R3: Robust Replication Routing in Wireless Networks with Diverse Connectivity Characteristics

Xiaozheng Tie, Arun Venkataramani
University of Massachusetts Amherst
{xztie, arun}@cs.umass.edu

Aruna Balasubramanian
University of Washington
arunab@cs.washington.edu

ABSTRACT

Our work is motivated by a simple question: can we design a simple routing protocol that ensures robust performance across networks with diverse connectivity characteristics such as meshes, MANETs, and DTNs? We identify packet replication as a key structural difference between protocols designed for opposite ends of the connectivity spectrum—DTNs and meshes. We develop a model to quantify under what conditions and by how much replication improves packet delays, and use these insights to drive the design of R3, a routing protocol that self-adapts replication to the extent of uncertainty in network path delays. We implement and deploy R3 on a mesh testbed and a DTN testbed. To the best of our knowledge, R3 is the first routing protocol to be deployed and evaluated on both a DTN testbed and a mesh testbed. We evaluate its performance through deployment, trace-driven simulations, and emulation experiments. Our results show that R3 achieves significantly better delay and goodput over existing protocols in a variety of network connectivity and load conditions.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Store and forward networks, Wireless communication

General Terms

Design, Experimentation, Performance

Keywords

Wireless routing, forwarding, replication

1. INTRODUCTION

Routing in wireless networks has seen a huge body of work over the last decade as networking researchers have been identifying diverse target environments such as mesh networks, mobile or vehicular ad hoc networks (MANETs), and disruption-tolerant networks (DTNs). A variety of current or foreseeable applications motivate this research such as extending the reach of the Internet [31, 2, 17], wildlife monitoring [24, 5], content distribution [20], tactical operations, etc. In response, researchers have developed a number of routing protocols tailored for specific points such as meshes, MANETs, or DTNs along a diverse connectivity spectrum.

Although much research has been expended on specific points along the connectivity spectrum, little attention has been devoted to designing a protocol that is robust to diverse connectivity characteristics. Indeed, state-of-the-art routing protocols designed for one environment either work poorly or break down completely in others. For example, mesh protocols based on traditional link-state or distance vector routing break down in DTNs where a contemporaneous end-to-end path is unavailable. Most proactive as well as on-demand MANET protocols also assume the availability of a contemporaneous end-to-end path. Likewise, DTN protocols commonly use packet replication to reduce delays, but packet replication performs poorly in predictable, well connected mesh networks. We experimentally show (§2) that the performance penalty of fielding state-of-the-art protocols (or straightforward optimizations thereof) outside of their target environments can be severe.

Our work is motivated both by the diversity of existing network environments and the compartmentalization of existing routing protocols. We start with the following question: *can we design a wireless routing protocol that ensures robust performance across diverse and varying connectivity characteristics all the way from well-connected mesh networks to always-partitioned DTNs?* The pursuit of this question is rooted in concerns more practical than just the intellectual challenge it offers. As wireless networks proliferate, users are likely to encounter environments with increasingly varying connectivity. We show in §2 that there are already several examples of real-world network deployments that exhibit diverse connectivity characteristics, e.g., *spatially-varying* connectivity as in a WiFi mesh interconnected with a vehicular DTN or *temporally-varying* connectivity [13] where the underlying topology sometimes resembles a MANET and at other times a DTN. A self-adapting protocol can significantly improve performance compared to protocols designed with specific assumptions about connectivity characteristics.

To design a self-adapting protocol, we look at routing protocols designed for diametrically opposite ends of the connectivity spectrum—well-connected mesh networks and always-partitioned DTNs—and observe that a critical structural difference between the two is packet replication, a mechanism that yields significant delay benefits for the latter but yields little benefit for and often hurts the former. We develop a simple model to quantify under what conditions and by how much replication improves packet delays. Although the simple model ignores effects of load and interference, it enables us to formally show that replication yields significant delay gains if and only if path delays exhibit high *unpredictability* without making any other assumptions about the underlying delay distribution.

Based on this model's insights, we design and implement R3, a routing protocol that self-adapts replication to the unpredictability of path delays as well as the load in the network. R3 achieves these properties using the following key insights. First, as per the model, it monitors the distribution of path delays unlike traditional DTN routing protocols that simply monitor the expected delay. Second, it leverages the empirical finding that replicating each packet along a small number of paths suffices to capture most of the achievable replication gain. Third, R3 uses load-aware replication that allows it to be responsive to changing load conditions. R3 turns off replication and switches to single-path forwarding when it determines that the actual delay is significantly higher than the estimated delay.

We implemented and deployed R3 on a 16-node mesh testbed and a 20-node vehicular DTN testbed. We extensively evaluate R3 using (1) our prototype deployment, (2) trace-driven simulation experiments based on a hybrid mesh-DTN testbed, DieselNet-Hybrid, and a MANET testbed [13], and (3) emulation experiments over a mesh testbed designed to emulate network topologies with varying levels of connectivity all the way from well-connected meshes to highly disconnected DTNs. We validate our simulator using results from our deployment experiment. Our key results are as follows: (1) R3 achieves up to $2\times$ better delay and $1.3\times$ better goodput compared to a naive "multi-configuration" protocol that simultaneously runs a mesh and DTN routing protocol and dynamically selects one of the two routes depending on the connectivity. (2) When the connectivity is fixed at the well-connected (or sparsely-connected) extreme, R3 achieves delay and goodput comparable to the state-of-the-art protocols tailored for that extreme. (3) In sparsely-connected networks under high load, R3 improves delay and goodput by up to $1.4\times$ and $1.3\times$ respectively compared to RAPID, a state-of-the-art DTN routing protocol, i.e., R3 at the least is a better DTN routing protocol. We also show that the overhead of R3 is only 0.5% of the total data transfered.

More broadly, although we focus on quantitative performance metrics in this paper, our position is that a common self-adapting routing protocol merits investigation for reasons beyond just performance. First, it can simplify the interconnection of diverse wireless networks and thereby spur longer-term growth and innovation. On the other hand, having different protocols for different environments increases the engineering complexity of maintaining multiple code-bases and the management complexity of tuning them for different environments. Second, it enables separation of concerns, e.g., a duty-cycling algorithm for a sensor network can be designed without worrying about whether or to what extent the underlying topology remains connected and how it might affect routing performance. A first step towards quantifying the strength of these arguments is to go through the exercise of designing and implementing a self-adapting routing protocol—a key contribution of this paper.

2. WHY DESIGN FOR DIVERSITY?

In this section, we experimentally make the case for a self-adapting routing protocol for networks with diverse connectivity characteristics. First, we show that state-of-the-art routing protocols perform poorly outside the specific environment for which they are designed. Second, we present real-world examples of networks exhibiting significant temporal and spatial diversity in connectivity characteristics.

2.1 Performance outside target environments

State-of-the-art wireless routing protocol designs deeply embed assumptions about the connectivity characteristics of the underlying network. For example, in sparsely-connected DTNs, packets are routed by *replicating* copies through multiple nodes. In contrast, in well-connected mesh networks, packets are *forwarded* over a single path to the destination.

To understand how protocols perform outside their target environment, we conduct a simple experiment comparing the performance of several DTN, MANET and mesh routing protocols, namely, RAPID [8], Random replication, DTLSR [16], AODV [29] and OLSR [3]. The first two are replication protocols for sparsely-connected networks, and the latter three are forwarding protocols for intermittently-disconnected or well-connected networks. We conduct trace-driven experiments based on a sparsely-connected testbed DieselNet-DTN and deployment-based experiments on a well-connected mesh testbed (Figure 8).

We make simple modifications to the above protocols so that DTN protocols can work in a mesh and vice-versa. We choose workload parameters to focus specifically on low and high network load. A more detailed description of the modifications as well as the experimental setup is deferred to §6, which also presents a more exhaustive exploration of protocol, workload, and environment parameters.

Figure 1(a) shows that in a well-connected mesh, replication using RAPID increases delay by about $2\times$ compared to OLSR, the best forwarding protocol. Replication wastes resources and yields little benefit in well-connected networks.

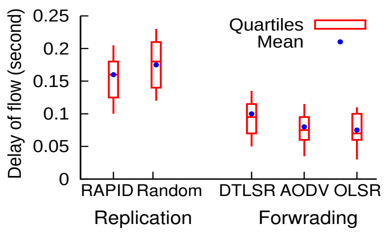
However, the situation reverses in sparse networks. Figure 1(b) shows that in the DTN environment DieselNet-DTN, replication routing using RAPID yields almost a $2\times$ reduction in delay compared to DTLSR, the best forwarding protocol. Even random replication significantly reduces delay compared to forwarding protocols.

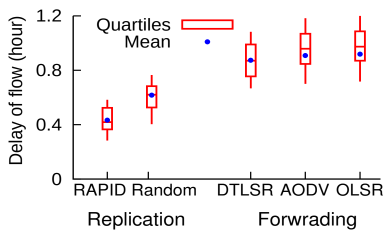
Replication is not always beneficial in sparse networks. Figure 1(c) shows that replication can hurt performance in sparse networks when the offered load is high. Under high load, replication increases the delay in DieselNet-DTN by 15% over forwarding. We observe similar trends for goodput across different networks and loads (not shown in figure).

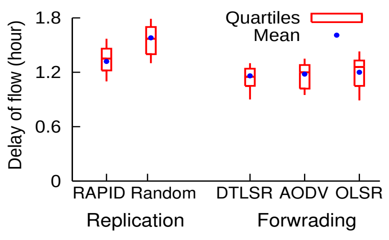
Taken together, these results suggest that existing protocols work poorly outside of the specific environment for which they are designed. This state of affairs would not be terribly disturbing if real networks exhibited stable connectivity characteristics, i.e., a given network always either looked like a DTN or like a mesh. However, we find significant temporal and spatial diversity in realistic network testbeds, as described next.

2.2 Temporal and spatial connectivity diversity in real-world networks

Figure 2(a) shows that the *connectivity*, i.e., fraction of connected node-pairs in the Haggle network varies *temporally*. Haggle [13] is an opportunistic network formed by 8

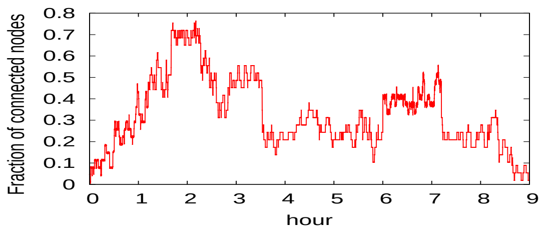




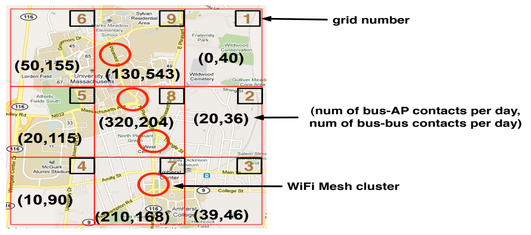


(a) Mesh: low load (1 pkt/second/flow) (b) DieselNet-DTN: low load (20 pkt/hour/flow) (c) DieselNet-DTN: high load (50 pkt/hour/flow)

Figure 1: Each boxplot shows min, max, 25%, 75% quartiles, median, and mean delays. All experiments use 30 concurrent flows. The delay of a flow is the average delay of all packets in the flow. Replication benefits significantly in the DieselNet-DTN network under low load but hurts performance in well-connected mesh. Replication hurts performance under high load in the DieselNet-DTN network.



(a) Haggle network: Connectivity varies temporally.



(b) DieselNet-Hybrid : Connectivity varies spatially. The region includes UMASS campus and Amherst town center (4 sq.mile)

Figure 2: Networks that exhibit varying connectivity characteristics temporally and spatially

mobile devices carried by users and one stationary device in the Intel Cambridge Lab. Figure 2(a) shows that the connectivity of the Haggle network changes dynamically between 10% and 80% as a result of user mobility. Connectivity can also vary temporally due to other causes such as node failures or duty cycling in sensor networks.

Similarly, Figure 2(b) shows that the DieselNet-Hybrid bus network's connectivity varies *spatially.* DieselNet-Hybrid is a hybrid mesh-DTN testbed consisting of 20 buses and a number of open access points (APs) in a 4 sq.mile area. In some areas, several APs cluster together forming a mesh network, as shown in the figure. Buses are connected either when they come in contact with the APs or other buses. Figure 2(b) shows that the connectivity of buses varies with geographical location. We divide the whole region into 9 grids and number them in increasing order of the total number of bus-AP and bus-bus contacts per day. For example, Figure 2(b) shows that the total number of contacts is over 500 in grid 8 and 9, but is below 100 in grids 1, 2, and 3.

Changes in network connectivity across location is often the result of the difference in wireless penetration. For example, in the hybrid mesh-DTN topology of DieselNet-Hybrid, buses are well connected in urban centers with high WiFi AP density (as shown by the mesh clusters in Figure 2(b)), but are poorly connected as they move to less urban areas. Recent measurement studies show that such variations in connectivity can occur even in cellular networks [10].

Mesh protocols today rarely utilize the connectivity available in DTN areas with poor WiFi penetration. However, exploiting hybrid mesh-DTN networks and utilizing the connectivity offered by disruption-prone DTNs when available (e.g., bus-bus contacts in DieselNet-Hybrid) can significantly increase wireless capacity. For example, Balasubramanian et al [9] show that the performance of delay-tolerant Web applications can be improved using a sparse-DTN when available. Similarly, Hui et al [23] show that delay-tolerant and opportunistic communications can double the throughput of a well-connected mesh network. Thus, in order to fully exploit the potential of hybrid networks, protocols should self-adapt to changing connectivity characteristics.

3. QUANTIFYING REPLICATION GAIN

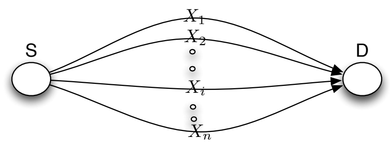
To address the challenge of designing a self-adapting routing protocol, we begin by identifying *packet replication* as a key structural difference between routing protocols designed for well-connected and sparsely-connected networks. Thus, it is important to understand under what network and load conditions should replication be used over forwarding.

To this end, we present a simple analytical model to first understand the benefits of replication in terms of the network characteristics, ignoring the impact of load and interference. The model shows formally that replication yields high delay gains if and only if network path delays are highly unpredictable. Although the model makes several simplifying independence assumptions, it yields insights into the best-case gain of replication. In the next section, we design R3 so as to adapt replication based on measured deviations in the observed delays from those predicted by the model, thereby accounting for load or interference effects.

3.1 Model

We quantify the benefit of replication in terms of end-to-end delay improvement. To this end, we model the end-to-end delays of the different paths connecting a source and destination pair. Figure 3 shows a node pair with source S, destination D, and n different (possibly multi-hop) paths connecting them. The end-to-end delays of the paths are represented by random variables $X_1, \cdots, X_n$ respectively.

We make a simplifying assumption that the random variables $X_1, \cdots, X_n$ are independent. Note that this assumption is rather simplistic as it implies the paths are fully disjoint and there is no interference between packets traversing different paths and thus ignores any effects induced by load.



There are n paths between S and D. Each path i has a delay X_i

Figure 3: Model

Forwarding and replication choose from among these paths for routing, but in different ways. We assume forwarding sends a packet on the path with the minimum expected delay, and replication sends copies of a packet on all paths. Let μ and $\mu_{(1)}$ denote the expected delay from S to D using forwarding and replication respectively, then

$$\mu = \min\{E[X_1], E[X_2], ..., E[X_n]\} \tag{1}$$

$$\mu_{(1)} = E[\min\{X_1, X_2, ..., X_n\}] \tag{2}$$

The random variable representing the delay when using replication, $\min\{X_1, X_2, ..., X_n\}$, is also commonly referred to as the first-order statistic. It is well known [14] and easy to show that $\mu \geq \mu_{(1)}$, and we define the ratio $\mu/\mu_{(1)}$ to be the *replication gain.*

3.2 How to compute the replication gain?

The replication gain is defined as $\mu/\mu_{(1)}$. Let m be the path with minimum expected delay. It is straightforward to show (see [35]) that the replication gain is

$$\frac{\mu}{\mu_{(1)}} = \frac{\displaystyle\int_0^{+\infty} P[X_m > x]dx}{\displaystyle\int_0^{+\infty} \prod_{i=1}^{n} P[X_i > x]dx} \tag{3}$$

Eq. 3 presents an important insights: Computing the replication gain, and in turn deciding whether to replicate, requires knowledge of the delay distribution of the different paths (X_i's), not just their expected values ($E[X_i]'s$).

State-of-the-art replication protocols often make replication decisions based on the expected delay[34, 32] or make assumptions about the delay distribution *a priori* [8, 34, 37, 32, 7]. For example, RAPID [8] assumes that the delay distribution is exponential, and therefore estimates the replication gain of k replicas to be k-fold relying only on expected delays. However, the k-fold replication gain assumption does not hold when the delay distribution is not exponential, e.g., in predictable mesh networks, two replicas are unlikely to halve the delay. Therefore, we design R3 (§4), which makes replication decisions based on the delay distribution, but makes no assumptions about the distribution *a priori*. Our evaluations (§6) shows that using the delay distribution can significantly improve performance of routing protocols over only using the expected values.

3.3 When is the replication gain high?

Intuitively, replication helps when the delays are highly unpredictable. For example, consider a source and destination connected by two paths whose delays are given by random variables X_1 and X_2. If X_1 is always equal to 1 second and X_2 is always equal to 3 seconds, then replication yields no benefit compared to simply choosing the first path. Now suppose that X_1 is 0.1s in 90% of the cases and 10s in 10% of the cases, and X_2 is 0.3s in 90% of the cases and 30s in 10% of the cases. The mean delays of X_1 and X_2 are still about 1s and 3s respectively. However, replication

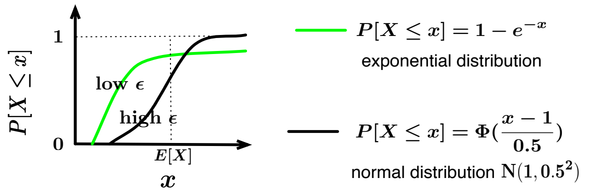


Figure 4: Distributions with more values below the mean have lower predictability

results in a mean delay of about 0.2s, a 5× improvement compared to the shortest path forwarding. We formally define *predictability* of delays as follows.

Definition 1. The predictability of a random variable X is the smallest ϵ such that its cumulative density below $\epsilon E[X]$ is at least $1 - \epsilon$, i.e., $P[X \leq \epsilon E[X]] \geq 1 - \epsilon$.

Note that $\epsilon = 1$ always satisfies the condition in the definition (as $P[X \leq E[X]] \geq 0$), so predictability is well defined and lies between 0 and 1. Low predictability (i.e., highly unpredictable delays) means that the delay is much smaller than the mean most of the time, but the mean is inflated by occasional large values. Figure 4 shows a pictorial example of distributions with low and high predictability.

In the previous example, when X_1 is always 1 second, $\epsilon = 1$. When X_1 is 0.1s in 90% of the cases and 10s in 10%, the predictability $\epsilon = 0.1$. We arrive at these predictability values by solving the equation $P[X_1 \leq \epsilon E[X_1]] \geq 1-\epsilon$. Intuitively, as in the example, lower predictability implies higher replication gain.We formalize this claim using the following theorem.

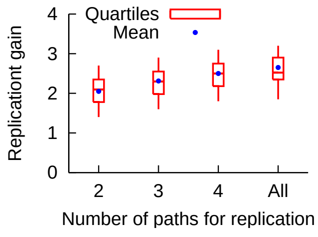
THEOREM 1. *For a source-destination pair connected by $n \geq 2$ paths, the delays of the paths are independent and identically distributed (i.i.d) as denoted by X.*
(a) If X has predictability ϵ, then the replication gain is at least $\frac{1}{1-(1-\epsilon)^2}$, i.e., low predictability implies high replication gain.
(b) If the replication gain is $G \geq 1$, then X has predictability at most $G^{-\frac{1}{n+1}}$, i.e., high replication gain implies low predictability.

The relation between replication gain and predictability generalizes to independent but nonidentical distributed random variables. We define *relative predictability* as:

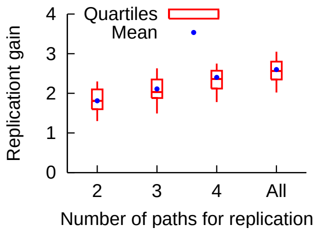
Definition 2. For a set of random variables $X_1, \cdots, X_n$, let X_m have the minimum expected delay. Then the relative predictability of a random variable X_i is the smallest δ such that $P[X \leq \delta E[X_m]] \geq 1 - \delta$.

THEOREM 2. *For a source-destination pair connected by $n \geq 2$ paths, the delays of the paths are independently but non-identically distributed as denoted by $X_1, \cdots, X_n$ and $E[X_m] = \min\{E[X_1], \cdots, E[X_n]\}$*
(a) If there exists a variable $X_i \neq X_m$ with relative predictability δ, then the replication gain is at least $\frac{1}{1-(1-\delta)^2}$, i.e., low predictability implies high replication gain.
(b) If the replication gain is $G \geq 1$, and there exists X_i such that $P[X_i \leq G^{-\frac{1}{n+1}} E[X_m]] = \max_{1 \leq j \leq n} P[X_j \leq G^{-\frac{1}{n+1}} E[X_m]]$, then X_i has relative predictability at most $(G^{-\frac{1}{n+1}})$ i.e., high replication gain implies low predictability.

The proofs for both theorems are presented in [35].



(a) DieselNet-DTN



(b) Haggle

Figure 5: Replication gain across node pairs. Two paths get most of the gain on both traces.

3.4 Implications

The theorems yield an important implication: replication gain grows unbounded as the predictability approaches 0. Furthermore, as the theorems hold for any $n \geq 2$, even two paths can yield unbounded gains. To study how replication gain increases with the number of paths used for replication in practice, we perform trace-driven analysis on DieselNet-DTN and Haggle. The traces log inter-contact times between each pair of nodes (details are deferred to §6), which yields a link-state graph with each link annotated with an empirically measured distribution of delay values. The individual link delay distributions are used to compute the end-to-end delay distribution of a path. We use these distributions to estimate the replication gain using Eq. 3 when two, three, four and all paths are used for replication respectively. Figure 5 shows that two paths give between 65% and 75% of maximum gain in real traces, and the marginal benefit of using additional paths is small.

4. DESIGN AND IMPLEMENTATION

In this section, we present R3, a routing protocol that self-adapts the level of replication to network and load conditions. R3's design is driven by the following insights revealed by our model (§3) and analysis of existing routing protocols (§2): (1) replication gain depends on the distribution of path delays, not just the mean value; (2) two paths suffice to capture much of the replication gain; (3) under high load, replication can hurt performance.

R3 consists of three main components—*(i)* estimating the end-to-end path delay distribution, *(ii)* selecting the best replication path based on the distributions, *(iii)* reigning in replication according to load.

4.1 Delay estimation

The model presented in §3 quantifies the replication gain as a function of the end-to-end delay distribution. However, estimating the delay distribution in a unified manner across diverse networks is non-trivial. For example, in well-connected mesh networks, ETT [19], Per-hop RTT [6] or similar metrics are used to estimate link delays, but these metrics are unsuitable to estimate delays in disruption-prone environments where there are no persistent links. Similarly, in sparse networks, inter-contact times [8, 7, 32] are often used to measure link delays, but they are not meaningful in mesh environments where links are persistent. Instead, R3 uses a unified metric to capture link delays in both well-connected and sparse networks.

4.1.1 Link delay metric

We define link delay as the sum of the *link availability delay* and the *delay to successfully transfer the packet* across the link. In sparse-networks, the availability delay contributes significantly to the link delay, while in well-connected networks, the delay to successfully transfer the packet is the significant contributor to the link delay. In this subsection and the next, we ignore load-dependent queuing delays and address them in §4.3.

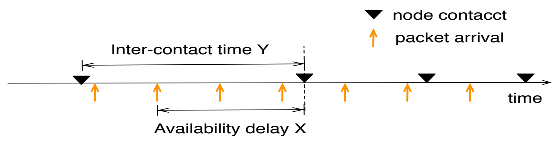


Figure 6: Availability delay X and inter-contact time Y

Link availability delay: This delay is the time until which the link remains down. In mesh networks, this time is zero for nodes that are connected. In disruption-prone networks, this time is often approximated by the expected inter-contact time between the corresponding nodes. However, there is a subtle but important distinction between the availability delay and the inter-contact time.

Let X denote the availability delay and Y denote the inter-contact time between two uncorrelated nodes (Figure 6). Then, as per [26]:

$$P[X \leq x] = \frac{1}{E[Y]} \int_0^x (1 - P[Y \leq y])dy \qquad (4)$$

and

$$E[X] = \frac{E[Y]}{2} + \frac{\sigma^2(Y)}{2E[Y]} \qquad (5)$$

where $\sigma^2(Y)$ denotes the variance of Y.

In general, $E[X]$ is not equal to $E[Y]$, contrary to explicit or implicit assumptions made by prior routing protocols [8, 33, 7, 32]. For example, if Y represents periodic node meetings, i.e., the variance is 0, then X is uniformly distributed between 0 and $E[Y]$, so $E[X] = E[Y]/2$. If Y is uniformly distributed in the interval $[a, b]$, then $E[Y] = (a + b)/2$ and $E[X] = a/6 + b/3$. To enable delay estimation for arbitrary distributions, R3 explicitly measures the availability delay distribution X.

Delay to successfully transfer the packet: This delay depends on the transmission delay, propagation delay, and the loss rate of the link. To incorporate loss rates, the delay to transfer packets includes the delay incurred in retransmitting lost packets.

4.1.2 Estimating link delay

The total link delay is measured using link probes. Each node broadcasts probes every second and one-hop neighbors who receive the probe immediately send an acknowledgement. If a probe is acknowledged, the sender estimates the link delay as half of the corresponding round-trip time. If a probe is not acknowledged, the sender estimates the corresponding round-trip time as the time since sending the unacknowledged probe and receiving an acknowledgement (for a subsequent probe). Thus, the delays for acknowledged probes incorporate the transmission and propagation delays, and the delays for unacknowledged probes incorporate the delay introduced by unavailability and loss.

Each node computes a summary of the link delay distribution obtained from the samples collected using the probes as above. This summary consists of the mean value and the decile values (i.e., the tenth, twentieth, thirtieth, etc. percentiles) for each of its neighbors. These eleven values constitute a link-state advertisement (LSA). Nodes periodically disseminate the LSA to all other nodes by piggybacking them on the link probes, and thereby maintain a link-state graph of the network with each link annotated with its delay distribution summary as reported by the most recent LSA. R3 implicitly assumes that delay distributions are stationary in the short-term, i.e., recent past is predictive of the immediate future, similar to the assumption existing protocols make about expected delay or loss rate.

4.1.3 Estimating path delay

The path delay is computed as the sum of its constituent link delays. The expected delay of a path is the sum of the expected delays of its constituent links. However, the path delay distribution is the convolution of its constituent link delay distributions. More precisely, let X_n denote the end-to-end delay of a path consisting of n links, and let $Y_1, Y_2, \cdots, Y_n$ denote the constituent link delays. Let $X_i = Y_1 + Y_2 + \cdots + Y_i$, $1 \leq i \leq n$, denote the delay of the path consisting only of the first i links. Then, $X_i = X_{i-1} + Y_i$ and the distribution of X_i is the convolution of the distributions of X_{i-1} and Y_i. In this way, a node iteratively computes the delay distributions of paths from the link delay X_1 to the total path delay X_n.

The convolutions are computed by discrete link delay values obtained from the decile values in the LSAs. The end-to-end path delay distribution as computed above may consist of many more than ten values, however this distribution is only used locally by a node to select paths as described next.

4.2 Path selection

Next, we describe how R3 selects paths for replication based on the delay distributions. The simplistic model in the previous section predicts more replication gain by using more paths, but in practice, load and interference effects will limit the achievable gain. Furthermore, the trace-driven analysis in §3 suggests that, even without load, two paths yield much of the replication gain. So, our implementation of R3 limits replication along at most two paths.

A node selects the first path by running Dijkstra's shortest path algorithm. The algorithm takes as input the expected delay of links and outputs the path with minimum expected delay for each destination. A node selects the second path by choosing one that minimizes the combined two-path delay. The choice of the second path depends on path delay distributions, not just their expected values. Let X_1 denote the delay on the first path with the minimum expected delay as computed above and $X_2, \cdots, X_m$ denote the delays of other candidate paths. The expected delay $D_{1,i}$ of replicating along both paths is

$$D_{1,i} = \int_0^{+\infty} P[X_1 > x] P[X_i > x] dx \tag{6}$$

A node picks as the secondary path (possibly partially joint with the first path) the path that minimizes $D_{1,i}$, $2 \leq i \leq m$, where $D_{1,i}$ is computed as above. Since the delay estimation algorithm using the decile values results in a discrete random variable, R3's implementation computes $D_{1,i}$ as

$$D_{1,i} = \sum_{k=0}^{N} P[X_1 > k\Delta] P[X_i > k\Delta] \Delta \tag{7}$$

Link probe period	1 second
LSA dissemination period	10 seconds
Route recalculation period	30 seconds
Hop threshold for pruning path	2
Replication threshold for adapting to load	2

Table 1: Parameters

for a suitably small interval Δ and the sum is over integers $0 \leq k < N$, where N is the smallest integer such that $N\Delta$ exceeds the largest observed delay across all candidate paths.

It is straightforward to extend equations (6) and (7) to select the best combination of $k-1$ other paths for $k > 2$. In §6, we show that this increases computational complexity but yields little additional benefit and can in fact hurt performance in realistic network and load scenarios we considered. To further limit useless replication, R3 uses the second path to replicate packets only when $D_{1,i} < 0.9 \cdot E[X_1]$, i.e., the replication gain is at least 1.1.

The number of paths to a destination can be exponential in the size of the network, so a brute force search for the best path combination can be expensive. R3 uses two simple techniques to prune the set of candidate second paths. R3 only considers paths whose hop count are within a *hop threshold* of the first path. Second, if a path is discarded, all paths longer than it are discarded.

4.3 Load-aware replication

R3 as explained so far ignored the impact of load. However, under high load conditions, replication yields little benefit and can in fact severely hurt performance (even when limited to two paths). So, R3 uses a load-aware scheme to switch from replication to forwarding.

Each source node tracks the actual delay of replicating packets along two paths by having the destination send a packet acknowledgment for the earliest delivered copy of each packet. If the actual two-path delay exceeds a *replication threshold* of the estimated two-path delay, the node treats it as a sign of network congestion and reverts to single-path forwarding along the shortest path.

The node keeps monitoring the actual two-path delay by sending infrequent probe packets on the second path. It switches back to replication when the actual two-path delays falls below the replication threshold of the estimated two-path delay. Our experiments in §6 suggest that this simple heuristic is sufficiently responsive to high load enabling R3 to achieve performance comparable to state-of-the-art forwarding protocols.

4.4 Implementation details

R3 source-routes data packets by including the entire path in the packet header in order to avoid routing loops. When the destination receives the packet, it sends an acknowledgment reporting the time of receipt on each path along which it received the packet. If the absolute delay values are on the order of minutes or longer (as in DTNs), R3 uses the timestamp reported in the acknowledgment to approximate the one-way path delay. This assumes loosely synchronized clocks across nodes, which we expect to hold in practice. If the absolute delay values is small on the order of seconds or shorter (as in meshes), R3 estimates one-way path delay as half the round trip delay to receive the acknowledgment.

We send link probes and probe acknowledgements as broadcast packets to avoid retransmissions by the 802.11 MAC. We also give them the highest priority in the wireless driver

Day	1	2	3	4	5	6	7	8	9	10
Load (pkt /hour/flow)	5	10	15	20	25	30	35	40	45	50
Avg delay (hour)	.41	.43	.51	.57	.63	.69	.73	.79	.82	.89
Percentage delivered (%)	92	88	91	91	88	84	83	83	81	81

Table 2: Deployment results of R3 on DieselNet-DTN

so that they are inserted at the head of the driver queue and don't experience high queuing delay as reported in [18]. To keep routing overhead low, R3 nodes propagate LSAs only if the expected delay or any of the decile values change by more than 10%. Each node has a limited buffer, and it removes old packets when the buffer is full.

Table 1 shows the parameters used in our implementation. We pick these parameters based on our experiments.

5. DEPLOYMENT

We deployed a prototype of R3 on a well-connected mesh testbed and a highly disconnected vehicular testbed. To our knowledge, R3 is the first routing protocol that is implemented on both a mesh and a vehicular testbed.

5.1 Vehicular deployment

We deployed R3 on a vehicular testbed, which we refer to as DieselNet-DTN. DieselNet-DTN comprises of 35 public transport vehicles operating in a 150 sq. mile area, out of which 20 are on the road everyday, on an average. Each vehicle is equipped with a Hacom OpenBrick 1GHz Intel Celeron M system running Linux 2.6 and MadWiFi cards.

5.1.1 Vehicular deployment results

We measured the routing performance of R3 on ten weekdays by generating increasing loads from five packet per hour per flow to fifty packet per hour per flow. The measurement results are shown in Table 5. The average packet delay across days is between 0.41 hour and 0.89 hour and the average percentage of packets delivered are between 81% and 92%.

5.1.2 Using the deployment results for validation

While we evaluate the performance of R3 using our deployment, for a broader evaluation and for comparisons, we conduct trace-driven experiments using a variety of traces. One of the traces is collected from our vehicular testbed, where we collect GPS traces from our vehicles and compare multiple protocols on the same trace. We validate our simulator by comparing simulation result on R3 against the 10-days measurements from the deployment. Figure 7 shows the average delay of the deployment results and the simulator. Delays measured using the simulator were averaged over 5 runs and the error-bars show a 95% confidence interval. The close correlation shows the accuracy of our simulator.

5.2 Mesh deployment

We deployed a prototype of R3 on the Mesh testbed referred to as Mesh. Mesh is a well-connected wireless mesh testbed consisting of 16 nodes in one floor of our computer science building (Figure 8). Each node is an Mac Mini computer running Linux 2.6 with 802.11b Atheros/MadWiFi wireless card. The cards are configured to send at 5.5Mbps. RTS/ CTS is turned off, and the cards are set to ad hoc mode. The R3 prototype runs as a user-space daemon. Path

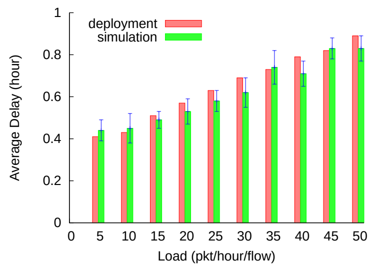


Figure 7: Average delay for 10 days of R3 deployment on DieselNet-DTN compared to trace-driven simulator.

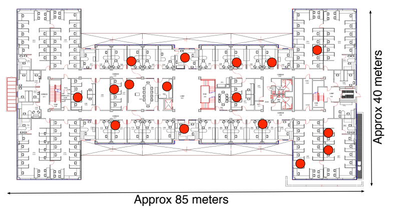


Figure 8: The Mesh testbed showing node locations.

lengths vary between 1 and 5 hops. We defer results from the mesh deployment to Section 6.5.

6. EVALUATION

The evaluation of R3 is broadly categorized into the following categories (1) Varying connectivity: We show that R3 is robust in diverse networks and can adapt to changing connectivity better than protocols that are designed for specific environments, (2) Varying load: We show that R3 adapts well to changing load, (3) Analysis: We analyze the overhead of R3 and isolate the reasons for the better performance of R3, and (4) Homogenous networks: We show that in a specific environment, the performance of R3 is comparable to state-of-the-art protocols designed for the specific environment.

6.1 Experimental setup

6.1.1 Trace-driven evaluation

Our trace-driven evaluation is based on traces from three different testbeds referred to as Haggle, DieselNet-Hybrid, and DieselNet-DTN and use the QualNet simulator [4]. We summarize the statistics of all testbeds in Table 3. Recall that the DieselNet-DTN testbed was also our deployment testbed, and we validated our simulator using traces collected from DieselNet-DTN.

Haggle [13] is an opportunistic network with temporally diverse connectivity (Figure 2(a)). The trace consists of a list of contacts in the format (i, j, s, e), where i and j are two nodes, s and e are the start and end time of a contact between them. DieselNet-Hybrid is a hybrid mesh-DTN testbed with spatially diverse connectivity. The testbed consists of 20 buses in a 4 sq.mile area forming a sparse network and 40 mesh APs. The mesh APs in the testbed form several mesh clusters (shown in Figure 2(b)), and the buses have continuous connectivity to the mesh APs when in range of the mesh cluster, forming a well-connected network. When a bus is not in a mesh cluster, it routes data through other buses using DTN routing. The DieselNet-DTN testbed is a sparsely-connected network where we remove the mesh clusters from the DieselNet-Hybrid testbed.

	Num of nodes	Connectivity	Load interval
Mesh	16	Well connected	[1, 6] pkt/sec/flow
Haggle	9	Temporally varying	[0.5, 3] pkt/min/flow
DieselNet-Hybrid	60	Spatially varying	[0.5, 3] pkt/min/flow
DieselNet-DTN	20	Sparsely connected	[5, 50] pkt/hour/flow

Table 3: Testbeds used in the evaluation

In DieselNet-Hybrid and DieselNet-DTN, we *a priori* infer the GPS coordinates of the APs in the mesh cluster and log the GPS coordinate of each bus. Any two nodes in the trace are said to be in contact when they are within 100 meters of each other, as inferred from their GPS locations.

Given the length of a contact between two nodes (that is obtained from the contact schedule in Haggle and from the GPS coordinates in DieselNet-Hybrid and DieselNet-DTN), QualNet simulates data transfer during the period of the contact. The data rate is set to 5.5Mbps using two-ray pathloss model and rayleigh fading model.

6.1.2 Emulation-based evaluation

To further stress-test R3's performance in diverse connectivity scenarios, we use the mesh testbed to emulate networks with changing connectivity (but without emulating effects of mobility). We let each node to be in an on or off state, and vary the on/off duration according to an exponential distribution. We vary the mean of the exponential distribution to get different connectivity, and as before, we define connectivity according to the fraction of connected nodes. The connectivity (or the fraction of connected nodes) increases from 0.01 to 1, ranging from sparsely-connected networks to well-connected networks. We conduct 30 minute experiments at each connectivity level. Figure 11(a) shows the mean on/off time of each node and the fraction of connected nodes over time.

6.1.3 Alternate routing protocols

We compare R3 against the following protocols:

Replication-based: *(i)* RAPID [8], a DTN routing protocol that makes replication decision based on packet utilities and *(ii)* Random, a simple replication-based protocol that replicates packets with a fixed probability of 0.5.

Forwarding-based: *(i)* DTLSR [16], a DTN routing protocol based on link-state forwarding that uses delays as the link metric, *(ii)* OLSR [3], a mesh routing protocol based on link-state forwarding that uses ETX [15] as the link metric, and *(iii)* AODV [29], a mesh routing protocol based on distance-vector forwarding using the hop count metric.

Multi-configuration: We implement a "multi-configuration" protocol referred to as SWITCH. SWITCH concurrently runs both OLSR and RAPID and switches between the two depending of whether a node locally perceives itself as being in a mesh or in a DTN. SWITCH forwards packets using OLSR if there is a contemporaneous end-to-end path reported by OLSR; otherwise it replicates packets via RAPID. SWITCH nodes make routing decisions independently for each packet, so a packet may switch mid-route from replication to a forwarding route or vice-versa.

We implemented SWITCH, RAPID, Random and DTLSR in our Mesh testbed and in the QualNet simulator, and modify existing implementations of AODV [1] and OLSR [3].

While evaluating the above protocols in environments they are not designed for, we make some straightforward changes for a fair comparison. AODV and OLSR assume a contemporaneous end-to-end path. To make AODV and OLSR work (i.e., deliver any packets at all) over sparsely-connected networks, we allow each node to buffer data packets until the packets are transferred; if the buffer is full, then we purge the packets. RAPID's design based on inter-contact times and transfer opportunities makes it unusable as-is on a mesh, so we modify it to use expected delays or ETT [19] (specifically in Eq. 8 in [8]) in mesh networks. We preserve RAPID's assumption of exponentially distributed delays, i.e., k replicas of a packet reduce delay k-fold, as that is central to its design and ignores the nature of actual delay distributions.

To reduce clutter, we defer the results of Random and AODV to a technical report [35]. Random consistently performs worse than R3 and RAPID, and AODV consistently performs close to OLSR.

6.1.4 Load and metrics

We generate 30 concurrent flows between randomly chosen source-destination pairs in all experiments. In the DieselNet-Hybrid testbed, where nodes can either be a bus or an AP, packets either flow from a bus to the AP or vice versa. We vary the rate of the flows to change the network load. Table 3 shows the load interval on each testbed. Each packet is of size 1.5KB and node buffers are limited to 1MB.

We quantify the performance of the protocols in terms of *delay* and *goodput*. We measure the average delay across all packets, where the delay of undelivered packet is the time it has spent in the network. We measure goodput as the average rate of packet reception over the experiment period.

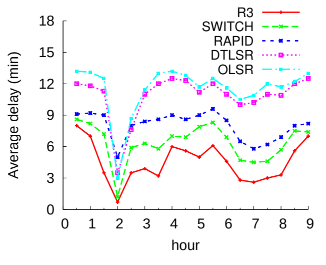
6.2 Experiments with varying connectivity

6.2.1 Haggle: Temporally diverse connectivity

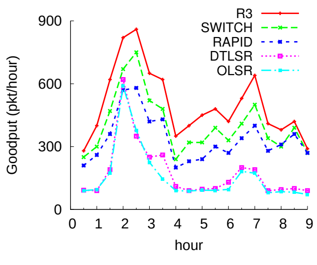
Figure 9 shows the delay and goodput of different protocols over Haggle. Figure 2(a) previously showed the connectivity, i.e., the fraciton of connected nodes, in Haggle over time. The load is set to 1 packet/minute/flow, which represents a total load of 270 packets/min across the network. The delay and goodput are measured every half hour.

Figure 9 shows that R3 consistently performs better than SWITCH, RAPID, OLSR, and DTLSR under varying connectivity. R3 improves delay by up to 1.6× and improves goodput by up to 1.3× over the second best protocol SWITCH. We note that SWITCH tunes its routing algorithm according to the network connectivity. When the fraction of connected nodes drops below 0.1 (in the first and last half hour) or increases above 0.7 (between 1.5 and 2 hours) in Figure 2(a), the corresponding performance in Figure 9 of SWITCH is comparable to R3. The two connectivity ranges can be loosely characterized as "sparsely-connected" or "well-connected". In other words, in the two ends of the connectivity spectrum, a multi-configuration protocol that switches routing according to connectivity performs similar to an adaptive protocol such as R3. However, when the network connectivity cannot be classified as either well-connected or sparsely connected (fraction of connected nodes between 0.1 and 0.7), R3 outperforms SWITCH. We analyze the underlying reasons further in Secton 6.4.2.

Since RAPID, DTLSR, and OLSR perform worse than R3 and SWITCH in other diverse connectivity environments,

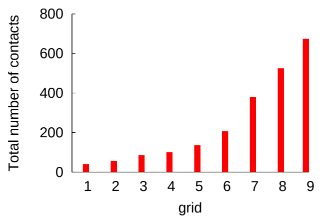


(a) Delay

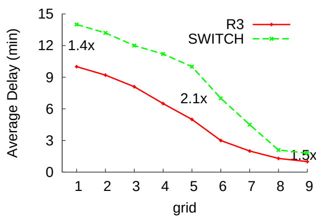


(b) Goodput

Figure 9: [Haggle] Temporally varying connectivity: Delay and goodput are measured every half hour. R3 improves delay by up to 1.6× and goodput by up to 1.3×.

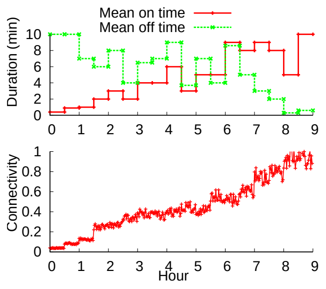


(a) Total no. grid-wise contacts

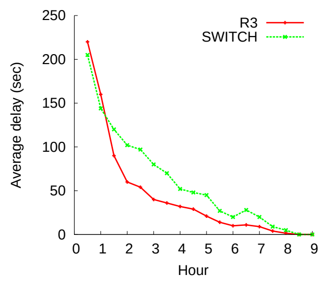


(b) Avg. delay in each grid

Figure 10: [DieselNet-Hybrid] Spatially varying connectivity: R3 has 2.1× better delay than SWITCH.



(a) Connectivity



(b) Delay

Figure 11: [Mesh] Emulated connectivity patterns: R3 improves delay by up to 2×.

we defer their results to a technical report [35] and plot only SWITCH's performance for clarity of presentation in the rest of the graphs. RAPID, DTLSR and OLSR performs much worse than R3 and SWITCH, because they don't adapt to changing connectivity. RAPID performs comparably to R3 and SWITCH in the first and last half hour of Figure 9(a), which confirms that it is designed to work in sparsely-connected networks. However, its replication policy wastes resources and hurts when network connectivity improves. DTLSR and OLSR as forwarding protocols only work well in mostly well-connected networks.

6.2.2 *DieselNet-Hybrid: Spatially diverse connectivity*

We evaluate R3 on DieselNet-Hybrid that exhibits spatially diverse connectivity as shown in Figure 2(b). The total number of contacts in each grid is shown in Figure 10(a). We set the load to 1 packet/minute/flow and conduct the experiment over half a day of the trace data. Delay in a grid is the average delay of all packets received in this grid, i.e., the packets whose destination is a static AP in the grid or a mobile bus travelling in this grid when it receives the packet. Recall that for an undelivered packet, we count its delay as the time it spends in the network. Thus if the undelivered packet is destined to an AP, its delay is counted in the delay of the grid in which the AP locates; if it is destined to a mobile bus, its delay is counted in the delay of the grid in which the bus travels at the end of the experiment.

Figure 10(b) shows the average delay of packets received in each grid. R3 improves delay by up to 2.1× over SWITCH. The improvement of R3 varies across grids. In grids 1, 2 and 8, 9, where the network is either sparsely connected or well-connected, R3 has delay improvement of around 1.4× and 1.5×. In grids 4, 5 and 6, where the connectivity cannot be classified as either sparsely connected or well-connected, R3 has an improvement of 2.1× over SWITCH. This trend is similar to the Haggle results shown earlier in Figure 9.

6.2.3 *Mesh: Emulating diverse connectivity*

Figure 11(b) shows the average delay in the emulated mesh testbed every half hour. We set the load to 1 pkt/min/flow in this experiment. Similar to our observation in Haggle (Figure 9) and DieselNet-Hybrid (Figure 10(b)), R3 improves delay by up to 2× over SWITCH when connectivity varies between 0.1 and 0.8 (from the time 1 hour to 8 hour). R3 performs close to SWITCH in both sparse conditions of less than 0.1 fraction of connected nodes and in well-connected environments of greater than 0.8 fraction of connected nodes.

6.3 Experiments with varying load

The above experiments evaluate R3 under a fixed network load. We now evaluate it under varying load. We compare the following protocols: R3, R3 without load-aware adaptation, and SWITCH. The load for each flow is increased from 0.5 packets/minute to 3 packet/minute. In Haggle, for example, this represents a total of 135 to 800 packets/min. The experiment duration is 9 hours on Haggle and half a day on DieselNet-Hybrid. We measure the average delay of all packets at the end of each experiment.

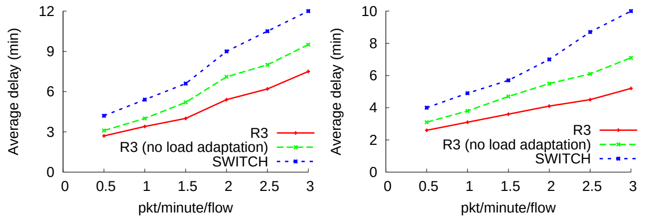
Figure 12 shows that R3 performs well across varying loads and improves delays by up to 1.8× in Haggle and 2.2× in DieselNet-Hybrid. Even even R3 does not adapt to load, it improves delays by up to 1.35× on Haggle and 1.40× on DieselNet-Hybrid over SWITCH.

Figure 13 compares the packet delay CDF of R3 and SWITCH under two load conditions in Haggle. Low load is set to 1 packet/minute/flow and medium load is set to 2 packet/per/minute per flow. As load increases, the performance difference between R3 and SWITCH increases. At low load, the median delay improvement of R3 over SWITCH is 1.6×, while at medium load, the median delay improvement is 2.1×. By design, R3 adapts replication to the load, but, SWITCH does not and continues aggressive replication hurting performance especially when network resources are stressed due to high load.

6.4 Analysing R3

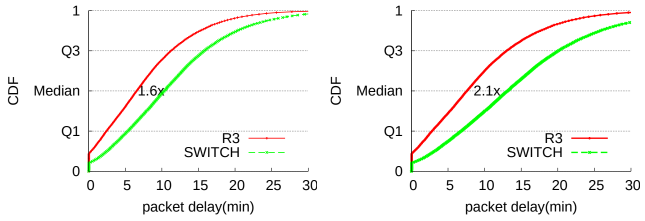
6.4.1 *Overhead of R3*

Although the previous experiments implicitly incorporated the effect of routing overhead for all protocols, we explicitly analyze the routing and computation overhead of R3. Different protocols incur different overheads, depending on what information they exchange with their neighbors *(i)*



(a) Haggle (b) DieselNet-Hybrid

Figure 12: Performance under varying load: R3 improves delay by upto 1.8× on Haggle and upto 2.2× on DieselNet-Hybrid.



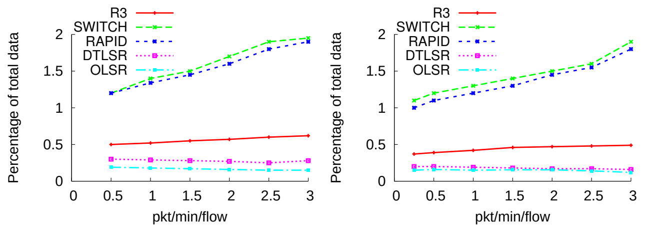
(a) Low load (b) Medium load

Figure 13: [Haggle] Delay CDF under varying load: R3 improves median delay by 1.6× under low load, and by 2.1× under medium load.

R3: link-state announcement and packet acknowledgments, *(ii)* RAPID: information about packet replicas and contact information, *(iii)* DTLSR: link state announcements, *(iv)* OLSR: link state announcement and other control messages, *(v)* SWITCH: the routing overhead of RAPID + OLSR. We compute the routing overhead as the percentage of the total traffic data.

Figure 14 shows the routing overhead of different protocols on Haggle and DieselNet-Hybrid. R3 has less than 0.5% routing overhead across both testbeds, though it's overhead is higher than DTLSR and OLSR. The previous experiments suggest that the benefits of R3 justifies this increase in overhead. The fact that the DieselNet-Hybrid network has 60 nodes suggests that R3 scales well with network size. The routing overhead of DTLSR and OLSR decreases as load increases because they do not incur per-packet overhead. SWITCH and RAPID incur the highest overhead because they disseminate packet replica locations in addition to information about past node contacts.

R3's computation overhead stems from: (1) estimating the path delay distribution, which is a function of the number of links in the path, and, (2) selecting $k-1$ secondary paths that combine best with the primary path, which is the

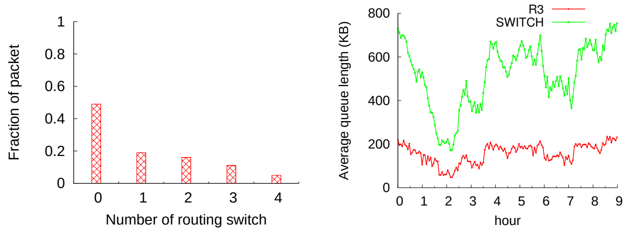


(a) Haggle (b) DieselNet-Hybrid

Figure 14: Routing overhead as a percentage of total traffic data. R3 has less than 0.5% overhead.

	Haggle	DieselNet -Hybrid	DieselNet -DTN
No. of paths	14.6	23.8	15.2
Avg. no. links per path	2.4	3.7	3.1
Path combinations ($k=2$)	14.6	22.8	15.2
Path combinations ($k=3$)	45.3	160.2	62.6

Table 4: Computational overhead on different testbeds



(a) Number of switches (b) Queue lengths

Figure 15: [Haggle] Analyzing SWITCH forwarding.

function of the number of path combinations. The average hop count of the paths is less than 4 on all testbeds, thus computing the convolution doesn't induce much complexity. However, the number of path combinations increases exponentially with the number of replication paths, k, as shown in Table 4, so the empirical observation that a small number of paths suffice to yield most of the replication gain is a fortunate one for R3.

6.4.2 Why R3 outperforms SWITCH?

R3 outperforms SWITCH due to two reasons: (1) SWITCH vacillates between different routing protocols based on each node's local view resulting in inconsistent and sub-optimal routes; (2) SWITCH (or the underlying RAPID protocol) makes poor replication decisions compared to R3 resulting is longer queues and increased packet delays. To validate these claims, we perform the following analyses.

Figure 15(a) shows the number of times SWITCH switches routing protocols (in this case, OLSR and RAPID) to deliver a packet. We count for each packet, the maximum number of routing switches of all its copies, where routing switch occurs when a packet is routed by OLSR and then switches to RAPID, or vice versa, in two consecutive hops. For example, when the packet is routed either by OLSR or RAPID throughout its route, the number of routing switches for the packet is 0. Figure 15(a) shows that 50% of the packets experience at least one routing switch and 30% of the packets experience two routing switches or more. These inconsistencies suggest that SWITCH makes some unnecessary replicas that increase load-dependent queuing delays, especially for packets that are only forwarded along a single route.

Figure 15(b) shows the average queue lengths at nodes when using R3 and SWITCH. SWITCH has on an average 3× longer queues compared to R3. The higher queue lengths in SWITCH results in longer packet delays. The reason for the inflated queue lengths is SWITCH's (or RAPID's) aggressive replication that does not adapt replication either to load or to the extent of delay uncertainty.

6.4.3 Fairness

We also evaluate R3 in terms of fairness. We generate 30 concurrent backlogged flows and Figure 16 shows the CDF of goodput across flows in Haggle. R3 outperforms SWITCH for every percentile and the median gain is 1.25×. The Jain's

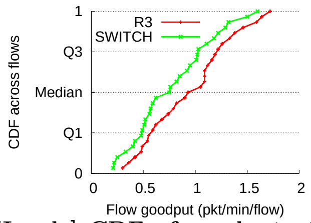


Figure 16: [Haggle] CDF of goodput of 30 concurrent backlogged flows: Jain's fairness index for R3 is 0.85 compared to 0.81 for SWITCH.

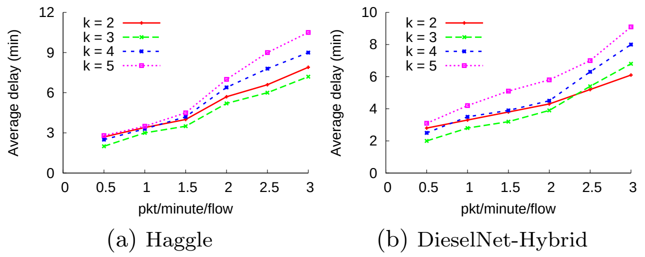


Figure 17: Average delay when R3 uses $k = 2, 3, 4, 5$ replication paths. $k = 2$ (default R3) gets most of the replication benefit; more paths hurt under high load.

fairness indexes for R3 and SWITCH are 0.85 and 0.81, respectively. R3 is fairer than SWITCH, indicating that R3 improves goodput without hurting fairness.

6.5 Homogeneous connectivity

Finally, we conduct experiments in a fixed connectivity network and choose two end points of the connectivity spectrum: sparsely-connected and well-connected networks.

6.5.1 *Sparsely-connected networks*

Figure 18 shows that under low load, R3 performs comparably to RAPID, a replication routing protocol, on the DieselNet-DTN testbed. However, R3 achieves 1.4× delay and 1.3× goodput improvement under high load. This is because RAPID does not adapt its replication to the load and can waste resources while R3's load-aware replication enables better performance under high load. DTLSR and OLSR perform much worse compared to both RAPID and R3 as DTLSR does not use replication and OLSR is not designed for mostly disconnected networks.

6.5.2 *Well-connected networks*

Figure 19 was conducted on our mesh deployment (Section 5.2), and shows that R3 has similar performance to OLSR, a mesh protocol on the mesh testbed. As expected, R3 has up to 2× delay and 2.2× goodput improvement over RAPID, since RAPID is not designed for meshes.

7. RELATED WORK

Our work differs from prior work primarily in its goal—to design and implement a simple routing protocol that achieves robust performance across wireless networks with diverse connectivity characteristics—that to our knowledge has not been pursued before. To address this goal, R3's liberally borrows insights from a large body of prior work in wireless routing as discussed below.

Replication routing, also known as epidemic routing [36], multi-copy routing [32], or controlled flooding [22] in the literature, has been well studied over the last decade [27, 33, 8, 30, 12, 34, 28, 36, 32]. Unlike R3 however, most existing replication routing protocols are primarily designed for highly disconnected networks where pairs of nodes meet each other infrequently. Furthermore, although existing protocols leverage replication to improve delays in DTNs, the question of when replication helps and by how much has not received careful attention. Given our goals, a foundational as well as trace-driven analysis of this question forms an important focus of our work (as in §3 and §2).

Many DTN routing protocols make explicit or implicit assumptions about the inter-meeting of nodes. For example, RAPID [8] assumes that the meeting times are exponentially distributed; Thrasyvoulos et al. [33, 34] assume a random-walk mobility model; These assumptions can restrict their applicability to other environments. For example, RAPID assumes that two replicas will halve the delay, an assumption that is unlikely to hold in well-connected meshes. Even in DTNs, if the mobility schedule is known a priori, as may be the case if buses stick to a fixed schedule, replication is unnecessary.

In comparison, R3 explicitly measures the distribution of path delays and uses the nature of this distribution to control replication, so it is applicable to broad spectrum of mobility or disconnection patterns. This idea is similar in spirit to Francois et al. [21] who develop a theoretical routing framework based on known delay distributions to replicate packets so as to achieve statistical delay guarantees in DTNs with unconstrained bandwidth. In comparison, our results in §3 do not assume knowledge about the underlying delay distribution and our primary focus is on a design and implementation effort targeting diverse wireless network environments with realistic bandwidth constraints.

Existing replication protocols use several schemes to control replication such as probabilistic replication [22, 27], utility replication [33, 8], prioritizing transmit order [30, 8], acknowledgments to remove useless packet [12], and explicitly bounding replicas [34, 33]. R3's design shares some of these ideas including bounding the number of replicas to two, but additionally compares actual packet delays to pre-

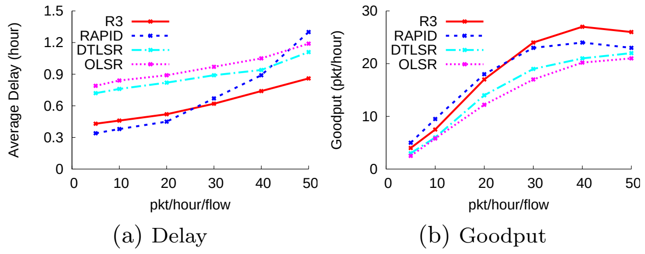


Figure 18: [DieselNet-DTN] Sparsely-connected network: R3 performs comparably to RAPID under low load, and outperforms RAPID under high load.

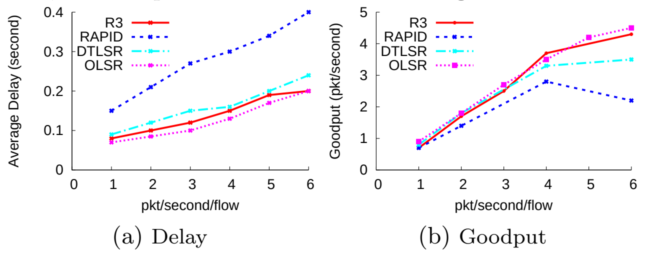


Figure 19: [Mesh] Well-connected network: R3 performs within 8% of OLSR, a mesh protocol.

dicted packet delays to judge the effectiveness of replication and turn it off as needed, thereby making it more responsive to load or interference.

We were able to compare R3 against only a small number of state-of-the-art protocols for which implementations were available to us. R3 by no means is intended to subsume the staggering diversity of existing routing protocols. For example, R3's design may be inefficient in settings to which a geographic (forwarding-based) routing protocol is better suited, e.g., when the node mobility pattern is known a priori. Combining R3 with complementary cross-layer optimizations such as opportunistic routing [11], network coding [25], etc. or application-specific optimizations such as information retrieval [9] etc. also requires more research. We believe that investigating all of these broad research issues related to a unified routing protocol for diverse wireless networks requires the context of a concrete (even if preliminary) proposal on the table, which is R3.

8. OPEN ISSUES AND CONCLUSIONS

Although we started out with an ambitious goal—to develop a simple routing protocol that ensures robust performance across diverse wireless networks—our work in retrospect is but a first step towards that goal leaving open several questions for future research. First, our evaluation is limited to connectivity traces from a small number of real-world networks and synthetically emulated connectivity patterns, which may not generalize to other connectivity patterns. For example, the empirical observation that two paths achieve the most replication gain may not hold, in which case generalizing R3 to use $k > 2$ paths will increase its computational as well as design complexity. Second, our simple model ignores the effects of load and interference and extending it, e.g., to compute the delay gain of replication given a feasible demand matrix, link delay distributions, and interference model is an open problem.

Acknowledgments We gratefully acknowledge the feedback we received from Manikandan Somasundaram, our shepherd Edward Knightly and anonymous reviewers. This work was supported in part by NSF grants CNS-0845855, CNS-1040781, and CNS-0910671.

9. REFERENCES

[1] Aodv. http://moment.cs.ucsb.edu/AODV/aodv.html.

[2] Google wifi for mountain view. http://wifi.google.com/.

[3] Olsr. http: //www.olsr.org/.

[4] Qualnet. http://www.scalable-networks.com/products.

[5] Turtlenet. http://prisms.cs.umass.edu/dome/turtlenet.

[6] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for ieee 802.11 wireless networks. *BroadNets'04*.

[7] A. Al Hanbali, A. A. Kherani, and P. Nain. Simple models for the performance evaluation of a class of two-hop relay protocols. *NETWORKING'07*.

[8] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. *Sigcomm'07*.

[9] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Enabling interactive applications in hybrid networks. *Mobicom'08*.

[10] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3g using wifi: Measurement, design, and implementation. *MobiSys*, 2010.

[11] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. *SIGCOMM'05*.

[12] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. *INFOCOM'06*.

[13] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. *Infocom*, 2006.

[14] A.-M. Croicu and Y. M. Hussaini. On the expected optimal value and the optimal expected value. *Applied Mathematics and Computation*, 2006.

[15] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *MobiCom'03*.

[16] M. Demmer and K. Fall. Dtlsr: delay tolerant routing for developing regions. *NSDR*, 2007.

[17] A. Doria, M. Uden, and D. P. Pandey. Providing connectivity to the Saami nomadic community. *2nd Int. conf. Development by design*, 2002.

[18] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. *SIGCOMM*, 2004.

[19] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. *MobiCom*, 2004.

[20] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular Content Delivery Using WiFi. *MOBICOM'08*.

[21] J.-M. François and G. Leduc. Routing based on delivery distributions in predictable disruption tolerant networks. *Ad Hoc Netw.*, 2009.

[22] K. A. Harras, K. C. Almeroth, and E. M. Belding-Royer. Delay tolerant mobile networks: Controlled flooding in sparse mobile networks. *IFIP Networking*, 2005.

[23] P. Hui, A. Lindgren, and J. Crowcroft. Empirical evaluation of hybrid opportunistic networks. *COMSNETS*, 2009.

[24] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 2002.

[25] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-Level Network Coding for Wireless Mesh Networks. *SIGCOMM'08*.

[26] L. Kleinrock. *Theory, Volume 1, Queueing Systems.* Wiley-Interscience, 1975.

[27] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 2003.

[28] S. Nelson, M. Bakht, and R. Kravets. Encounter-based routing in dtns. *INFOCOM*, 2009.

[29] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. *The Second IEEE Workshop on Mbile Computing Systems and Applications*, 1999.

[30] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. *MobiOpp*, 2007.

[31] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. *MobiCom 2006.*

[32] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM Trans. Netw.*, 2008.

[33] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. *PERCOMW'07*.

[34] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. *WDTN'05*.

[35] X. Tie, A. Venkataramani, and A. Balasubramanian. Robust routing in wireless networks with diverse connectivity. Technical report, UMASS, 2011.

[36] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, 2000.

[37] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. *WDTN'05*.