Routing for Diverse Wireless Networks

Paper ID: 1569334617 Number of pages:12

Xiaozheng Tie, Aruna Balasubramanian, Manikandan Somasundaram, Arun Venkataramani
University of Massachusetts Amherst
{xztie, arunab, mani, arun}@cs.umass.edu

Abstract

Our work is motivated by a simple question: can we design a simple routing protocol that works well across diverse wireless network environments 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, namely, 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 ReGain, a routing protocol that self-adapts replication to the extent of connectivity in network conditions. We implement ReGain and evaluate its performance through deployment over a mesh testbed and trace-driven evaluations on real network testbeds exhibiting diverse connectivity. Our results show that ReGain achieves up to 2× better delay and 1.3× better goodput over existing protocols in a variety of network connectivity and load conditions.

1 INTRODUCTION

Routing in wireless networks has seen a huge body of work over the last decade as networking researchers and practitioners have been identifying new target environments such as multi-hop mesh networks, mobile or vehicular ad hoc networks (MANETs), and disruption-tolerant networks (DTNs). A variety of current or foreseeable applications motivate this research including extending the reach of the Internet [7, 6, 8], wildlife monitoring [30], content distribution [15, 13, 20], etc. Consequently, a number of routing protocols have been developed for each one of these environments.

Unfortunately, the state of the art is that routing protocols designed for one environment work poorly or break down completely in others. For example, mesh routing 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 routing protocols also assume the availability of a contemporaneous end-to-end path. Likewise, DTN routing protocols commonly use packet replication to reduce delays, but packet replication performs poorly in predictable, well connected mesh networks. In other words, existing routing protocols do not adapt to diverse network environments.

However, as wireless networks continue to proliferate, users are likely to encounter diverse environments. As we show in section 2, there are already several examples of real-world network deployments that exhibit diverse connectivity—either spatially varying connectivity with a mix of WiFi mesh and opportunistic DTNs [2] or temporally varying connectivity [20]. Exploiting these variations can increase the capacity of the network and allow users to seamlessly operate in diverse environments [13, 28].

Our work is motivated by a simple question: *can we design a simple routing protocol that works well across diverse wireless network environments?* Designing such a self-adapting protocol can provide several benefits. First, it can significantly improve performance in diverse networks compared to protocols designed for a specific environment. Second, a self-adapting protocol makes it easier to interconnect different networks and allow seamless operation. Third, adapting to diverse environments will allow protocols to be robust under changing network conditions caused by node failures or duty cycling. Finally, it reduces the engineering and management complexity of protocol design as maintaining one protocol for diverse networks is simpler than maintaining different protocols for different networks.

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 model to quantify under what conditions and by how much replication improves packet delays. We show formally as well as through experiments based on real DTN traces that replication yields significant delay gains if and only if path delays exhibit high unpredictability.

Based on these insights, we design and implement ReGain, a routing protocol that self-adapts replication to the unpredictability of path delays as well as the load in the network. ReGain achieves these properties using two key insights. First, it monitors the distribution of path delays, not just their expected value, unlike traditional DTN routing protocols. Second, it leverages the empirical finding that replicating each packet along two paths suffices to capture most of the achievable replication gain. Furthermore, ReGain uses load-aware adaptation that allows it to be responsive to changing load conditions. ReGain turns off replication along the second path and switches to single-path forwarding when it determines that the actual delay is significantly higher than the estimated delay. Under extremely high load, even single path forwarding may not achieve the net-

work capacity required to support the load. So, ReGain uses a load-aware forwarding to switch from single-path forwarding to multi-path forwarding.

We extensively evaluate ReGain using (1) a prototype deployed over a 16-node mesh network testbed, (2) trace-driven experiments on two DTN testbeds, DieselNet [12] and Haggle [20], and (3) emulation of network topologies with varying levels of connectivity all the way from well-connected meshes to highly disconnected DTNs. Our experiments show that ReGain achieves up to 2× delay and 1.35× goodput over existing protocols in networks with spatially and temporally diverse connectivity, while achieving delay and goodput comparable to or better than the state-of-the-art protocols tailored for well-connected and sparsely-connected networks.

2 WHY DESIGN FOR DIVERSITY?

In this section, we motivate the need for a self-adapting routing protocols for diverse networks. To this end, we first show that state-of-the-art routing protocols perform poorly outside the specific environment for which they are designed. We then present real-world examples of networks exhibiting significant temporal and spatial diversity in connectivity characteristics. Finally, we describe the challenges in designing a self-adapting routing protocol that works well in networks with temporal or spatial diversity in connectivity.

2.1 Routing performance in diverse networks

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 [12], Random replication, DTLSR [23], AODV [33] and OLSR [4]. 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 two sparsely-connected testbeds, DieselNet[16], Haggle[20], and we conduct a deployment-based experiment 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 §5, which also presents a more exhaustive exploration of protocol, workload, and environment parameters.

Figure 1(a) shows that in well-connected mesh, replication routing using RAPID increases delay by about 2× compared to OLSR, the best forwarding protocol. Replication wastes resources while yielding little benefit in well-connected environments.

However, the situation reverses in sparse networks. Figure 1(b) shows that in DieselNet DTN environment, replication routing using RAPID yields almost a 2× reduction in delay compared to DTLSR, the best forwarding protocol. Even random replication significantly reduces delay compared to forwarding protocols. We observe qualitatively similar results for the Haggle traces (not shown in figure).

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 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.

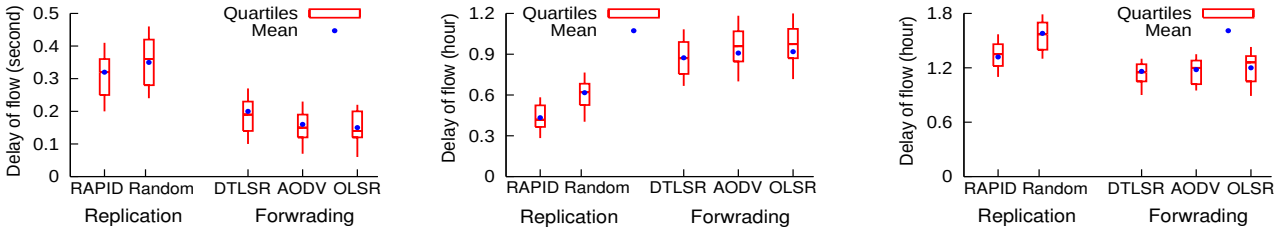
2.2 Temporal and spatial diversity

Figure 2(a) shows that the *connectivity*, i.e., fraction of connected node-pairs in the Haggle network varies *temporally*. Haggle [20] is an opportunistic network formed by 8 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 as a result of user mobility. The connectivity is less than 10% for 40% of the time, and is over 20% for 45% of the time. 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 bus network's connectivity varies *spatially*. DieselNet[2] is a hybrid mesh-DTN testbed consisting of 20 buses operating in a 150 sq.mile area. Buses are connected either when they come in contact with mesh access points (APs) or other buses. Figure 2(b) shows that the connectivity of buses varies with geographical location. The total number of bus-AP and bus-bus contacts per day are over 500 in the town (center grid) and the campus center (upper center grid), but are below 100 in locations farther from the campus (right three grids).

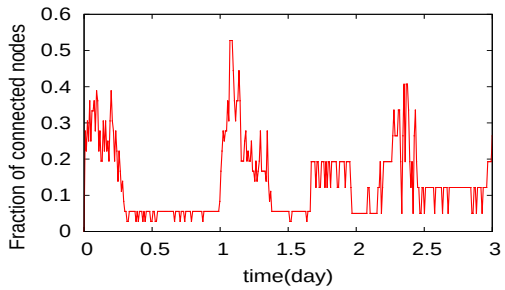
Changes in network connectivity across location is often the result of the difference in wireless penetration. For example, in the hybrid mesh-DTN topology of DieselNet, buses are well connected in urban centers with high WiFi AP density (as shown by the mesh clusters in Figure 2(b)), but are poorly connected as they move to less urban areas. Recent measurement studies show that such variations in connectivity can occur even in cellular networks [14].

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) can significantly increase wireless capacity. For example, Balasubramanian et al [13] show that the performance of delay-tolerant Web applications can be improved using a sparse-DTN when available. Similarly, Hui et al [28] show that delay-tolerant and opportunistic communications can double the throughput of a well-connected mesh network. Thus, in order to truly exploit the potential of hybrid networks, protocols should self-adapt to changing connectivity patterns.

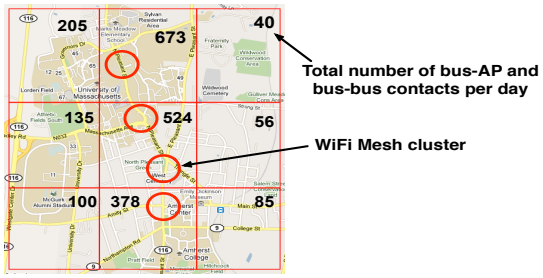


(a) Mesh: low load (4 pkt/second/flow) (b) DieselNe-DTN: low load (20 pkt/hour/flow) (c) DieselNet-DTN: high load (50 pkt/hour/flow)

Figure 1. Each boxplot shows min, max, 25%, 75% quartiles, median, and mean packet delays. 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: Connectivity varies spatially. The region includes UMASS campus and Amherst town center (4 sq.mile)

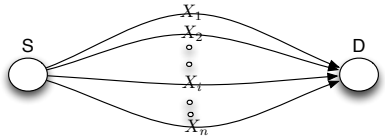
Figure 2. Networks that exhibit varying connectivity characteristics temporally and spatially

2.3 Designing a self-adapting routing protocol

In order to address the challenges of designing a self-adapting routing protocol, we begin by observing a key structural difference between routing protocols designed for well-connected networks and sparsely-connected networks, namely, replication. Therefore, we ask the question: *Under what connectivity and load conditions should replication be used over forwarding?* In §3 we develop a simple model to first understand the benefits of replication in terms of the network characteristics, ignoring the impact of load. We then use the insights from the model to design ReGain that tunes replication according to the network connectivity while being responsive to load.

3 QUANTIFYING REPLICATION GAIN

In this section, we develop an analytical model to quantify when and by how much replication improves delay. A foundational understanding of this question is crucial to the design of a routing protocol that self-adapts to diverse network environments. The model formally shows that replication yields big gains if and only if delays of different paths are highly unpredictable. We also perform trace-driven experiments to show that twofold replication is sufficient to achieve most of the available gain in practice.



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

Figure 3. Model

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. In §4 we describe a practical technique to measure this end-to-end path delays, considering both bandwidth and link loss rates.

We also make a simplifying assumption, namely, that the random variables $X_1, \cdots, X_n$ are independent. Note that this assumption is rather simplistic as it implies that there is no interference between packets traversing different paths and thus ignores any effects induced by load. However, abstracting away load effects yields simple insights into the best case gains of replication. Later, we consider the effect of load on the routing decision (§4).

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 [1]

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

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

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 [21] 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

[1] $E[\cdot]$ denotes the expected value of a random variable.

show (refer to the Appendix) that the replication gain is

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

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[37, 36] or make assumptions about the delay distribution *a priori* [12, 37, 41, 36, 11]. For example, RAPID [12] 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 ReGain (§4), which makes replication decisions based on the delay distribution, but makes no assumptions about the distribution *a priori*. Our evaluations (§5) 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 results in a mean delay of about 0.2s, a 5× improvement compared to the shortest path forwarding. We first 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 $\varepsilon E[X]$ is at least $1-\varepsilon$, i.e., $P[X \leq \varepsilon E[X]] \geq 1-\varepsilon$.

Note that $\varepsilon = 1$ 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, $\varepsilon = 1$. When X_1 is 0.1s in 90% of the cases and 10s in 10%, the predictability $\varepsilon = 0.1$. We arrive at these predictability values by solving the equation $P[X_1 \leq \varepsilon E[X_1]] \geq 1-\varepsilon$. Intuitively, as in the example, lower predictability implies higher replication gain[2]. We formalize this claim using the following theorem.

[2]The predictability is different from variance. It is easy to show that high variance doesn't lead to high replication gain.

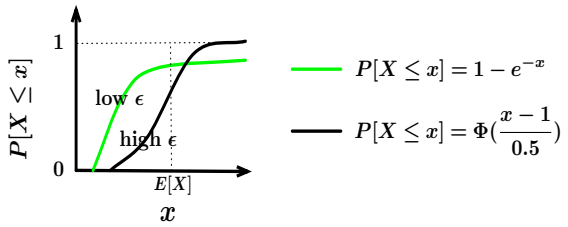


Figure 4. Distributions with more values below the mean have lower predictability

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 $\frac{1}{1-(1-\varepsilon)^2}$, 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:

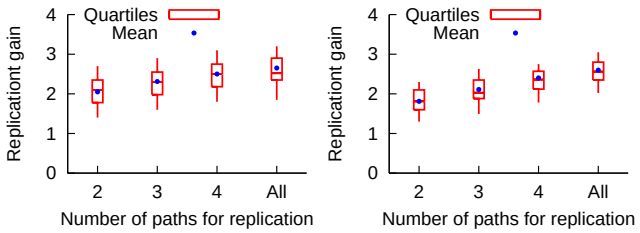
Definition 2. For a set of random variables $X_1, \cdots, X_n$, let X_m have the minimum expected delay. Then the relative predictability of a random variable X_i is the smallest δ such that $P[X \leq \delta E[X_m]] \geq 1-\delta$.

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 the Appendix.

3.4 Implications

The theorems yield two important implications. First, 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. Using the traces, we generate link-state graphs of link delays. Using this graph, we infer the path delay distributions and use Eq. 3 to estimate replication gain 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.



(a) DieselNet-DTN (b) Haggle

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

Further, since the predictability of a path can help infer replication gain, the theorems can be used to prune paths that are not suitable for replication. We use these insights to design ReGain, described in the next section.

4 ReGain DESIGN

In this section, we present ReGain, a routing protocol that self-adapts the level of replication to network and load conditions. ReGain'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 their mean value; (2) two paths suffice to capture most of the replication gain; (3) under high load, replication yields little benefit and hurts performance.

ReGain consists of three main components—*(i)* estimating the end-to-end delay distribution, *(ii)* selecting the optimal path for replication based on the delay distribution, *(iii)* adapting replication and forwarding on multiple paths to network and load conditions.

4.1 Delay estimation

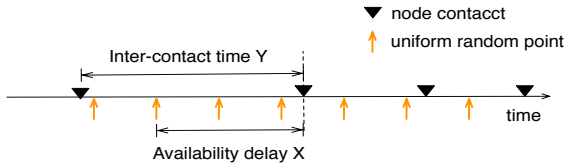
The model presented in §3 quantifies the replication gain as a function of the end-to-end delay distribution. However, estimating the delay distribution in a unified manner across diverse networks is non-trivial. For example, in well-connected mesh networks, ETT[24], Per-hop RTT[10] or similar metrics 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[12, 11, 36] are often used to measure link delays, but they are not meaningful in mesh environments where links are persistent.

Instead, ReGain uses a unified metric to capture link delays in both well-connected and sparse networks. DTLSR[23] uses a similar unified metric to capture link delays but is a forwarding-only protocol that does not self-adapt. We compare the performance of ReGain with DTLSR in Section 5.

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.

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,



Nodes contact intermittently.
The contact durations are much smaller than inter-contact time

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

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.

To appreciate the distinction formally, let X be a random variable representing the availability delay and Y be a random variable representing the inter-contact time. For simplicity, assume that the contact durations are much smaller than the inter-contact times. By definition, X represents the time until the next contact sampled at a uniformly random point in time, while Y is the time between contacts. Figure 6 illustrates the difference between the availability delay and the inter-contact time. It can be shown that (see proof in the Appendix) that X and Y are related as follows:

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

and

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

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

In other words, $E[X]$ is not equal to $E[Y]$ in the general case, contrary to explicit or implicit assumptions made by prior routing protocols [12, 38, 11, 36]. However, in the special case when the inter-contact time Y is exponentially distributed, X is identical to Y. 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, ReGain explicitly measures the availability delay 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 periodically sends out probes and one-hop neighbors who receive the probe 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). In other words, 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). 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.

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 (by linearity of expectations [18]). 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 one-link path X_1 to n-link path 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

In this section, we describe how ReGain selects paths for replication based on the delay distributions. ReGain uses at most two paths for replicating packets in accordance with the trace-driven analysis in §3. 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

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

A node picks as the secondary 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, ReGain's implementation computes $D_{1,i}$ as

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

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. A node 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 candidate second paths to a destination can be exponential in the size of the network, so a brute force search for the best second path using delay distributions as above can be expensive. ReGain uses a simple heuristic to prune the set of candidate second paths. Based on Theorem 1 and 2 (§3), paths with low predictability yield high replication gain. Accordingly, ReGain estimates the relative predictability of each path using the delay distribution. ReGain then only considers paths whose predictability $\varepsilon < 0.7$ as a candidate second path (which yields a replication gain of at least 1.1 using Theorem 2).

4.3 Load-aware adaptation

ReGain as explained so far ignored the impact of load. However, under high load conditions, replication yields little benefit and can instead severely hurt performance (even when limited to two paths). So, ReGain uses a load-aware replication to switch from replication to (single-path) forwarding. Under extremely high load, even single-path path forwarding may not achieve the network capacity required to support the load. So, ReGain uses a load-aware forwarding to switch from single-path forwarding to multi-path forwarding. We explain both mechanisms below.

4.3.1 Load-aware replication

Each source node tracks the actual delay of replicating packets along two paths by having the destination send an acknowledgment for the earliest delivered copy of each packet. If the actual two-path delay exceeds twice 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 twice the estimated two-path delay. Our experiments in §5 suggest that this simple heuristic is sufficiently responsive to high load enabling ReGain to achieve performance comparable to state-of-the-art forwarding protocols.

4.3.2 Load-aware forwarding

A node uses load-aware forwarding to switch from single-path forwarding to multi-path forwarding under high network load. In contrast to replication that sends copies of a packet on multiple path, multi-path forwarding sends each packet along a single path but stripes packets along different paths. To appreciate when multi-path forwarding is fundamentally needed, consider a node pair connected by a low-delay, low-bandwidth path and a high-delay, high-bandwidth path. When the network load is low, the best strategy is to send all packets on the low-delay, low-bandwidth path. However, when the load exceeds the bandwidth of the low-delay path, a node must distribute some load on the high-delay, high-bandwidth path.

As above, each source node tracks the actual delay of packets. If the average actual delay on the shortest path is more than twice its expected delay, the source node considers the shortest path as congested. At this point, ReGain sends probe packets and measures the actual delay on the path with the second-lowest expected delay, and starts forwarding packets on both the shortest path and this path if the

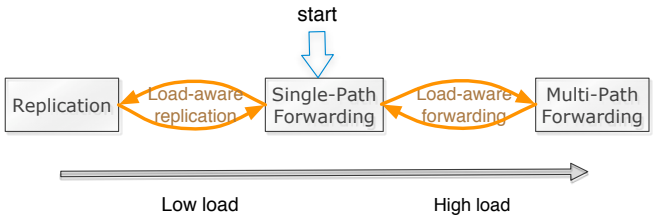


Figure 7. ReGain's state machine

latter's actual delay is less than half of the former's actual delay. The packets are striped along the two paths such that the load on each path is inversely proportional to their actual delays. A node switches back from two-path forwarding to single-path forwarding if the actual delay on the second-lowest delay path exceeds half of the actual delay on the shortest path.

Figure 7 shows ReGain's state machine that puts together all three components : replication, single-path forwarding, and multi-path forwarding.

4.4 Implementation details

ReGain 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), ReGain 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 (as in meshes), ReGain 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 and reduce overhead. Finally, to keep routing overhead low, ReGain nodes propagate LSAs only if the expected delay or any of the decile values change by more than 10%.

5 EVALUATION

We evaluate ReGain using a combination of trace-driven evaluations as well as deployment-based experiments. For a broad evaluation, we perform experiments over a range of parameters as listed below:

1. Diverse networks: We conduct experiments on networks that exhibit spatial (§5.2.1) and temporal (§5.2.2) variations to evaluate the performance of ReGain under different connectivity scenarios.
2. Homogeneous networks: We quantify the performance of ReGain in homogenous network conditions on the ends of the connectivity spectrum; i.e. sparsely-connected DTNs (§5.3.1) and well-connected meshes (§5.3.2) and show that ReGain outperforms the state-of-the-art replication-only and forwarding-only protocols in each of the environments.
3. Varying load: We show that ReGain adapts to varying load and outperforms state-of-the-art routing protocols under different load conditions. We also show that the second best performing protocol varies with changing load, motivating the need for a self-adapting protocol.

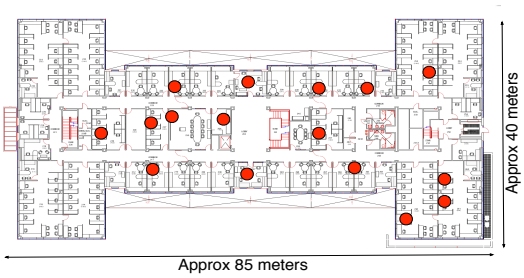


Figure 8. The Mesh testbed with dots representing nodes.

Finally, we show how different components of ReGain individually contribute to its performance, namely, *(i)* Using delay distributions rather than mean, *(ii)* Load aware replication and forwarding, and *(iii)* Using two-path replication.

5.1 Experimental setup

We evaluate ReGain using a deployed prototype on a wireless mesh testbed as well as trace-driven experiments using traces collected from real network testbeds exhibiting temporally and spatially diverse connectivity. We refer to these testbeds as Mesh, Haggle and DieselNet. The deployment as well as trace-driven evaluations use 1.5KB packets. The experiments are of one hour duration on Mesh and one day duration on Haggle and DieselNet. Each data point is averaged over five runs.

5.1.1 Mesh Deployment:

We deploy ReGain on the Mesh testbed. 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 Apple Mac Mini computer running Linux 2.6 with 802.11b Atheros/MadWiFi wireless card. The cards are configured to send at 5.5Mbps with a transmit power of 15dBm. RTS/CTS is turned off, and the cards are set to ad hoc mode. The ReGain prototype runs as a user-space daemon. Our implementation sends and receives raw 802.11 frames from the wireless device using the libnet interface[3]. Path lengths vary between 1 and 5 hops.

5.1.2 Trace-driven evaluation:

We conduct trace-driven evaluations over Haggle and DieselNet using the QualNet simulator[5]. Haggle[20] is a sparsely-connected 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 one contact between them.

DieselNet [2] is a hybrid mesh-DTN testbed with spatially diverse connectivity. The testbed consists of 20 buses in a 150 sq.mile area forming a sparse network. The testbed has 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. Unlike Haggle and Mesh the DieselNet testbed has two kinds of nodes: stationary APs and mobile buses.

We *a priori* infer the GPS coordinates of the APs in the mesh clusters. We log the GPS coordinate of each bus in the format (i, t, lat, lon) denoting that node i is at GPS location (lat, lon) at time t. 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), 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.

5.1.3 Alternate routing protocols

We compare the performance of ReGain to the following forwarding and replication based protocols:

1. Replication-based: *(i)* RAPID [12], 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 random probability of 0.5.
2. Forwarding-based: *(i)* DTLSR [23], a DTN routing protocol based on link-state forwarding that uses delays as the link metric, *(ii)* AODV [33], a mesh routing protocol based on distance-vector forwarding that uses hop count as the link metric, and *(iii)* OLSR [4], a mesh routing protocol based on link-state forwarding that uses ETX [22] as the link metric.

We implement RAPID, Random and DTLSR in our Mesh testbed and in the QualNet simulator, and modify existing implementations of AODV [1] and OLSR [4].

While evaluating the above protocols in environments they are not designed for, we make some straightforward changes. AODV and OLSR assume the existence of a contemporaneous end-to-end path. To adapt AODV and OLSR to sparsely-connected networks, we set a high timeout values allowing them to buffer 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 [24] (specifically in Eq. 8 in [12]) 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 in contrast to ReGain.

To reduce clutter, the lines for Random and AODV in the graphs are deferred to the Appendix, as we find that Random consistently performs worse than ReGain and RAPID, and AODV consistently performs close to OLSR.

5.1.4 Load and metrics

We generate 30 concurrent flows between randomly chosen source-destination pairs. In the DieselNet 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 load in the network.

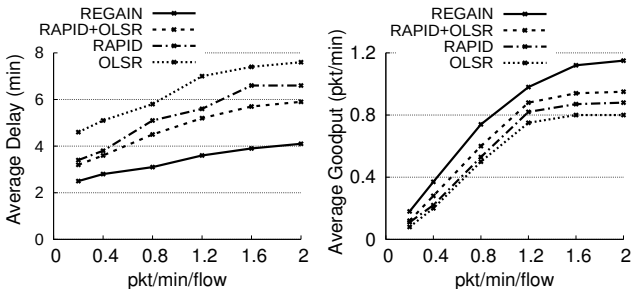
We quantify the performance of the protocols in terms of *delay* and *goodput*. Delay of a flow is the average delay of packets in a flow, and the delay of undelivered packet is the time the packet spent in the network. Goodput is the average rate of packet reception over the experiment period. The goodput metric is similar to the capacity of the network.

5.2 Diverse connectivity

We conduct experiments in diverse network environments to show that ReGain adapts well to different connectivity scenarios.

5.2.1 DieselNet: Spatial diversity

We first evaluate ReGain on DieselNet, a hybrid mesh-DTN network that has spatially diverse connectivity (Figure 2(b)). As no protocol is explicitly designed for this hybrid mesh-DTN environment exists, we compare ReGain to three protocol setups: (a) RAPID on the buses and OLSR on the mesh clusters (i.e., replication on DTN and forwarding on mesh respectively); (b) RAPID on both buses and mesh clusters; (c) OLSR on both busses and mesh clusters. We mark them as RAPID+OLSR, RAPID, and OLSR respectively. Recall that RAPID is well-suited for the DTNs while OLSR is well suited for mesh environments.



(a) Delay (b) Goodput

Figure 9. Spatial diversity on DieselNet. ReGain improves delay by 1.45× and goodput by 1.25×.

Figure 9 shows the average delay and goodput across flows on DieselNet. ReGain outperforms all other protocols at all loads and, in particular, improves delay by 1.45× and goodput by 1.25× over the second best protocol, RAPID+OLSR. The gains increase with load.

OLSR, a forwarding protocol routes most packets via direct contact and fails to utilize connectivity enabled by bus-bus DTN connectivity. ReGain, RAPID+OLSR and RAPID all leverage DTN connectivity and outperform OLSR. However, ReGain adapts replication better to the spatial diversity compared to the the latter two protocols.

We performed a paired t-test[18] to understand if the difference between the different protocols is statistically significant. We compare the average value across flows using ReGain to the average value across flows using the second best protocol, RAPID+OLSR. We found that the p-values were less than 0.001, indicating that the difference between the average values is statistically significant. The p-values are in the same ballpark for all of the experiments comparing averages in this paper, so we omit stating them explicitly.

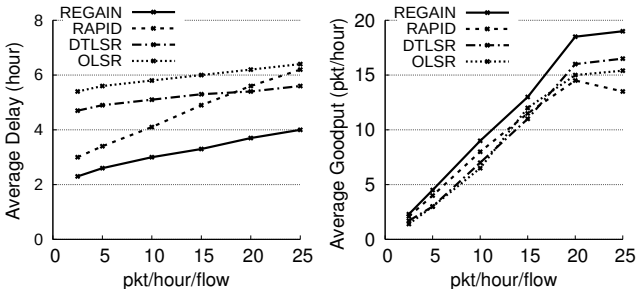
5.2.2 Haggle: Temporal diversity

As Haggle exhibits temporal diversity in connectivity (Figure 2(a)), we use this trace to evaluate ReGain's performance in scenarios with varying connectivity over time. Figure 10 shows that ReGain improves delay by 1.35× and improves goodput by 1.15× over the second best protocol. Furthermore, it consistently achieves the best delay and goodput even as the load increases.

Figure 10 further shows that at low load, RAPID, a replication-based protocol outperforms OLSR and DTLSR, both forwarding-based protocols. However, when the load increases, OLSR and DTLSR outperform RAPID, showing that replication hurts performance under high load.

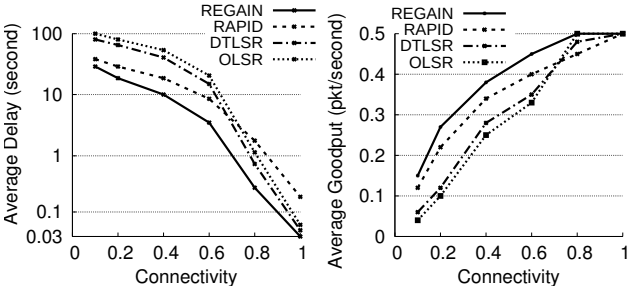
5.2.3 Mesh: Emulating diverse connectivity

To further stress-test ReGain's performance in diverse connectivity scenarios, we use the mesh testbed to emulate networks with changing connectivity. We define *connectivity* as the fraction of connected node pairs in the network graph.



(a) Delay (b) Goodput

Figure 10. Temporal diversity on Haggle. ReGain improves delay by 1.35× and goodput by 1.15×.



(a) Delay (b) Goodput

Figure 11. Emulating diverse connectivity on Mesh. The y-axis of (a) is in logscale. ReGain improves delay by 2× and goodput by 1.25× across varying connectivity.

We bring the nodes up and down, fixing the down time to one minute. We vary the node up time to get different levels of connectivity. The load is fixed to 0.5 pkt/second/flow to induce moderate load, while the connectivity increases from 0.1 to 1, holding each connectivity level for one hour.

Figure 11 shows the average delay and goodput across flows at time-varying connectivity. ReGain has up to 2× delay improvement and 1.25× goodput improvement over existing protocols.

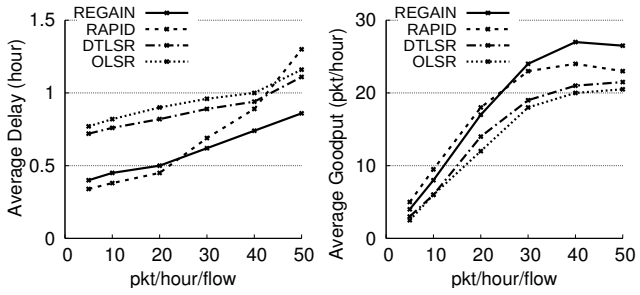
It is noteworthy that the other protocols work well in specific connectivity regimes but perform poorly in others. For example, Figure 11(a) shows that when connectivity is less than 0.7, RAPID outperforms forwarding-based OLSR and DTLSR. However, as connectivity improves, both OLSR and DTLSR outperform RAPID. On the other hand, ReGain outperforms all of the compared protocols at all connectivity levels.

5.3 Homogeneous network conditions

We conduct experiments in sparsely-connected and well-connected networks and show that in these environments, ReGain, a self-adapting protocol, outperforms protocols that are specifically designed for the environment.

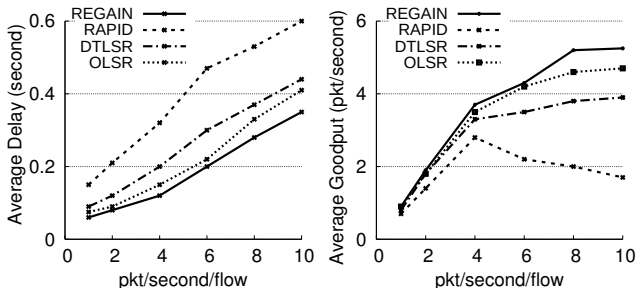
5.3.1 Sparsely-connected networks

We use the DieselNet testbed as our sparsely-connected network but remove the mesh clusters, and mark the testbed as DieselNet-DTN. Figure 12 shows that under low load, ReGain performs comparably to RAPID, a replication-based protocol specifically designed for the sparsely-connected DTN environment. However, ReGain achieves 1.40× delay and 1.30× goodput improvement under high load. ReGain's load-aware replication enables the better performance than



(a) Delay (b) Goodput

Figure 12. DieselNet-DTN: Sparsely-connected network. ReGain has comparable performance to RAPID, a DTN routing protocol, under low load. ReGain outperforms RAPID under high load.



(a) Delay (b) Goodput

Figure 13. Mesh: Well-connected network. ReGain has comparable performance to OLSR, a protocol designed for mesh networks, under low load. ReGain performs slightly better than OLSR under high load.

RAPID under high load.

DTLSR and OLSR perform much worse compared to both RAPID and ReGain; DTLSR does not use replication, and OLSR is not designed for highly unpredictable and disconnected topologies.

5.3.2 Well-connected networks

We use our mesh deployment as the well-connected network. Figure 13 shows that under low load, ReGain has similar performance to OLSR, a forwarding-based routing protocol designed specifically for well-connected meshes. At high loads, ReGain has a 1.16× delay and 1.11× goodput improvement over OLSR. ReGain employs load-aware forwarding to distribute load across multiple paths under high load, which results in the slightly better performance compared to OLSR.

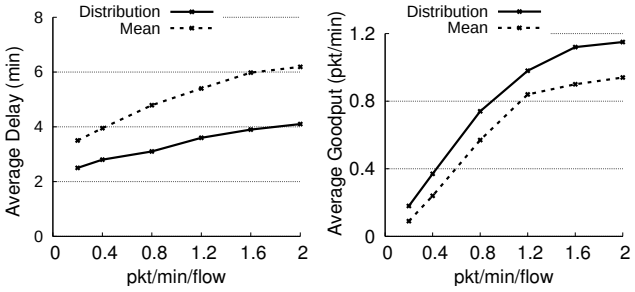
As expected, ReGain has up to 1.7× delay and 3× goodput improvement over RAPID, since RAPID is not designed for mesh environments.

5.4 Component of ReGain

In this section, we evaluate four aspects of ReGain's design: (a) using delay distribution rather than mean values for path selection, (b) load-aware adaptation, (c) using two-path replication, (d) routing overhead.

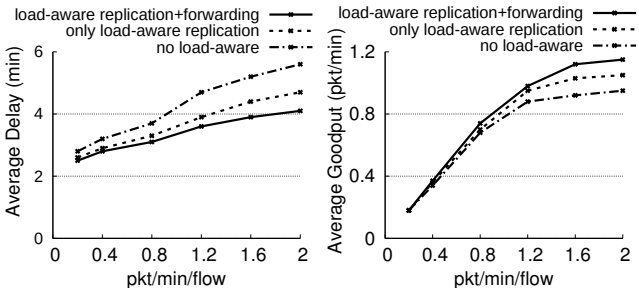
5.4.1 Distribution vs. mean

We modify ReGain to select the replication path using mean delay rather than delay distribution as follows: the first path is selected as the minimum expected delay path as before, and the second path is the one with the second



(a) Delay (b) Goodput

Figure 14. DieselNet: Using delay distribution versus mean delays for path selection. Using delay distribution improves delay performance by 1.5× delay and goodput performance by 1.22× over using mean delay.



(a) Delay (b) Goodput

Figure 15. DieselNet: Performance of load-aware replication and load-aware forwarding components. Load-aware replication improves delay by 1.35× and goodput by 1.21×, and load-aware forwarding further improves delay by 1.14× and goodput by 1.1×.

minimum expected delay. We mark the performance of this modified protocol as *Mean*. Recall that the default ReGain chooses the secondary path as the path whose delay distribution minimizes the combined two-path delay metric (Equation 6). We mark the performance of ReGain that uses delay distribution as *Distribution*.

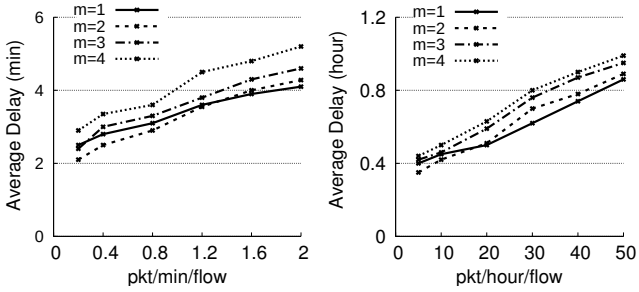
Figure 14 shows that selecting paths using distribution has 1.50× delay and 1.22× goodput improvement over using mean delay in DieselNet. The result suggests that distributions, unlike mean values, capture the unpredictability of path delays better, and in turn enables higher replication gain. Experiments on the other testbeds are consistent with this result and are deferred to the Appendix.

5.4.2 Load aware adaptation

We evaluate the load-aware adaptation component of ReGain by comparing it against the following variants: ReGain without any load-aware adaptation and ReGain with only load-aware replication but no load-aware forwarding. Figure 15 shows the under high network load, load aware replication improves delay by 1.35× and goodput by 1.21×, and load aware forwarding further improves delay by 1.14× and goodput by 1.1×.

5.4.3 Using two-path replication

The model in §3 suggests that replication gain increases as more paths are used for replication, when the effect of network load is ignored. Here, we experimentally analyze how ReGain's benefit changes as the number of paths used



(a) DieselNet (b) DieselNet-DTN

Figure 16. Average delay when m=1,2,3,4 secondary paths are used for replication in ReGain. m=1 (default ReGain) gets most of the replication benefit.

for replication increases, by taking into account load effects. We set m, the number of the second paths for replication, to be 1, 2, 3, 4 (note that the default ReGain uses $m = 1$). We assume that when a node decides to revert from replication to forwarding due to high load (Section 4.3), it does so on all the m secondary paths.

Figure 16 shows that two-path replication (m=1) is a sweet spot, and gets most of the benefits. Using more paths yields negligible benefit under light load but hurts as load increases. This reflects the protocol design tradeoff between leveraging replication benefit and adapting to load. We defer results from other testbeds to the Appendix.

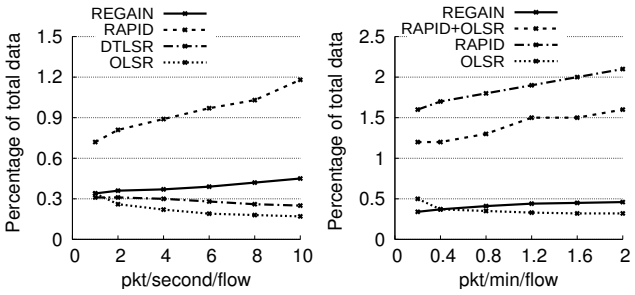
5.4.4 Routing overhead

Although the previous experiments implicitly incorporated the effect of routing overhead for all protocols, we explicitly evaluate it. Different protocols incur different overheads, depending on what information they exchange with their neighbors *(i)* ReGain: 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. We compute the routing overhead as the percentage of the total traffic data.

Figure 17 shows the routing overhead of different protocols on Mesh and DieselNet. ReGain has less than 0.5% routing overhead across both testbeds, though it's overhead is slightly higher than DTLSR and OLSR. The previous experiments suggest that the benefits of ReGain justifies this increase in overhead. The routing overhead of DTLSR and OLSR decreases as load increases because they do not incur per-packet overhead. RAPID incurs the highest overhead because it disseminates packet replica locations in addition to information about past node contacts. We observe similar results in other testbeds and present the results in the Appendix.

5.5 Other results

We conducted more experiments than those presented above to further investigate ReGain's performance, but omitted them due to lack of space. We briefly summarize them here. The experiments evaluate delay and goodput over spatially and temporally diverse connectivity networks as well as mesh and DTNs for a workload consisting of (1) a single flow between a randomly chosen node pair, (2) 30 concurrent flows with (nonuniform) powerlaw-distributed sending rates, (3) flows between all pairs of nodes with uniform



(a) Mesh (b) DieselNet

Figure 17. Routing overhead as a percentage of total traffic data: ReGain has less than 0.5% routing overhead.

sending rates, and (4) flows between all pairs of nodes with powerlaw-distributed sending rates.

In the single flow case, ReGain achieve even higher delay and goodput gains over the other protocols than in the multiple flow cases. For example, ReGain has 2.5× delay and 1.8× goodput improvement over the other protocols in the emulated time-varying connectivity Mesh. This is due to better utilization of multiple paths. The different variants of the concurrent flow experiments yield qualitatively similar conclusions to the experiments presented in this paper.

6 RELATED WORK

Our work differs from prior work primarily in its goal—to design and implement a simple routing protocol that works well across diverse wireless network environments—that to our knowledge has not been done before. The design of ReGain liberally borrows insights from a large body of prior work in wireless routing as discussed below.

Replication routing, also referred to as epidemic routing [40], multi-copy routing [36], controlled flooding [26], etc. in the literature, has been well studied over the last decade [26, 31, 38, 12, 34, 12, 16, 37, 32, 44, 40, 36]. Most existing replication routing protocols are primarily designed for highly disconnected networks where pairs of nodes meet each other infrequently. In comparison, ReGain is designed to work well across a broad spectrum of connectivity all the way from DTNs to well connected mesh networks. 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 investigation of this question forms an important focus of our work, and is addressed in §2.

Many DTN routing protocols make explicit or implicit assumptions about the inter-meeting of nodes. For example, RAPID[12] assumes that the meeting times are exponentially distributed; Thrasyvoulos et al. [38, 37] assume a random-walk mobility model; Jain et al. [29] use a Bernoulli and Gaussian path delivery model; Shin et al. [35] apply Levy walk pattern to optimize DTN routing strategy. These assumptions can restrict their applicability to other environments. For example, consider exponentially distributed meeting times, an assumption central to RAPID's design. Although this assumption is unlikely to hurt performance in DTNs where the inter-contact times follow a different distribution, it breaks down in a mesh environment as it implies that making K replicas of a packet reduce the delay by a factor of K. However, replicating packets in a mesh will only exacerbate delays by overwhelming the network. 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, ReGain explicitly measures the distribution of path delays and the nature of this distribution to control replication, and is applicable to broad spectrum of mobility or disconnection patterns. This idea is similar in spirit to Francois et al. [25] 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 work is foremost a design and implementation effort and targets diverse wireless network environments.

Our model formally shows that replication improves delay significantly if and only if path delays are highly unpredictable. Source coding techniques such as erasure coding can further reduce delay as shown by Wang et al. [41]. Our network model and use of order statistics to model replication gain is similar to Wang et al. [41], however they focus on quantifying the added benefit of erasure coding over simple replication for specific distributions, whereas our result does not assume a specific distribution. Network coding techniques can significantly improve throughput under unpredictable network conditions, but have limited benefit for delay. Zhang et al. [43] show that random linear coding performs worse than replication schemes with multiple flows as the destination must wait until all independent blocks are received to decode the whole packet.

Existing replication protocols use several schemes to control replication such as probabilistic replication[26, 31], utility replication [38, 12], prioritizing transmit order[34, 12], acknowledgments to remove useless packet [16], and explicitly bounding replicas [37, 38]. ReGain's design shares many of these ideas including bounding the number of replicas to two, but additionally compares actual packet delays to estimated packet delays to judge the effectiveness of replication and turn it off as needed. Using measured delay distributions allows ReGain to obtain more accurate estimates of packet delays and be more responsive to load or interference.

Mesh, MANET, and DTN routing protocols use a variety of link metrics such as hop count, ETX [22], ETT [24], inter-contact time [16], expected delay [12, 23] etc. ReGain's use of expected delays is similar in spirit to ETT in well-connected networks and inter-contact times in highly disconnected networks. Unlike protocols such as RAPID or DTLSR that attempt to estimate expected delays accounting for the number of buffered packets at a node, ReGain accounts for load based on the difference between the actual and estimated delays.

Some existing forwarding-based routing protocols use multipath forwarding to balance load [39, 42, 27, 19, 17]. These protocols measure path quality and load to explicitly optimize for delay [27, 19], goodput [42] or reliability [17] metric. ReGain shares similar goal to these works and explicitly optimizes for delay but only uses multipath forwarding when it helps reduce delay.

7 CONCLUSIONS

Wireless routing has seen an enormous body of research in recent times, but is becoming increasingly compartmentalized. Researchers and practitioners continue to develop sophisticated routing protocols that are designed and optimized for specific network environments such as meshes, MANETs, and DTNs, but perform poorly or break down in other environments. This state of affairs makes the case for a simple routing protocol that works well across diverse wireless network environments.

To address this challenge, we design and implement ReGain, a routing protocol that self-adapts replication to changing network conditions and load. The key insight behind ReGain's generality is to not embed specific assumptions about node mobility, but instead infer and leverage it in a fine-grained manner. We rigorously evaluate ReGain using a combination of prototype deployment, simulation, and emulation experiments over a broad spectrum of network environments and show that it achieves significantly better performance than state-of-the-art protocols in networks with diverse connectivity characteristics.

8 References

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

[2] Dieselnet. http://prisms.cs.umass.edu/dome/umassdieselnet.

[3] Libnet. http://libnet.sourceforge.net.

[4] Optimized link state routing protocol. http://www.olsr.org/.

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

[6] San francisco bawug. http://www.bawug.org/.

[7] Seattle wireless. http://seattlewireless.net/.

[8] Southampton open wireless network. http://www.sown.org.uk/.

[9] Variance. http://en.wikipedia.org/wiki/Variance.

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

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

[12] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *Sigcomm*, 2007.

[13] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Enabling interactive applications in hybrid networks. In *Mobicom*, 2008.

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

[15] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive wifi connectivity for moving vehicles. In *SIGCOMM*, 2008.

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

[17] P. M. Carthy and D. Grigoras. Multipath associativity based routing. In *WONS*, 2005.

[18] G. Casella and R. Berger. *Statistical Inference*. Duxbury, second edition, 2002.

[19] L. D. Cha M. Split-n-save multiplexing in wireless ad hoc routing. In *infocom workshop*, 2005.

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

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

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

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

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

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

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

[27] X. Huang and Y. Fang. End-to-end delay differentiation by prioritized multipath routing in wireless sensor networks. In *MILCOM*, 2005.

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

[29] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in a delay tolerant network. In *SIGCOMM '05*.

[30] 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.

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

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

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

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

[35] M. Shin, S. Hong, and I. Rhee. Dtn routing strategies

using optimal search patterns. In *CHANTS '08*.

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

[37] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN*, 2005.

[38] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *PERCOMW*, 2007.

[39] M. Tarique, K. E. Tepe, S. Adibi, and S. Erfani. Survey of multipath routing protocols for mobile ad hoc networks. In *Journal of Network and Computer Applications*, 2009.

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

[41] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. In *WDTN*, 2005.

[42] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. Effects of multipath routing on tcp performance in ad hoc networks. In *GLOBECOM'04*.

[43] X. Zhang, G. Neglia, J. Kurose, and D. Towsley. On the benets of random linear coding for unicast applications in disruption tolerant networks. In *IEEE Second Workshop on Network Coding, Theory, and Applications*, 2006.

[44] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *Communications Surveys & Tutorials, IEEE*, 2006.

APPENDIX

Derivation of Eq. 3

PROOF. Let m be the path with minimum expected delay and $f_{X_m}(x)$ denote the PDF (probability density function) of X_m, then

$$\begin{aligned}\mu = E[X_m] &= \int_0^{+\infty} x f_{X_m}(x)dx = \int_0^{+\infty} (\int_0^x dy)\ f_{X_m}(x)\ dx \\ &= \int_0^{+\infty}\int_0^x f_{X_m}(x)\ dy\ dx = \int_0^{+\infty}\left(\int_y^{+\infty} f_{X_m}(x)dx\right)\ dy \\ &= \int_0^{+\infty} P[X_m > y]dy\end{aligned} \tag{7}$$

Let $X_{(1)} = \min\{X_1, X_2, ..., X_n\}$, using similar steps used above, we have

$$\mu_{(1)} = E[X_{(1)}] = \int_0^{+\infty} P[X_{(1)} > x]dx = \int_0^{+\infty}\prod_{i=1}^{n} P[X_i > x]dx \tag{8}$$

And replication gain is

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

□

Proof of Theorem 1

PROOF. (a) From Eq. 8, we know that for n i.i.d random variables (with each denoted as X and $n \geq 2$)

$$\begin{aligned}\mu_{(1)} &= \int_0^{+\infty} (P[X > x])^n dx \\ &= \int_0^{\varepsilon\mu} (P[X > x])^n dx + \int_{\varepsilon\mu}^{+\infty} (P[X > x])^n dx \\ &\leq \int_0^{\varepsilon\mu} (P[X > x])^n dx + P[X > \varepsilon\mu]\int_{\varepsilon\mu}^{+\infty} P[X > x]dx \\ &\leq \int_0^{\varepsilon\mu} P[X > x]dx + (1 - P[X \leq \varepsilon\mu])(\mu - \int_0^{\varepsilon\mu} P[X > x]dx)\end{aligned}$$

We know that X has predictability ε, i.e., $P[X \leq \varepsilon\mu] \geq 1-\varepsilon$, thus

$$1 - P[X \leq \varepsilon\mu] \leq \varepsilon$$

We further let $a\mu = \int_0^{\varepsilon\mu} P[X > x]dx$, then

$$\mu_{(1)} \leq a\mu + \varepsilon(\mu - a\mu) = (a + \varepsilon - \varepsilon a)\mu = [\varepsilon + a(1-\varepsilon)]\mu$$

Since $a\mu \leq \int_0^{\varepsilon\mu} dx = \varepsilon\mu$, thus

$$\mu_{(1)} \leq [\varepsilon + \varepsilon(1-\varepsilon)]\mu = [1 - (1-\varepsilon)^2]\mu, \qquad \frac{\mu}{\mu_{(1)}} \geq \frac{1}{1-(1-\varepsilon)^2}$$

Thus replication gain is at least $\frac{1}{1-(1-\varepsilon)^2}$.

(b) Let $\varepsilon = G^{-\frac{1}{n+1}}$, we want to prove that $P[X \leq \varepsilon\mu] \geq 1-\varepsilon$.

Using Eq. 8:

$$\mu_{(1)} = \int_0^{+\infty} (P[X > x])^n dx \geq \int_0^{\varepsilon\mu} (P[X > x])^n dx \geq \varepsilon\mu(P[X > \varepsilon\mu])^n$$

$$\frac{\mu}{\mu_{(1)}} = G \leq \frac{1}{\varepsilon(P[X > \varepsilon\mu])^n}$$

Since $\varepsilon = G^{-\frac{1}{n+1}}$, i.e. $G = 1/\varepsilon^{n+1}$, then:

$$\frac{1}{\varepsilon^{n+1}} \leq \frac{1}{\varepsilon(P[X > \varepsilon\mu])^n}$$

Thus $P[X > \varepsilon\mu] \leq \varepsilon$, $P[X \leq \varepsilon\mu] \geq 1-\varepsilon$ for $\varepsilon = G^{-\frac{1}{n+1}}$. Since there can be smaller ε value that satisfies $P[X \leq \varepsilon\mu] \geq 1-\varepsilon$ and we are unable to find the smallest value of ε, thus X has predictability at most $G^{-\frac{1}{n+1}}$. □

Proof of Theorem 2

PROOF. Let $\mu = E[X_m]$

(a) From Eq. 8, we have

$$\begin{aligned}\mu_{(1)} &= \int_0^{+\infty} \prod_{j=1}^{n} P[X_j > x]dx \\ &= \int_0^{\delta\mu} \prod_{j=1}^{n} P[X_j > x]dx + \int_{\delta\mu}^{+\infty} \prod_{j=1}^{n} P[X_j > x]dx \\ &\leq \int_0^{\delta\mu} P[X_m > x]dx + P[X_i > \delta\mu] \int_{\delta\mu}^{+\infty} P[X_m > x]dx\end{aligned}$$

Since X_i has relative predictability δ, thus $P[X_i > \delta\mu] \leq \delta$. Let $a\mu = \int_0^{\delta\mu} P[X_m > x]dx$ and use imilar steps as the proof of Theorem 1(a), we have

$$\mu_{(1)} \leq a\mu + \delta(\mu - a\mu) \leq (1-(1-\delta)^2)\mu, \qquad \frac{\mu}{\mu_{(1)}} \geq \frac{1}{1-(1-\delta)^2}$$

i.e., replication gain is at least $\frac{1}{1-(1-\delta)^2}$.

(b) Let $\delta = G^{-\frac{1}{n+1}}$, we want to prove $P[X_i \leq \delta\mu]] \geq 1-\delta$. Using Eq. 8:

$$\begin{aligned}\mu_{(1)} &= \int_0^{+\infty} \prod_{j=1}^{n} P[X_j > x]dx \geq \int_0^{\delta\mu} \prod_{j=1}^{n} P[X_j > x]dx \\ &\geq \delta\mu \prod_{j=1}^{n} P[X_j > \delta\mu]\end{aligned}$$

Since $P[X_i \leq \delta\mu] = \max_{1\leq j\leq n} P[X_j \leq \delta\mu]$, i.e., $P[X_i > \delta\mu] = \min_{1\leq j\leq n} P[X_j > \delta\mu]$, then

$$\begin{aligned}\mu_{(1)} &\geq \delta\mu \, (P[X_i > \delta\mu])^n \\ \frac{\mu}{\mu_{(1)}} = G &\leq \frac{1}{\delta \, (P[X_i > \delta\mu])^n}\end{aligned}$$

Since $\delta = G^{-\frac{1}{n+1}}$, i.e. $G = 1/\delta^{n+1}$, then:

$$\frac{1}{\delta^{n+1}} \leq \frac{1}{\delta \, (P[X_i > \delta\mu])^n}$$

Thus $P[X_i > \delta\mu] \leq \delta$, $P[X_i \leq \delta\mu] \geq 1-\delta$ for $\delta = G^{-\frac{1}{n+1}}$, i.e., X_i has predictability at most $G^{-\frac{1}{n+1}}$. □

Derivation of Eq. 4

PROOF. Let X denote the link availability delay and Y denote the inter-contact time. By definition, X represents the time until the next contact sampled at a uniformly random point in time, it also represents the packets' waiting time to be transferred after they are generated.

Let Y' denote the "special" inter-contact interval in which a packet arrives. Note that Y' is not distributed the same as Y because a packet is more likely to arrive in a longer interval than a shorter one. So the probability that an interval of length y is chosen by a packet should be proportional to the length (y) as well as the relative occurrence of such intervals ($f_Y(y)dy$). We can write

$$P(y < Y' \leq y+dy) = f_{Y'}(y)dy = Kyf_Y(y)dy \qquad (9)$$

Integrating both sides of Eq. 9, we have

$$K = 1/E(Y), \text{ so } f_{Y'}(y) = \frac{yf_Y(y)}{E(Y)}$$

If we are told that $Y' = y$, then the probability that X does not exceed the value x is given by

$$P(X \leq x | Y' = y) = \frac{x}{y}$$

Thus we may write down the joint density of Y' and X as

$$\begin{aligned}P(x < X \leq x+dx, \, y < Y' \leq y+dy) &= (\frac{dx}{y})(\frac{yf_Y(y)dy}{E(Y)}) \\ &= \frac{f_Y(y) \, dx \, dy}{E(Y)} \quad (0 \leq x \leq y)\end{aligned}$$

Integrating over y we botain $f_X(x)$, namely,

$$f_X(x)dx = \int_{y=x}^{+\infty} \frac{f_Y(y) \, dx \, dy}{E(Y)}, \;\; f_X(x) = \frac{1-F_Y(x)}{E(Y)}$$

Thus

$$F_X(x) = \int_0^x f_X(y)dy = \frac{1}{E(Y)} \int_0^x (1-F_Y(y))dy$$

□

Derivation of Eq. 5

PROOF. First, let's compute $E(X)$:

$$\begin{aligned}E(X) &= \int_0^{+\infty} xf_X(x)dx = \int_0^{+\infty} x\frac{1-F_Y(x)}{E(Y)}dx \\ &= \frac{1}{E(Y)} \int_0^{+\infty} xP(Y > x)dx\end{aligned}$$

So we have

$$E(X) * E(Y) = \int_0^{+\infty} xP(Y > x)dx \qquad (10)$$

We also know that

$$E(Y) = \int_0^{+\infty} P(Y > y)dy \qquad (11)$$

Finally, we can express variance[9] of Y using its CCDF:

$$Var(Y) = 2\int_0^{+\infty} yP(Y > y)dy - (\int_0^{+\infty} P(Y > y)dy)^2 \qquad (12)$$

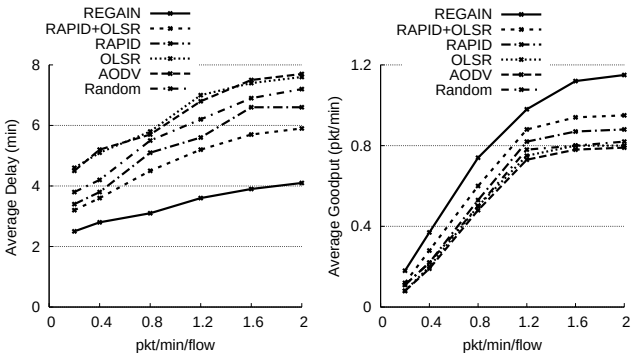
Using Eq. 10 and 11 to replace the rightsize of Eq. 12, we have

$$Var(Y) = 2E(X) * E(Y) - (E(Y))^2$$

So we have:

$$E(X) = \frac{E(Y)}{2} + \frac{Var(Y)}{2E(Y)} = \frac{E(Y)}{2} + \frac{\sigma^2(Y)}{2E(Y)}$$

□



(a) Delay (b) Goodput

Figure 18. Spatial diversity on DieselNet. ReGain improves delay by 1.45× and goodput by 1.25×.

EXTENSIVE EVALUATION OF REGAIN

Diverse connectivity

DieselNet: Spatial diversity

The results are shown in Figure 18. The experimental setup is the same as Figure 9.

Haggle: Temporal diversity

The results are shown in Figure 19. The experimental setup is the same as Figure 10.

Mesh: Emulating diverse connectivity

The results are shown in Figure 20. The experimental setup is the same as Figure 11.

Homogeneous network conditions

Sparsely-connected networks

The results are shown in Figure 21. The experimental setup is the same as Figure 12.

Well-connected networks

The results are shown in Figure 22. The experimental setup is the same as Figure 13.

Component of ReGain

Distribution vs. mean

The results are shown in Figure 23 and 24. The experimental setup is similar to Figure 14.

Load aware adaptation

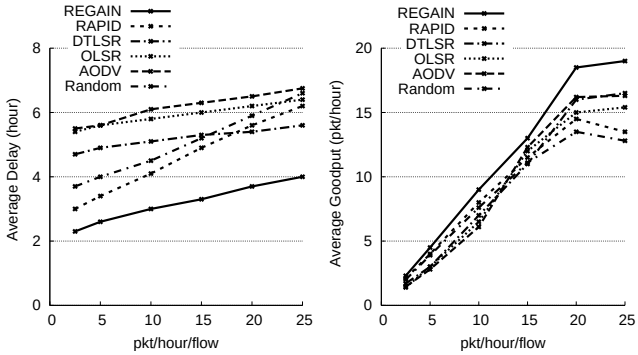
The results are shown in Figure 25 and 26. The experimental setup is similar to Figure 15.

Using two-path replication

The results are shown in Figure 27. The experimental setup is similar to Figure 16.

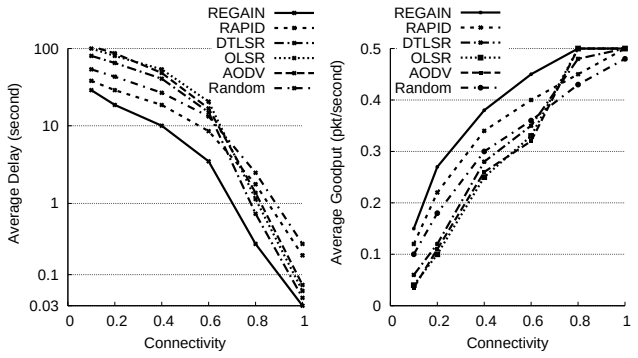
Routing overhead

The results are shown in Figure 28. The experimental setup is similar to Figure 17.



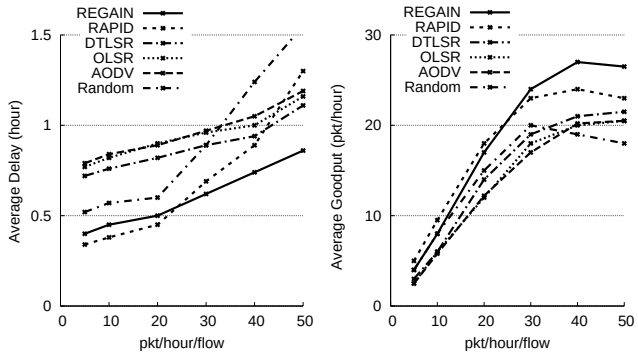
(a) Delay (b) Goodput

Figure 19. Temporal diversity on Haggle. ReGain improves delay by 1.35× and goodput by 1.15×.



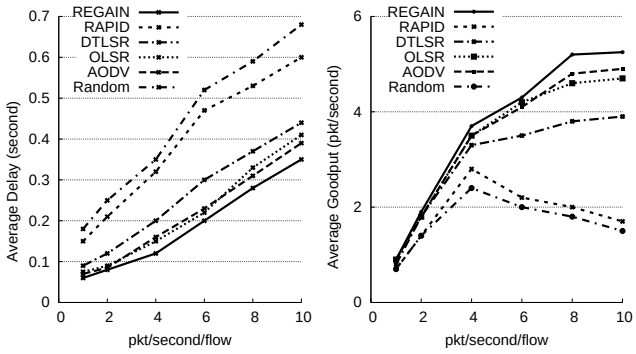
(a) Delay (b) Goodput

Figure 20. Emulating diverse connectivity on Mesh. The y-axis of (a) is in logscale. ReGain improves delay by 2× and goodput by 1.25× across varying connectivity.



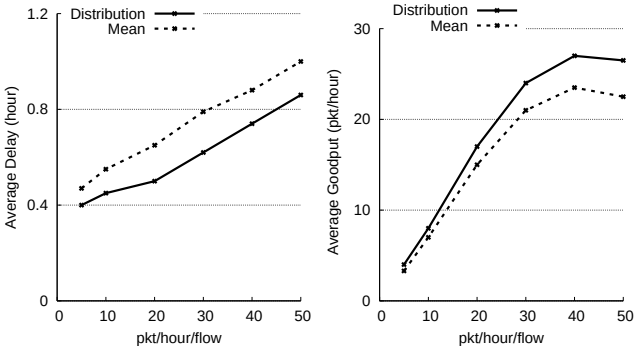
(a) Delay (b) Goodput

Figure 21. DieselNet-DTN: Sparsely-connected network. ReGain has comparable performance to RAPID, a DTN routing protocol, under low load. ReGain outperforms RAPID under high load.



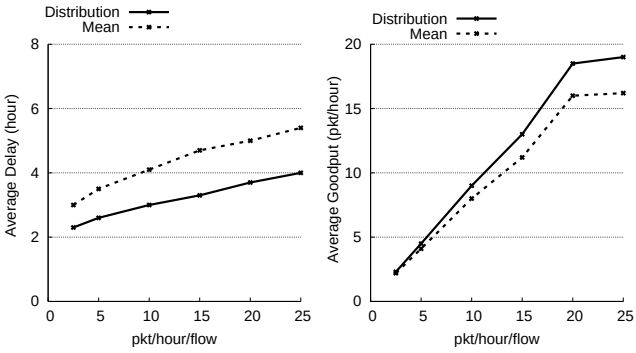
(a) Delay (b) Goodput

Figure 22. Mesh: Well-connected network. ReGain has comparable performance to OLSR, a protocol designed for mesh networks, under low load. ReGain performs slightly better than OLSR under high load.



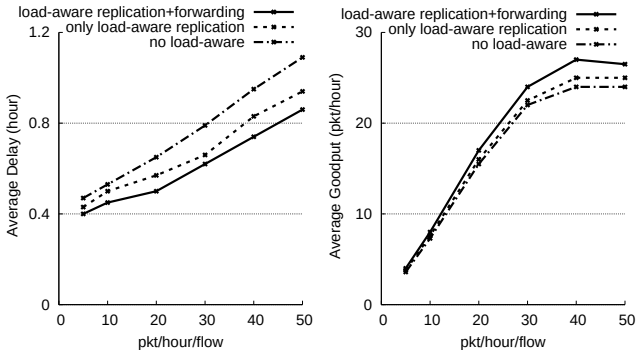
(a) Delay (b) Goodput

Figure 23. DieselNet-DTN: Using delay distribution versus mean delays for path selection. Using delay distribution improves delay performance by 1.3× delay and goodput performance by 1.17× over using mean delay.



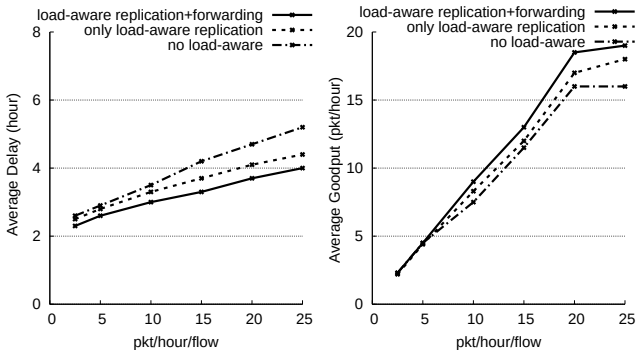
(a) Delay (b) Goodput

Figure 24. Haggle: Using delay distribution versus mean delays for path selection. Using delay distribution improves delay performance by 1.4× delay and goodput performance by 1.16× over using mean delay.



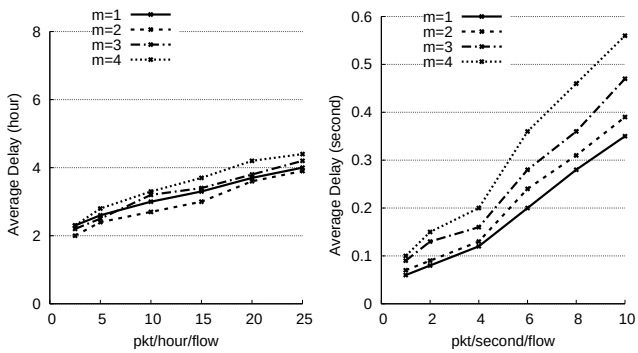
(a) Delay (b) Goodput

Figure 25. DieselNet-DTN: Performance of load-aware replication and load-aware forwarding components. Load-aware replication improves delay by 1.16× and goodput by 1.07×, and load-aware forwarding further improves delay by 1.09× and goodput by 1.08×.



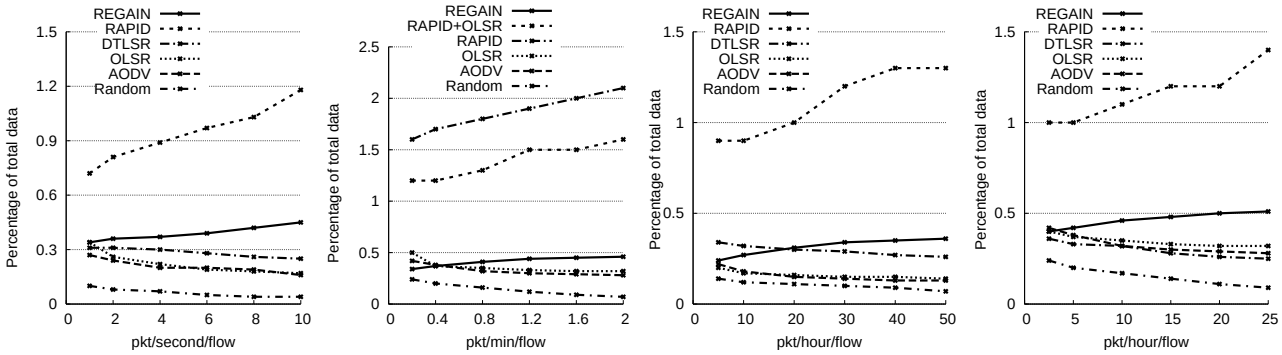
(a) Delay (b) Goodput

Figure 26. Haggle: Performance of load-aware replication and load-aware forwarding components. Load-aware replication improves delay by 1.18× and goodput by 1.12×, and load-aware forwarding further improves delay by 1.1× and goodput by 1.06×.



(a) Haggle (b) Mesh

Figure 27. Average delay when m=1,2,3,4 secondary paths are used for replication in ReGain. m=1 (default ReGain) gets most of the replication benefit.



(a) Mesh (b) DieselNet (c) DieselNet-DTN (d) Haggle

Figure 28. Routing overhead as a percentage of total traffic data: ReGain has less than 0.5% routing overhead.