights.ucf.edu, †murat.yuksel@ucf.edu

2022 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) | 978-1-6654-8353-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/LANMAN54755.2022.9820113

***Abstract*—We present a disconnection tolerant routing protocol, Binary State Distance Vector Routing (BSDVR), that can provide unicast routing on partitioned networks. BSDVR introduces binary state information for distance vector (DV) entries to compute unicast paths even if the network is partitioned. We compare BSDVR against the traditional DV routing (TDVR) in a controlled network. Our experiments confirm that BSDVR generates more control overhead during single link failures that do not cause partitions. In terms of partition-causing link failures, BSDVR's control overhead is less than TDVR's by an order of magnitude, leading to much better convergence times.**

***Index Terms*—delay-tolerant networking; ad-hoc networks; distance vector routing**

I. Introduction

Delay/Disconnection Tolerant Networks (DTNs), as a form of ad-hoc networks, have a vast array of applications where sustainable network infrastructure may be absent, e.g., communication among first responder teams during a disaster, or communication within isolated units in battlefield. Disconnections or long end-to-end delays are the norm in such environments and the network typically forms an ad-hoc opportunistic fashion [1]. Such ad-hoc DTNs are highly dynamic and require specialized routing protocols that operate under resource constraints and can handle frequent disconnections or link failures. Traditionally, two different family of routing protocols operate in such disconnected environments, namely *epidemic* and *distance vector (DV)* routing.

Epidemic routing protocols involve probabilistic exchange of data messages among nodes to increase the delivery rate [2]. Their key advantage is a minimal control state (e.g., no or partial forwarding tables) but they have a costly data plane as every data message may get copied many times during the forwarding process. These copies cause extra load on the network capacity that can become prohibitive in limited-resource networks such as first responders' network during a disaster. To avoid forwarding multiple copies of the data messages, DV routing makes the nodes maintain a localized view to their direct neighbors where routes get retrieved on-demand using the Bellman-Ford algorithm to calculate the shortest paths, with little reliance on periodic advertisements, which reduces the overall bandwidth demand, therefore, making it suitable for highly dynamic networks [3].

This work was funded in part by NIST grant number 70NANB17H188 and U.S. NSF awards 1647189 and 2115215.

However, DV routing requires the underlying network to be connected. To handle disconnections that may cause partitions, most DV-based routing protocols employ on-demand reactive mechanisms involving a signaling phase before transmission of data messages (e.g., AODV [3]) or additional state in packet headers (e.g., DSDV [4]). Without such extra cost, DV routing cannot avoid the well-known count-to-infinity problem when it faces with the problem of finding a unicast path between nodes located in different partitions of a network.

We introduce Binary State DV Routing (BSDVR) protocol that introduces binary state information for each DV entry to avoid the count-to-infinity problem. Even when the network is partitioned, BSDVR successfully calculates the shortest paths among all nodes by augmenting the Bellman-Ford algorithm with the binary state information. With proper tuning of a data plane, BSDVR can attain a hybrid between traditional DV and epidemic routing, and enable *unicast*-like paths that is more suitable for limited-capacity networks. Our contributions are:

- The concept of categorizing DV entries by *active* vs. *inactive*, a.k.a. binary state, based on whether or not the entry corresponds to a connected or disconnected path.
- A detailed control logic to augment the Bellman-Ford's algorithm to process DV entries with binary states.
- An emulation-based evaluation of the proposed protocol.

The rest of the paper is organized as follows: First, we cover related work in distance vector routing in ad-hoc and delay-tolerant networks in Section II. Section III provides an overview of BSDVR and how it differs from traditional DV routing. Section IV details how BSDVR's control plane handles link failures and provides the logic for merging active and inactive states of DV entries as DV update messages are exchanged. Section V presents results from the emulation-based evaluation of BSDVR in scenarios involving single link failures and partition-causing link failures. Finally, Section VI summarizes our work and discusses future directions.

II. Related Work

Mobile ad-hoc routing literature is vast. Since our work focuses on disconnected or partitioned networks, the most relevant routing literature is DTN routing. Due to the difficulty of handling partitions when calculating end-to-end shortest paths, DTN routing literature focused on using epidemic forwarding methods where data messages are probabilistically sent to neighbors with the hope of increasing their delivery rate. These routing protocols aim to develop the most optimal forwarding strategy because overall performance in terms of its delivery

rate and packet loss is greatly affected by how v
relay node gets selected and an optimal data d
policy which determines the number of copies a
message so that its delivery likelihood is maximi

DTN routing protocols can be classified intc
categories, pure *opportunistic* and *social-based*, b
forwarding strategies they employ [5], [6]. The lat
selects relay node based on social group informat
from historical encounters to build multi-level s
[7]. The former uses the same mobility patterns an
history to assign a probability to relay nodes for
message. Only nodes with high delivery probabi
selected to forward a message which reduces the o
of the message in the network.

The opportunistic DTN routing protocols are n
to BSDVRP since they also assume a highly i
connected network. In this group, ProPHET [8
existence of non-random node mobility patterns in intermittent networks by evaluating a delivery predictability metric from every node's history of encounters. MaxProp [9] assigns priority to both message delivery and message drops at the node by dividing the message buffer into two sections which are sorted by delivery probability and hop counts respectively to achieve a lower delivery latency. OLSR [10] selects a group neighbors one-hop away with bi-directional links as multi-point relays to minimize the flooding of control messages when updating route information and data messages trading off message overhead with delivery latency by sending fewer copies down potentially longer routes in the network. To improve the delivery probability, Weak State Routing [11] introduces per-node state by making nodes periodically announce their location in random directions resulting in aggregated ID-to-location mappings at receivers in the proximity of the broadcasting node. Messages are forwarded by source nodes in random direction so that they eventually come across a relay that maintains state about the destination node to re-direct them.

The opportunistic DTN routing protocols above employ multiple copies of data messages to cope with possible partitions in the network. BSDVR, however, aims to deliver a single copy of every data message, i.e., unicast delivery, even if the network is partitioned. To the best of our knowledge, this is the first time a DTN routing protocol can achieve unicast forwarding of data messages over a partitioned network.

III. BSDVR Overview

We design a new BSDVR protocol to obtain a better trade-off between control and data plane overheads in a dynamic network where partitions are prevalent. Traditional approach to DTN routing has been to use epidemic routing. In BSDVR, the key novelty is to achieve nearly *unicast forwarding* by using an enhanced version of the legacy DV routing design for control plane and optionally allowing conventional epidemic methods in the data plane. Following a traditional notion of link weight in routing, BSDVR assigns each link with a cost based on metrics such as geo-distance, capacity, or loss rate.

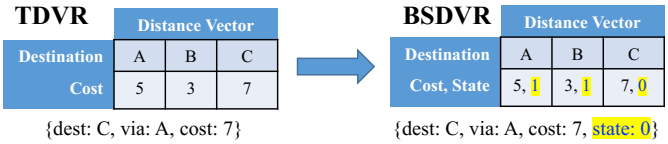
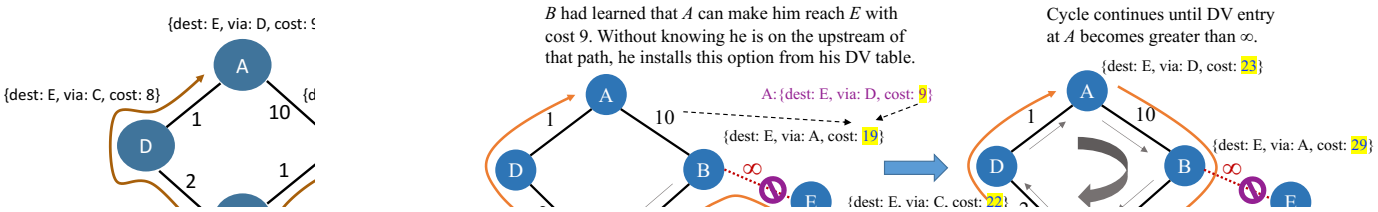


Fig. 1: TDVR vs. BSDVR

Control Plane: BSDVR includes additional state information in its DV table (DVT) entries as shown in Fig. 1. Each node can have either *active* or *inactive* DV entries, allowing them to preserve information about past connections along with the currently active connections. Active DV entries are those that comprise of present paths that are still active. Conversely, inactive DV entries are made up of paths that were active at some point in the past, but have become inactive. Both types of entries are used in constructing the forwarding table for the data plane. Given these binary state DVs, we carefully integrate the active and inactive states when DV updates are being exchanged. These binary states on paths provide nodes with more information in making data dissemination decisions. This binary separation allows BSDVR to detect link failures on the 'shortest path first (SPF) tree' used to install the paths in the forwarding table entries, and hence to treat connected paths separately than the failed or disconnected ones.

Data Plane: The binary state information of paths also gets used to create additional states for disseminating data packets stored in the message buffer of nodes. The data packets in the message buffer of a node can have three possible states: *not-forwarded*, *inactive-forwarded* and *active-forwarded*. *Not-forwarded* packets have not yet been forwarded to any of the other encountered nodes, e.g., because they did not have any path to the packet's destination. *Inactive-forwarded* packets are those that have passed onto a relay node that has an inactive path to the destination. Finally, *active-forwarded* packets are those that either got passed to their destination node directly or to a relay node that has an active path to the destination. Once a packet becomes *active-forwarded* by a node it never gets forwarded again. When the message buffer is full and old packets are removed in the order of *active-forwarded*, *inactive-forwarded*, and *not-forwarded*. This increases the packets' delivery likelihood. In contrast to traditional epidemic routing, BSDVR can attain more controlled and informed flooding of the data packets by using these three states.

BSDVR vs. Traditional DV Routing: In traditional DV routing (TDVR) based on Bellman-Ford's algorithm, it is not possible to entirely avoid the count-to-infinity since the nodes downstream to a failure cannot infer the emergence of a partition unless they are notified that the destination is unreachable. TDVR works with three local actions that guarantee global convergence: (1) Monitor and update a link table that keeps the costs of local links to neighbors, (2) Recompute DV (to all destinations in the network) as well as the forwarding table by using the Bellman-Ford procedure whenever there is a change in the link table or a DV from a neighbor is



Binary State DVRP – Key Design Components

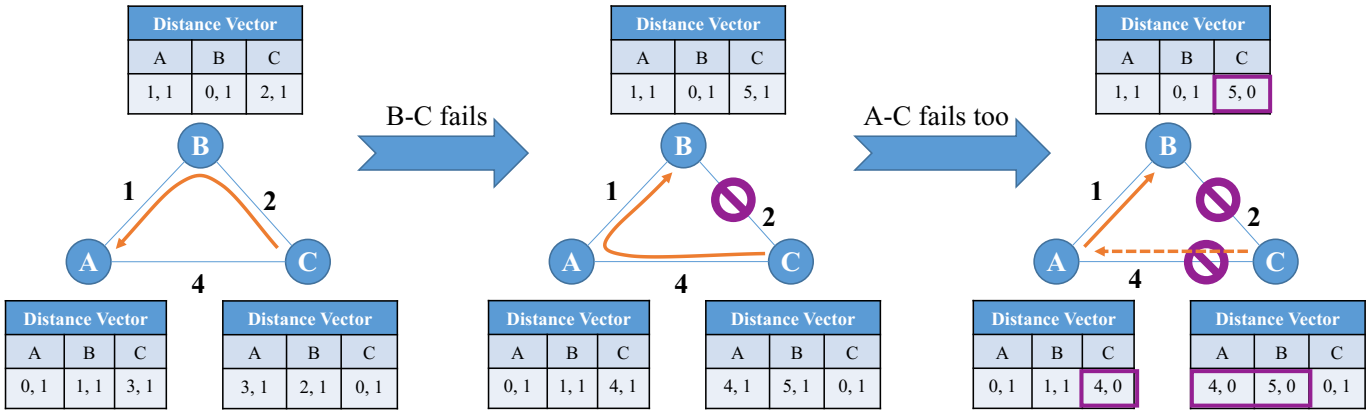


Fig. 4: Integration of state information in BSDVR

received in an update message, and (3) If there was a change in the forwarding table (e.g., a shorter or longer path is calculated for a destination), broadcast the new DV to all neighbors. Three typical optimizations to this procedure are broadcasting the updated part of DV instead of entire DV, not including the sender of the update message in the DV broadcast, and applying *poisoned reverse* to the DV entries by either removing them from the DV or setting them to *infinity*. In BSDVR, we implement these optimizations and assume them for the rest of the discussion.

Avoiding Count-to-Infinity: Consider the SPF tree[1] for destination E in Fig. 2. When the link $B-E$ fails, the network is partitioned and E is unreachable for the rest of the network as shown in Fig. 3. Since, prior to the failure, B had learned that A can make him reach E with cost 9, without knowing he is on the upstream of that path, B installs this option from his DVT and sends the "new" (but fake) path to C as an update message. Note that this effectively changes the SPF tree for E such that there is a loop in it, as shown in Fig. 3. Since C has no better option, it installs this new path from B and propagates it to D, which then to A. A further propagates it to B, thereby establishing a count-to-infinity cycle.

The fundamental reason why TDVR cannot avoid count-to-infinity when it faces partition-causing failures is that it does not separate paths to unreachable destinations from the paths to reachable destinations. In the example above, if the nodes below the failure were notified that the failure could potentially be making the destination unreachable, then precautions could be taken. BSDVR handles this by associating *active* or *inactive* state to each DV entry and explicitly marks the paths with reachability information, as shown in Fig. 1. Likewise, the DV updates being sent to the neighbors, which is part of the regular process in TDVR, also include this binary state information. In the example shown in Fig. 4, the link $B-C$ is disconnected, which causes new active shortest paths to be calculated. This is a failure that TDVR can handle. But, when the link $A-C$ also fails, a partition emerges; and this causes the paths from A and B towards C as well as from C to A and B to become inactive. In BSDVR, these inactive paths are explicitly acknowledged in the state fields of the corresponding DV entries.

IV. BSDVR Control Plane with Binary States

We now detail how the active and inactive states are handled in the control plane of BSDVR. In routing terminology, the shortest path to a destination is the *primary path* towards that destination and is typically illustrated as the upstream part of the SPF tree for that destination. We will use this terminology to discuss how the DV updates with binary state DV entries are used to handle link failures that may be causing partitions and then outline how to merge DV entries with different states.

[1]In DVR, destination i is reached via the SPF tree where i is the source. The SPF tree for each destination is conceptual and maintained across DVTs of nodes. The forwarding tables in each node only maintain one-hop forwarding decisions of these network-wide trees. The end-to-end shortest paths emerge from concatenation of destination-based forwarding along these trees.

After detecting a failure on primary path towards the source, erase all the DV entries in downstream nodes and "force" recalculation of new paths.

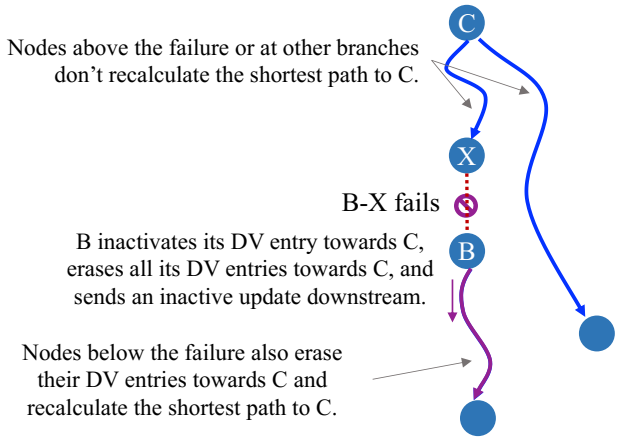


Fig. 5: Failure on Primary Path

A. Handling Link Failures

To avoid count-to-infinity cycles from partition-causing failures (as in Fig. 3), BSDVR detects failures on primary paths (by inspecting binary states of the DV entries) and introduces proactive messages to make sure any alternative active paths are disseminated to the nodes downstream of the failures.

1) Forced Recalculation on Primary Path: A key component BSDVR introduces is that if nodes realize that their primary path towards a destination has a failure, they erase all the DV entries for this destination and update other neighbors about the change in the path. Since the DV entries are marked with active or inactive states, the nodes can realize the failure on the primary path when they receive an inactive DV update from the next hop towards the destination, which is also the parent on the SPF tree for that destination. As seen in Fig. 5, this will cause all the nodes below the failure (on link $B-X$) to inactivate their existing path (i.e., mark the DV entry for the destination C as inactive) and erase all the alternative entries in their DV tables for the destination C. This inactivation of the primary path and erasure of alternative DV entries together guarantee that the nodes below the failure will avoid count-to-infinity if destination C has become unreachable for them, i.e., they are disconnected from the rest of the network.

2) Proactive Reply to Inactive DV Updates: Given that the nodes below the failure on primary path towards the destination are erasing all they know, how do the nodes below failure discover the availability of other paths to the destination via other branches of the SPF tree that are not affected by the failure? We introduce a proactive component to TDVR to solve this issue. In TDVR, a node does not send a DV update message to its neighbors unless there is a change in its forwarding table. We change this behavior so that a node replies to an inactive DV update (i.e., an update message that includes an inactive DV entry for a destination) from a neighbor, who is not on the primary path to the destination, by sending its current active DV entry for that destination.

In Fig. 6, consider the node E below the failure. It will send an inactive DV update to all its neighbors (Fig. 6(a)), including A that is residing on another branch of the SPF tree and has an active path to the destination C. When A receives the inactive update from E, it will prefer its existing path and will not make any changes to its existing DV. So, there will not be any change to the forwarding table of A as a result of the inactive update from E. In TDVR, A should not send any update as its shortest path has not changed. But, to make sure E is informed about the existence of the alternative path via A, we introduce new logic here as shown in Fig. 6(b) and

Binary State DVRP – Handling a Link Failu

How do the nodes below the failure learn about the alternative path

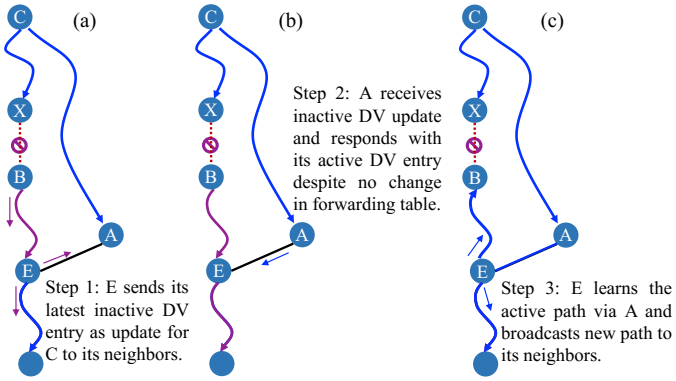


Fig. 6: Proactive response to inactive DV updates in BSDVR

3) Overhead in Handling Failures Not Causing a Partition: A careful inspection reveals that BSDVR introduces additional messages when handling a link failure that is not causing a partition. This is due to the fact that the nodes below a failure are forced to erase all the alternative path information while some of those alternative paths may still be valid. This is visible in Fig. 6. In TDVR, since E would not erase its alternative path via A when it received the inactive DV update from its primary path, it would immediately install this alternative path and the nodes below E would not even know about the failure. In BSDVR, however, we force E to erase what it knows for the possibility that E and its neighbors may be part of a partition caused by the failure. This causes E to send an inactive update to A as well as all the nodes below E. Then, after receiving the availability of an active path via A, E sends another DV update with this alternative active path. This process causes the nodes below E to receive two updates, both of which would be avoided in TDVR when handling the failure. We will later show that this is a good tradeoff since BSDVR avoids the huge cost of count-to-infinity in return of small extra overhead during failures not causing a partition.

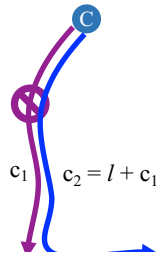
B. Merging Active and Inactive States

A major complication as well as novelty is how to calculate paths given two types of DV entries. Since we may have multiple paths available for reaching the same destination as active or inactive, there is a need to assign preference between

active Active

This happens when an active DV entry arrives in a DV update message for an existing inactive

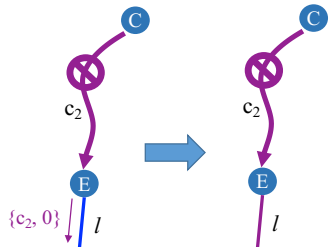
Approach: Choose the new active entry if the new entry cost is below infinity.



Current New

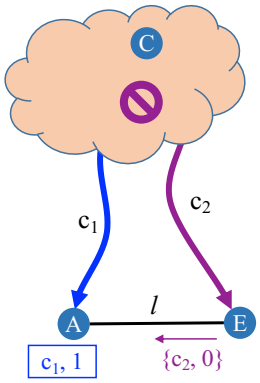
Active Inactive

- This happens when an inactive DV entry arrives in a DV



and Inactive States (cont'd

message for an existing active DV entr

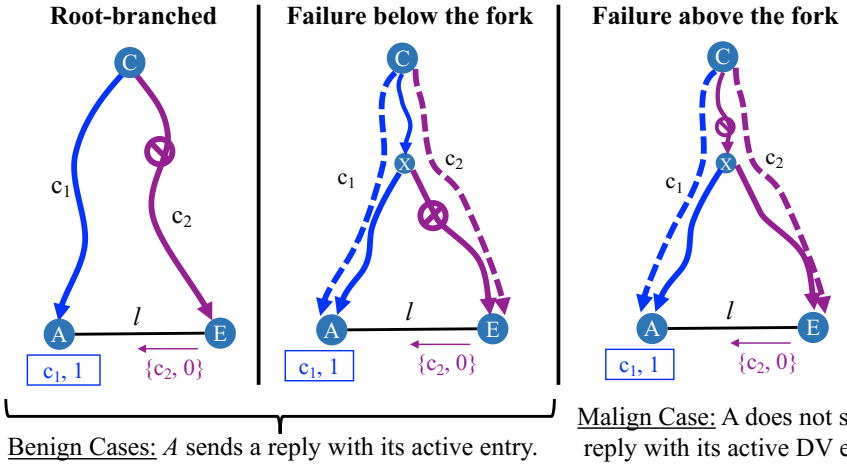


is not on the primary path to C. Should
th its active DV entry? The failure that trig
send an inactive entry may be upstream to

Binary State DVRP – Merging Active and Inactive States (co

Current New

Active Inactive



Benign Cases: A sends a reply with its active entry.

Malign Case: A does not send a reply with its active DV entry.

Ambiguous: A starts Pending Reply Timer and waits before sending a reply with its active DV entry.

Fig. 11: Inactive DV update $\{c_2, 0\}$ from a non-parent node

them. Table I summarizes BSDVR's control logic in handling different DV entries when an update message is received. If both the incoming and existing DV entries are the same type, BSDVR uses Bellman-Ford's algorithm to calculate minimum cost paths. However, there are two new cases to consider: 1) An active DV entry arrives while there is an inactive entry for the same destination. 2) An inactive DV entry arrives while there is an active entry for the same destination. Next, we detail how BSDVR handles these cases by using the following notation based on the scenario in Fig. 7:

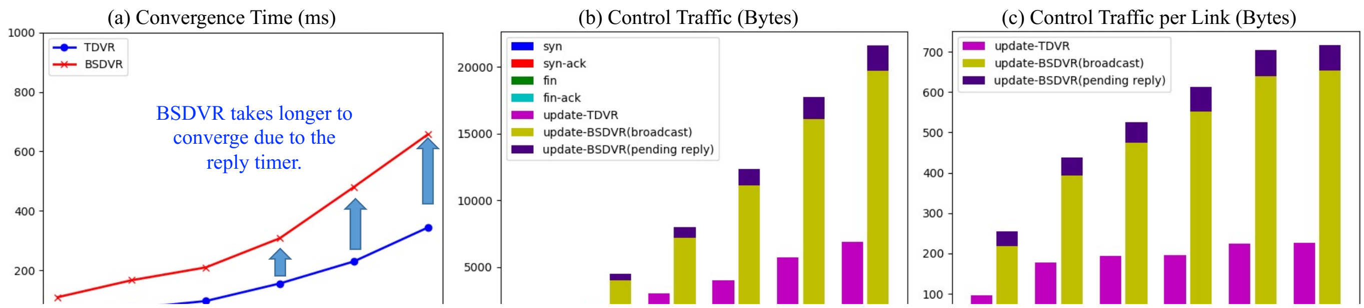
- C is the destination,
- E is the sender of the DV update,
- A is the recipient of the inactive DV update,
- l is the cost of the link $A - E$,
- c_1 is the length of the shortest path $A \rightarrow C$, and
- c_2 is the length of the shortest path $E \rightarrow C$.

1) Active DV Update for an Inactive DV Entry: When an active DV update arrives while current DV entry for the same destination is inactive, BSDVR chooses the new active entry if the new entry cost is below infinity. This is shown in Fig. 7. E sends an active DV update $\{c_2, 1\}$ informing A about the existence of an active path towards the destination C with a finite distance c_2. Since A's existing DV entry $[c_1, 0]$ for destination C is inactive, it is best for A to install the new path with the cost $l + c_2$ regardless of how c_1 compares to this active path's length. This is simply because of the fact that an active path is reachable while an inactive path is unreachable, and hence an active path is preferable. An exceptional case is that the DV update may be poisoned, which will present an active path with infinite cost as illustrated in Fig. 8. This happens when E has been using A to reach C and A has not informed E about the failure yet. Observing the ∞ cost value in the DV update, A should not accept the active path via E as that would constitute a loop.

Current	New	Rule
Active	Active	Choose min-cost DV entry
Inactive	Inactive	Choose min-cost DV entry
Inactive	Active	Choose the active entry iff its cost is finite
Active	Inactive	**IF** {cost of the active entry is infinity} Choose the inactive entry **ELSE** **IF** {the inactive entry is on primary path} Choose the inactive entry Remove alternative DVT entries **ELSE** Reply to the inactive update

TABLE I: Merging Current and New DV Entries

Emulation Results: Single Link Failures



Emulation Results: Partitions

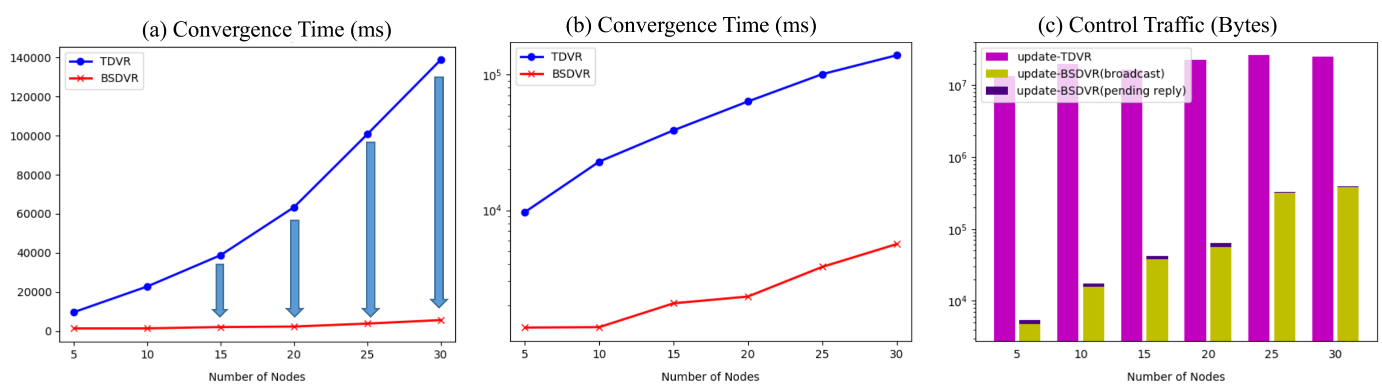


Fig. 13: Results from experiments with partitions.

2) Inactive DV Update for an Active DV Entry: Even though it may seem that there should be no action to be taken when an inactive DV update comes for an active DV entry, there are two critical cases for BSDVR to handle: 1) This inactive DV update may be an indication of a failure on the primary path. 2) The inactive update is coming from a neighbor that just inactivated its existing path (due to an inactive update from its primary path) and it needs to be informed about the existence of the alternate active path. We now detail BSDVR's approach in these cases:

Failure on Primary Path: An inactive DV update may come from the parent due to a failure upstream to the parent. In Fig. 9, E inactivated its DV entry for destination C due to a failure on its upstream path towards C and needs to let its neighbors know about this. Thus, it sends an inactive DV update to A. At A, this presents itself as a situation where the parent node on the primary path towards the destination C is reporting that its path to C is now inactive. Since this means that the upstream is now failed, A accepts this update and erases all its alternative DV entries for C and sends an inactive update to its other neighbors. As explained in Sec. IV-A1, this is an intentional inactivation embedded in the protocol so that all the nodes below a failure are forced to recalculate their paths.

Proactive Reply: When an inactive update comes from a non-parent neighbor, this means the sender of the update does not know about the existing active path. From A's perspective, E is a neighbor that does not have an active path to C while A itself has one. So, A should proactively inform E about the existence of an active path to C so that E can also utilize this path. However, as illustrated in Fig. 10, the complication here is that the failure that triggered E to send an inactive entry may be upstream to A too. This situation can happen when failure information propagates to E before A. If that is the case, A should not reply and wait until it is sure that its active entry $[c_1, 1]$ is still valid. These possibilities must be differentiated in order for A to decide when to proactive reply by sending an active DV update to E. We categorize the possibilities into three cases as shown in Fig. 11: First, E could be on an entirely different branch from the root of the SPF tree. We call this case as the "Root-branched" case, and in this case, A should reply to E with its active DV entry for destination C. It is also possible that the paths from A to C and E to C overlap upstream and there is a fork at a node X. This fork case can be good or bad. If the failure is below the fork node X, then A should reply with its active entry. However, if the failure is above the fork, then A should wait until it gets notified that its primary path to C has become inactive. At A, it may be possible to distinguish these cases and identify the benign and malign ones. But, if it is ambiguous, A starts a *Pending Reply Timer* and waits before sending a reply to E. We rely on this timer for resolving ambiguous cases of sending

proactive replies. In the current version, we judge that there is no possibility of a fork node X (which means it has to be a benign case) if (1) A's parent is C or (2) the cost of reaching E from A via A's parent is equal to $c_1 + c_2$. Otherwise, A starts the timer due to doubt on the presence of fork node X.

V. Evaluation & Analysis

We implemented BSDVR[2] using multiple processes, which emulates a routing environment among mobile devices where processes open or close TCP-like connections with each other using Bluetooth or Wi-Fi interfaces. We were able to emulate up to 30 nodes placed randomly in a topology with 3 to 5 average node degree. We set the *Pending Reply Timer* of BSDVR to 100 ms and compared it against the TDVR in terms of convergence time and messaging overhead. In TDVR, we represented the infinity with the maximum integer value. We repeated the experiments 3 times on different topologies.

Single Link Failures: We failed the links in the network one by one. Since the topologies had a minimum degree of 2, this meant that there was no partition due to the link failures. In terms of convergence time, we observe in Fig. 12(a) that BSDVR performs worse than TDVR in handling failures that do not cause partitions. This is due to the fact that there is a fixed *Pending Reply Timer* that sometimes cause the nodes to wait unnecessarily before sending an active path to its neighbor. This is also visible on the control traffic, in Fig. 12(b), as BSDVR generates more DV updates (Sec. IV-A3). Even though the number of replies is few, unnecessary DV updates cause notable additional control traffic in BSDVR. The good news is that BSDVR's control traffic overhead does saturate with respect to the network's capacity as seen in Fig. 12(c). The takeaway is that optimizing the *Pending Reply Timer* or eliminating the need for the timer is crucial for BSDVR's performance over a non-partitioned network.

Partition-Causing Failures: The real benefit of BSDVR is in handling the failures that cause partitions. To test this, in the same topologies as before, we failed the links of a node simultaneously and caused a partition in the network. We then measured the convergence time and control traffic in the rest of the network without the disconnected node. We repeated this for every node in the network. As shown in Fig. 13(a), TDVR experiences count-to-infinity and takes much longer time to converge, while BSDVR's convergence time is virtually unaffected by the partition in the network. This is more clear in the logarithmic scale plot in Fig. 13(b). We see that BSDVR's convergence time increases only linearly as the network grows. In terms of the control traffic, in Fig. 13(c), we see an order of magnitude benefit for BSDVR as it prevents count-to-infinity during partition-causing failures.

VI. Summary and Future Work

We introduced BSDVR protocol that uses binary state information to facilitate a hybrid between traditional DV and epidemic routing protocols. BSDVR's key goal is to achieve unicast-like routing that can work on DTNs where partitions are prevalent and link capacities are too limited for carrying multiple copies of data messages. We compared BSDVR against the traditional DV routing protocol in a controlled network. Our emulation experiments showed that BSDVR generates more control overhead during link failures that do not cause partitions; however, this control overhead saturates with respect to the network's capacity. In terms of failures causing network partitions, BSDVR control overhead has an order of magnitude benefit over traditional DV routing, leading to significantly better convergence times.

BSDVR offers several interesting future research directions. Legacy routing protocols essentially ignore the paths that are disconnected, and hence delete the state left from those disconnected paths. BSDVR opens a new dimension by showing, in the case of shortest-path routing, how the state information of a disconnected path can be integrated with the state information of connected paths. To this end, it is worthy to revisit distance vector routing protocols that run in wireline networks and see if the concept of binary state can help.

Data plane performance of BSDVR needs to be studied. In terms of its control plane design, BSDVR can use further optimizations of its *Pending Reply Timer*. Also, when differentiating the benign and malign cases during the reception of an inactive DV update from a non-parent node, it is possible that A can use more sophisticated logic to further avoid the reply timer. Finally, BSDVR's theoretical analysis of worst, average, and best case scenarios in terms of convergence time, messaging overhead, and state complexity at routers.

[2]Please refer to https://tinyurl.com/3w59tr45 for code and detailed explanation of the implementation.

References

[1] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni, and D. Turgut, "Routing protocols in ad hoc networks: A survey," *Computer networks*, vol. 55, no. 13, pp. 3032–3080, 2011.

[2] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: a distance-vector multipath routing protocol," in *Proceedings of IEEE INFOCOM*, vol. 1, 2001, pp. 557–564.

[3] C. Perkins, E. Belding-Royer, and S. Das, "RFC3561: ad hoc on-demand distance vector (AODV) routing," 2003.

[4] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *ACM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[5] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: Positive and negative social effects," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 387–401, 2012.

[6] M. J. Khabbaz, C. M. Assi, and W. F. Fawaz, "Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges," *IEEE Comms. Sur. & Tut.*, vol. 14, no. 2, pp. 607–640, 2011.

[7] L. Zhao, F. Li, C. Zhang, and Y. Wang, "Routing with multi-level social groups in mobile opportunistic networks," in *Proceedings of IEEE GLOBECOM*, 2012, pp. 5290–5295.

[8] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19–20, 2003.

[9] J. Burgess, B. Gallagher, D. D. Jensen, B. N. Levine *et al.*, "MaxProp: routing for vehicle-based disruption-tolerant networks." in *Proceedings of IEEE INFOCOM*, vol. 6. Barcelona, Spain, 2006.

[10] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "Rfc7181: The optimized link state routing protocol version 2," 20145.

[11] U. G. Acer, S. Kalyanaraman, and A. A. Abouzeid, "Weak state routing for large-scale dynamic networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1450–1463, 2010.