# Listen First, Broadcast Later: Topology-Agnostic Forwarding under High Dynamics

Michael Meisel
UCLA
meisel@cs.ucla.edu

Vasileios Pappas
IBM Research
vpappas@us.ibm.com

Lixia Zhang
UCLA
lixia@cs.ucla.edu

## ABSTRACT

In this paper, we return to the drawing board to rethink the basic approach to multi-hop forwarding for highly dynamic wireless networks. The result is Listen First, Broadcast Later (LFBL), a surprisingly minimalist forwarding protocol. LFBL is topology-agnostic, that is, it has no knowledge of neighbors, routes, or next hops. LFBL receivers, not senders, make the forwarding decisions, and they only keep a small, fixed amount of state per active communication endpoint in order to do so. As a result, there is little state to go stale, and no pre-determined paths to be broken. Frequent topology changes do not adversely impact performance. LFBL uses exclusively broadcast communication for all packets, making it a more natural fit for a wireless medium and allowing for more flexibility in the selection of MAC layer protocols. In addition to physical mobility of nodes, LFBL also supports logical mobility of application-level identifiers. Under simulation, LFBL significantly outperforms AODV.

## 1. INTRODUCTION

Despite some fundamental differences, multi-hop wireless networks have typically adopted the Internet protocol stack, complete with traditional routing at the network layer and unicast support at the MAC layer. These networks simply swap out wired MAC and physical layers for their wireless equivalents, and traditional wired routing protocols for their reactive counterparts. It is a testament to the design of the Internet protocol stack that it can work in the foreign environment that is a multi-hop wireless network. However, we have returned to the drawing board to rethink how aligning routing at the network layer with the fundamentally broadcast nature of the wireless channel, and have found that this alignment can vastly improve performance.

In this paper, we present the Listen First, Broadcast Later (LFBL) protocol, a forwarding protocol for wireless networks that does not require a unicast primitive. The basic operation is simple. At each hop, the responsibility for forwarding decisions is placed squarely in the hands of the receiver, rather than the sender. After receiving a packet, a potential forwarder pauses to listen to the channel, waiting to see if a more optimal node forwards the packet first. Otherwise, it forwards the packet itself.

While the idea of receiver-based forwarding is not new in and of itself (e.g., [1]), its full potential in the context of highly dynamic multi-hop wireless networks has not yet been explored. Though opportunistic routing protocols such as ExOR [1] may take advantage of the broadcast nature of the channel to involve receivers in forwarding decisions, they still often rely on traditional routing protocols and extended measurements of the network's static topology.

In contrast, LFBL not only completely shifts forwarding decisions to the receiver, but it completely does away with the notion of routing based on the network topology. For each end-to-end flow, potential LFBL forwarders make decisions based on only a single scalar value for each endpoint, capturing the distance of the forwarding node from the endpoint. This distance can be determined by any number of metrics, and it is agnostic to the actual network topology. In essence, LFBL removes the need for *any* explicit knowledge about the network topology when calculating the cost of end-to-end paths.

This topology-agnostic approach to forwarding enables LFBL to cope with high dynamics more gracefully than existing routing protocols, such as the Ad-Hoc, On-Demand Distance Vector protocol (AODV) [7]. Protocols like AODV attempt to track the state of at least some of the links in the topology, and must reconfigure their routes when the state of these links change. By design, LFBL forwarding does not depend on topology changes. Our evaluation shows that this change enables LFBL to significantly outperform AODV in a variety of scenarios.

The main contribution of this work is a forwarding protocol with the following unique features:

- LFBL is topology-agnostic, that is, it has no knowledge of neighbors, routes, or next hops.
- LFBL uses exclusively broadcast communication for all packets, allowing for more flexibility in the selection of MAC layer protocols.

- LFBL supports mobility not only of nodes, but of application-level identifiers as well.

## 2. PROTOCOL OVERVIEW

Listen First, Broadcast Later (LFBL) operates on four basic tenets:

- **Keep the absolute minimum amount of state.** As opposed to mesh networks that have a fixed topology that they can measure and distribute information about, the topology of a highly dynamic network is constantly changing. Any state kept at a node will go stale quickly, and we will incur overhead for upkeep of that state. We address this issue by keeping the absolute minimum amount of state required for correct forwarding behavior.
- **Use data packets to distribute control information.** In order to minimize collisions due to additional packet transmissions, LFBL generates no separate control packets. Instead, a small LFBL header is piggybacked on every data packet.
- **Broadcast everything and learn from listening.** In alignment with our goal of a unicast-free MAC layer, LFBL uses only broadcast packets. Furthermore, it takes advantage of any overheard packet, and the control information it places within, to maintain the small amount of state that it does keep.
- **Make forwarding decisions exclusively at the receiver.** Using the information it has gathered, each node decides for itself whether to forward a received packet. It does this without any explicit coordination with its neighbors or the packet's sender. All signaling is done implicitly through the data packets themselves.

### 2.1 End-to-End Communication

From end-to-end, communication in an LFBL network is composed of two phases: a request phase and a response phase.

The *request phase* is similar to route request phases in traditional on-demand routing protocols, where (assuming the requester has no prior knowledge) the requester broadcasts a request packet that is flooded through the network. In traditional routing, the intended responder is identified by its IP address, and only the node with that IP address may respond. In LFBL, however, the request phase is meant to be aligned with a connection setup or discovery phase at the upper layers of the protocol stack, such as TCP connection establishment, name resolution, or P2P peer discovery. Thus, the intended responder is identified by an identifier provided by the application (which may or may not be an IP address), and it is the application at the responding node that decides whether it should respond to the request. This combines two separate discovery phases into one, reducing the number of floods required.

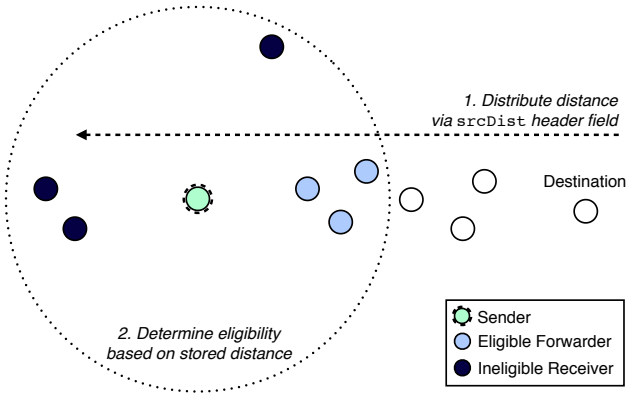


**Figure 1: Eligible forwarders.**

The *response phase* begins when a response reaches the requester. In this phase, the requester and responder exchange data normally, without flooding. Packets will follow the best available path through the intermediate nodes based on the distances of the nodes discovered through the request phase. In all subsequent request and response phases, packets are forwarded, not flooded, unless the requester or receiver ceases receiving packets. LFBL inserts its own header into each data packet, which contains information used to maintain end-to-end connectivity. This information is overheard by all nodes in hearing distance of the active path. Any of these nodes can potentially become forwarders at any time without any explicit coordination or path setup.

### 2.2 Forwarding

In this section, we will use the terms *sender* and *receiver* only in the context of single-hop transmission and reception on the wireless channel. We will use the terms *source* and *destination* to represent the communication endpoints.

As we have mentioned previously, all forwarding decisions in LFBL are carried out at the receiver side. Senders simply broadcast packets. Upon reception, a receiver first decide whether it is an *eligible forwarder* for the packet. If so, it waits for some amount of time, listening to the channel to see if another node closer to the destination forwards the packet. If not, the node forwards the packet itself.

Thus, a receiver has two important decisions to make: whether it is an eligible forwarder for a packet, and, if so, how long to wait before forwarding it.

#### *2.2.1 Am I an Eligible Forwarder?*

Since the goal of a forwarder is to move packets towards the destination, this question essentially maps to

the question: "Am I closer to the destination than the node I received this packet from?" In order to answer this question, the LFBL header in every packet contains a `srcDist` field, which contains the distance from the source of the packet to the sender. At each hop, a forwarder inserts its own distance from the source in the `srcDist` field before forwarding the packet. Nodes use the `srcDist` field to discover and update their distance from active endpoints in the network. Note that any number of distance metrics may be used (see Section 4.1).

Once a node knows its distance from another node, it can use this information when forwarding packets back *towards* that node. Specifically, only nodes closer to a packet's destination than the previous sender are eligible to be forwarders. Thus, the eligible forwarders at each hop are those nodes who (1) received the transmission, and (2) are closer to the destination. (See Figure 1.)

#### 2.2.2 How Long Should I Listen?

Since there will generally be more than one eligible forwarder for each transmission, the algorithm for determining a node's *listening period*, where the node waits to see whether other nodes will perform a forwarding task, serves two important purposes: forwarder prioritization and collision avoidance.

Prioritization means assigning shorter listening periods to eligible forwarders that believe they are on a better path than their neighbors between the sender and the destination. We will discuss various prioritization metrics in Section 4.1.

Collision avoidance simply means reducing the chance that two eligible forwarders will choose the exact same duration for their listening period, transmit simultaneously, and cause a collision. Adding a random factor to the listening period proves to be highly effective for this purpose according to our simulations (see Section 4).

## 3. LFBL IN DEPTH

This section describes the detailed operation of the LFBL protocol. We frequently refer to the LFBL header, a header placed in each data packet in lieu of separate control packets. A definition of each of the fields in the LFBL header can be found in Table 1.

### 3.1 Node State

One of the tenets of LFBL is to keep as little state as possible at each node. In particular, LFBL keeps state only about communication endpoints, never about any intermediate nodes or any topological information. The state that LFBL nodes *do* keep is described below.

#### 3.1.1 Distance Table

The LFBL distance table is the closest analog that LFBL has to the traditional routing table, in that it maps a destination IP address to some state about that node which is used for forwarding. However, the LFBL distance table only ever stores exactly three values per known active endpoint, independent of the number of neighbors a node has, the number of paths used to reach the endpoint, or any other topological information. For some node $N$ and endpoint $E$, these three values are: the highest sequence number (seqnum) seen in a packet sent by $E$, the distance from $N$ to $E$, and the variance of the distance from $N$ to $E$.

| | |
|---|---|
| `seqnum` | A monotonically increasing sequence number assigned by the source node. |
| `acknum` | A cumulative acknowledgement number also set by the source node. |
| `srcDist` | The distance between the source node and the most recent forwarder, modified at each hop. |
| `dstDist` | The distance between the most recent forwarder and the destination node, modified at each hop, if known. |
| `type` | The LFBL message type, which is either request (REQ), response (REP), acknowledgement (ACK), or a combined acknowledgement and request (ACK+REQ). |
| `appID` | The application-level identifier being requested, provided, or acknowledged. |

**Table 1: Fields in the LFBL header.**

A node $N$ may potentially update its distance table any time it overhears a packet being transmitted, even if it is not the destination or even an eligible forwarder. Upon overhearing a packet transmitted by sender $S$, $N$ first calculates its distance to the source via $S$. This distance, which we will call $d_{N\to S\to R}$, is simply the sum of the distance from $S$ to $N$ and the value in the `srcDist` field of the received LFBL header. The distance from $S$ to $N$ is determined by the network's distance metric, which may, for example, always be 1 (that is, the hop count), or a number based on the received signal strength. We discuss the choice of distance metric in Section 4.1.

Once $N$ has calculated $d_{N\to S\to R}$, it checks to see if it needs to update its distance table. If $N$ does not have an entry for $R$ in its distance table, it simply adds one using the calculated $d_{N\to S\to R}$ and the `seqnum` field in the received packet header. If $N$ already has an entry for $R$, its behavior depends on the `seqnum` field in the packet and the seqnum $s_R$ in $N$'s distance table. If `seqnum` is less than $s_R$, $N$ does nothing. If `seqnum` is equal to $s_R$, $N$ only sets its distance to $d_{N\to S\to R}$ if it is less than the value currently in $N$'s distance table. If `seqnum` is greater than $s_R$, $N$ sets its distance to $d_{N\to S\to R}$, regardless of the previous value, under the

assumption that a packet with a higher seqnum carries a fresher distance measurement.

Whenever $N$ updates a distance table entry with a new seqnum, it also adjusts its third and final value: the distance variance. Before replacing its distance table entry, it takes absolute value of the difference of the old and new distances, and rolls it into a rolling variance average. The specific algorithm is the same as is commonly used in TCP for estimating the variance in roundtrip times [6].

Since distance table entries are meant to track only *active* endpoints and are expected to go stale quickly, they can expire. A distance table entry is erased after it has not been updated for a configurable amount of time.

#### 3.1.2 Application-Level Identifier Map

This map is only used by active requesters. When a requester receives a response for application-level identifier (appID) $x$ from a responder with IP address $a.b.c.d$, it stores a mapping from $x$ to $a.b.c.d$ in its appID map. Requesters store only the IP address of the most recent responder. This allows the requester to address its next request directly to the most recent responder instead of having to fall back to flooding. Entries in the appID map time out after they have remained unused for a configurable period of time, or when requests to an IP address retrieved from the map go unanswered for a configurable period of time.

### 3.2 The Request Phase

A new request phase starts when an application wants to request some application-level identifier (appID). The appID is placed in the `appID` field of the outgoing LFBL header. If both (a) the requester has the IP address for a previous responder in its appID map, and (b) it has an entry for the the responder's IP address in its distance table, it addresses the packet to the responder. In this case, the request packet (REQ) is forwarded normally, as discussed in Section 3.3. Otherwise, the REQ is flooded.

Flooded REQs server two purposes: ensuring that any available responder is found, and distributing distance information so that other nodes in the network learn their distance from the requester. This way, any node is ready to help forward the response if need be. In fact, this can allow the response packet to implicitly establish multiple, disjoint paths between the requester and the responder (see Figure 2).

To avoid collisions and ensure accurate distance measurements during flooding, we use a technique inspired by Ye et al. [8] where each node delays rebroadcasting the flooded packet for a time period relative to its distance from the neighbor that sent the packet. We make two minor modifications to this algorithm: first, a node never rebroadcasts the same flooded packet twice, even if its distance improves. Though this may result in initially inaccurate distance measurements, it ensures minimal overhead for flooding. Any inaccuracies will quickly be corrected at any node that overhears the ensuing data exchange.

Once a responder receives the REQ, it produces a response packet (REP). In addition to application-layer data, the REP contains the distance from the responder to the requester in the `dstDist` field of the LFBL header. This distance is used by intermediate nodes to make forwarding decisions that ultimately get the REP back to the requester (see Section 3.3). Note that, as the REP travels, all forwarders and all of their neighbors that hear the packet will update their distance tables, learning their distance to the responder (see Section 3.1.1).

As a result, all of the nodes that forwarded or overheard the REP packet can serve as forwarders for future REQ packets, without the need for any more floods. In a reasonably dense network, there is a high likelihood that this will make multiple paths available for the requester to reach the responder. Furthermore, as nodes move and bidirectional traffic continues to flow between the requester and the responder, new nodes that move into range of the current path overhear these transmissions. These new nodes also update their distance tables, providing a fresh crop of eligible forwarders.

### 3.3 Forwarding

Once the distance tables in the network have been populated, normal, flood-free forwarding ensues. Receivers make forwarding decisions based on their distance table and information in the LFBL headers of received packets. Specifically, whenever a node forwards a packet, it updates the `srcDist` and `dstDist` fields in the LFBL header from its distance table before transmitting the packet. Receiving nodes then compare these values to those in their own distance tables as described below.

#### 3.3.1 Eligible Forwarders

As we briefly discussed in Section 2.2, an eligible forwarder is any node that both receives a transmission and is closer than the sender to the destination. Nodes determine whether they are closer to the destination by comparing the distance in their distance table to the `dstDist` field in the received packet.

This means that the distance metric is quite important – an overly optimistic distance metric will cause unnecessary contention for the channel and extra overhead due to all of the eligible forwarders. An overly pessimistic or insufficiently granular distance metric will provide too few eligible forwarders, so that if the few eligible ones don't receive the packet, forward progress

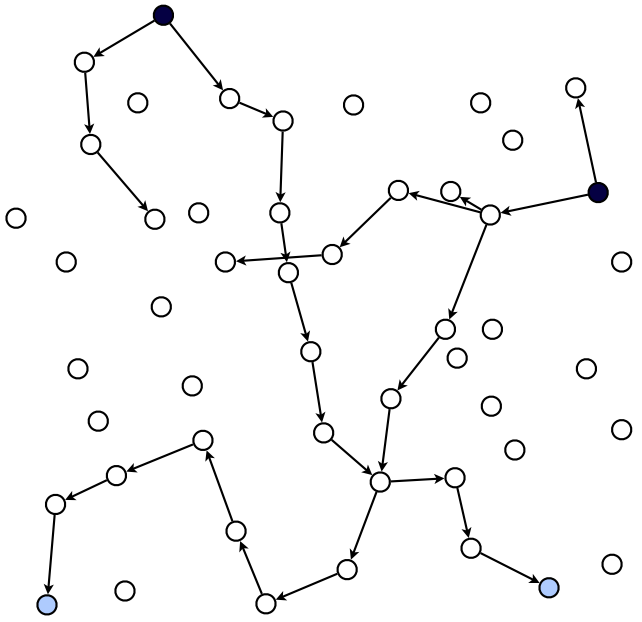

**Figure 2: LFBL REP forwarding paths.**

will stop and the packet will be lost. A pessimistic distance metric may even provide no eligible forwarders at a particular hop if the only previously eligible forwarders have moved. In Section 4.1, we evaluate different distance metrics for LFBL.

Note that a node is never an eligible forwarder if it does not have an entry for a packet's destination in its distance table. In this case, the node will drop the packet.

### 3.3.2 Listening Periods

As mentioned in Section 2.2, there are two reasons to set different listening periods across nodes. One is collision avoidance and the other is prioritization.

To avoid collisions, we introduce an element of randomness to a node's listening period. Additionally, we require that the MAC layer expose the state of the channel so that listening period timers can be paused when the channel is busy. This also decreases collisions, and allows nodes with a clear channel to transmit first.

To prioritize closer nodes, LFBL sets a threshold based on the distance from the best path. Despite the fact that LFBL uses no explicit paths, it can determine whether it is on the best path using just three pieces of information: (1) the distance provided by the sender in the `dstDist` field of the LFBL header, (2) the node's distance $d$ to the destination according to its distance table, and (3) the length $\ell$ of the hop the packet just traversed, according to the network's distance metric.

The logic is as follows. If the node is on the best path between the sender and the destination, the sender must have used this node's distance to update its own distance table. In other words, the sender would have set its distance to $d$ plus the distance from the node to the sender. If the network's distance metric is symmetrical and stable, then $d$ should be *exactly* $\texttt{dstDist} - \ell$. Though it is not always the case that the distance metric will be so reliable, we can reasonably assume that, the closer $d$ is to $\texttt{dstDist} - \ell$, the closer this node is to the best path from the sender to the destination. Furthermore, if $d$ is in fact less than or equal to $\texttt{dstDist} - \ell$, we assume that this node is *on* the best path.

This information can be used to adjust the node's listening period in a number of ways. In Section 4.1, we compare three methods, or *delay metrics*, for prioritization of nodes' listening period:

- A purely random listening period, using the distance metric only to separate eligible forwarders from ineligible ones.

- The *slotted random* metric, which divides eligible forwarders into two groups. The primary group is for those forwarders that appear to be on the best path ($d <= \texttt{dstDist} - \ell$). The rest of the eligible forwarders are placed in the secondary group. The primary group's time slot starts immediately upon reception of the packet, while the secondary group's time slot begins a fixed time period later. Within each slot, randomness is used for collision avoidance.

- The *distance + variance + random (DVR)* metric, where a node's listening period is based on its distance from the best path ($max(0,\ d - (\texttt{dstDist} - \ell))$) and the variance of its distance over time. (The details of how the variance is calculated is explained in Section 3.1.1.) To these two values, we also add a random factor to break ties and avoid collisions.

Regardless of the delay metric used, a node uses the listening period to listen for other eligible forwarders, to see if any of its neighbors forward the packet first. If a node hears a neighbor forward a packet with the same source and `seqnum`, it next examines the `dstDist` field. If the forwarder is closer to the destination than the receiving node, the node cancels its pending forward. However, if the forwarder is further away from the destination than the receiving node, it should compute a new listening period and restart its timer, as it may still be needed to make forward progress towards the destination.

Note that both interference from other transmissions and the hidden terminal problem can cause result in multiple eligible forwarders forwarding the packet. On the plus side, this can result in the use of alternate paths, creating path diversity. On the other hand, it

can create unnecessary extra transmissions. In the evaluation section, we see that the level of overhead caused by this issue is actually less than the overhead imposed by traditional routing, and the ability to discover alternate, disjoint paths probably contributes to the success of the protocol. We can see an example of this effect at work in Figure 2. These are the actual paths traversed by LFBL REP packets during a simulation of two simultaneous flows in a 50-node network. An arrow is drawn pointing from each forwarder to any other forwarder that received its transmission.

### 3.4 Implicit Request Handoff

One last feature of LFBL that helps to support logical mobility, where appIDs either move between nodes, or are present at multiple nodes at the same time, is *implicit request handoff*. In LFBL, one node can respond to a request addressed to a different node, as long as it can provide the requested appID. If the requester receives more than one response to its request, it can pick the one it prefers by choosing which to send an acknowledgment to. A responder which ceases to receive acknowledgements for its responses will eventually give up.

## 4. EVALUATION

Many dynamic routing protocols are evaluated using constant bit rate traffic, sent from one node to another without so much as a request or acknowledgement. This resembles a DDoS attack more than it does real network application traffic. Real network applications in use today are bidirectional, even if one endpoint is just sending requests and acknowledgements and the other side is sending all of the data. As a result, we use only bidirectional flows in our evaluation.

To obtain the simulation results presented in this section, we implemented LFBL in the QualNet network simulator[1]. AODV simulations were run using QualNet's built-in implementation of the protocol with bidirectional connection establishment enabled. At the physical layer, all simulations use 802.11b radios operating at a fixed rate of 11 MBps. We used two different MAC protocols: a simple Carrier Sense Multiple Access (CSMA) MAC, and the full 802.11 MAC. The CSMA MAC simply senses the channel, sending if it is free, or backing off for a random interval if it is busy. It uses no retransmissions, acknowledgements, RTS/CTS, etc.

Except where otherwise noted, all of the simulations below were conducted using 100 randomly placed nodes in a 1500 by 1500 meter area. Each individual simulation ran for five minutes of simulated time. We ran every simulation eight separate times with different random seeds. The values presented in the figures are the median of the eight means from the different simulation runs. In figures where error bars are present, they display the interquartile range of the eight means. For both LFBL and AODV, each bidirectional flow is composed of a requester, which sends a new request every 100 milliseconds, and a responder, which responds to any request it receives from the responder. All request packets are 36 bytes long and all response packets are 1400 bytes long.

[1]http://www.scalable-networks.com/

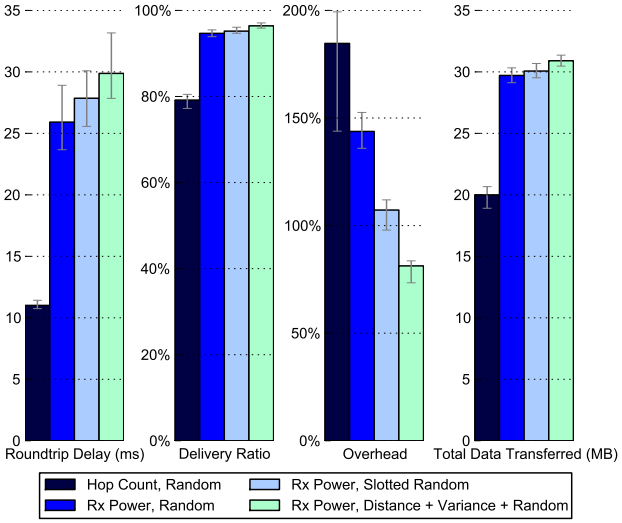


**Figure 5: A comparison of different distance and delay metrics.**

We evaluate each simulation using four evaluation metrics: roundtrip delay, delivery ratio, overhead, and total data transferred. The roundtrip delay is the amount of time elapsed from when a request is sent by a requester until it receives a response. The delivery ratio is the number of packets received (by any node, requester or responder) divided by the number of packets sent. Overhead is computed as the total number of packets sent to the MAC layer for transmission, divided by the total number of hops traversed by successfully received packets, minus 100 percent. Total data transferred is the sum of all bytes received by all requesters over the entire duration of the simulation.

### 4.1 Distance and Delay Metrics

Though LFBL does not depend on any particular distance metric, the choice of distance metric can significantly effect its performance. The leftmost two bars in each group in Figure 5 compare LFBL's performance with two different distance metrics: hop count and received signal strength. In these simulations, all 100 nodes move using a random waypoint model with no pause time and speeds between 0 and 30 meters per second. There are eight simultaneous, independent flows.

Clearly, the received signal strength metric performs

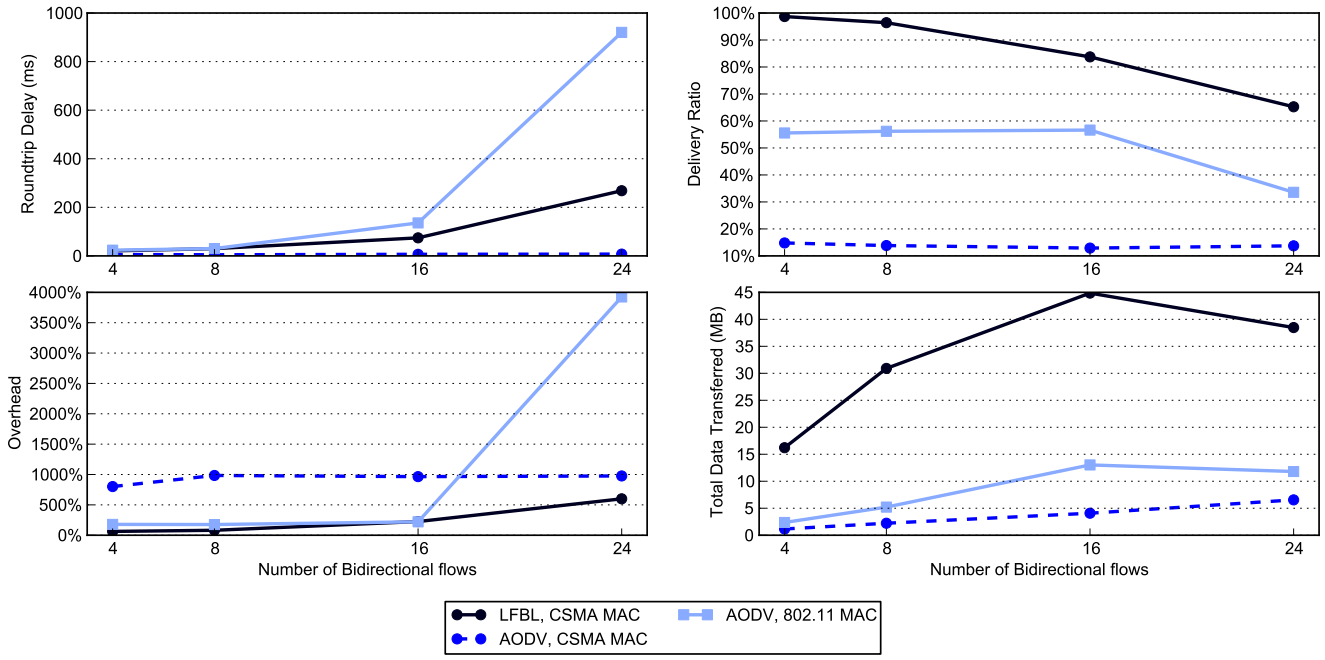


**Figure 3: Effect of the number of simultaneous data flows in the network.**

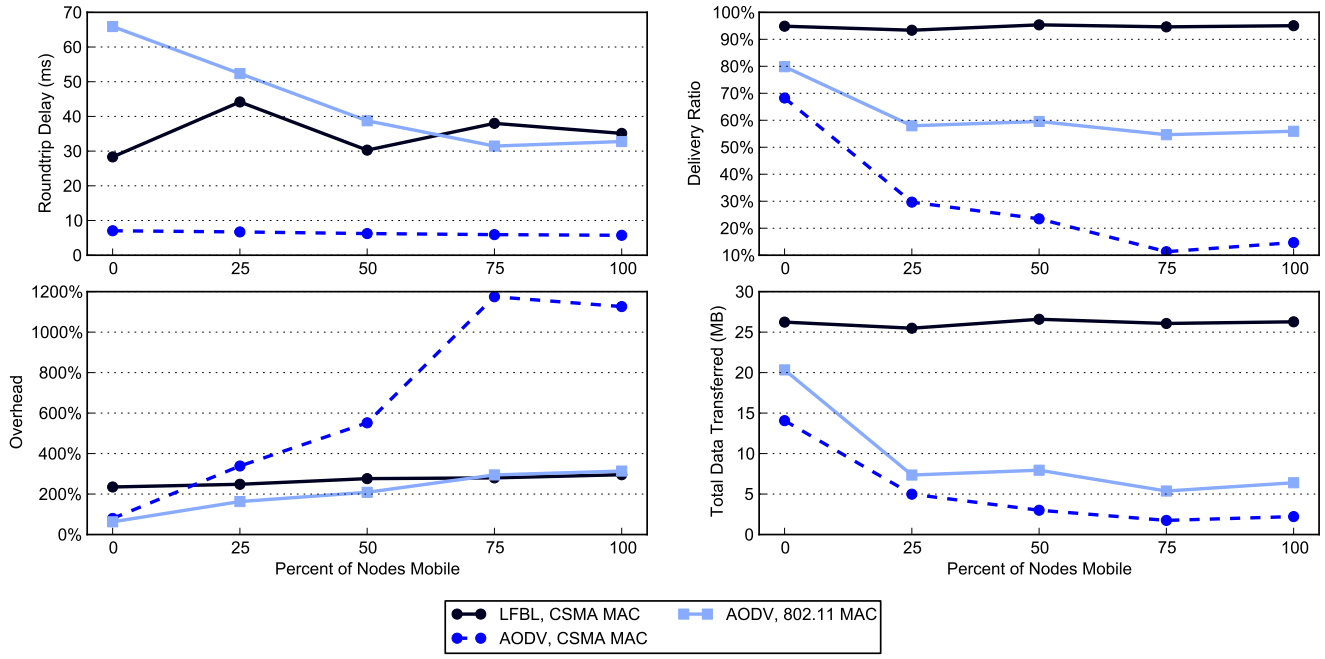


**Figure 4: Effect of the number of mobile nodes in the network.**

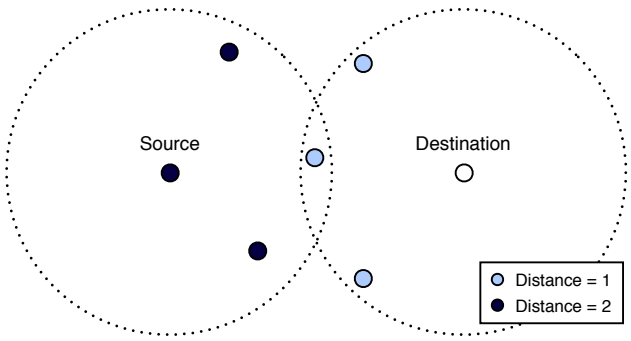


**Figure 6: Why hop count makes for a poor distance metric in LFBL.**

significantly better than the hop count. Figure 6 illustrates the likely cause. With the nodes in this configuration, the center node is the only eligible forwarder for the source, since it is the only node in the source's transmission range that is closer to the destination, according to the hop count distance metric. However, there are two perfectly useful alternate paths, one through the upper two nodes and another through the lower two nodes, neither of which will be used by LFBL. This hypothesis is supported by the lower delivery rate coupled with significantly lower roundtrip delays observed when using the hop count metric in Figure 5.

Figure 5 also shows the effects of different choices of delay metrics. Delay metrics determine the length of time an eligible forwarder will wait before forwarding a packet. We evaluate three different delay metrics in Figure 5: random, slotted random, and distance + variance + random (DVR). For the random delay metric, each eligible forwarder simply selects a random delay between zero and four milliseconds. For the slotted random delay metric, nodes select one of two slots based on the distance metric, as discussed in Section 3.3. Within each two-millisecond slot, the node selects a random delay between zero and two milliseconds. For the DVR delay metric, the node assigns itself a delay penalty based on its distance metric and its computed variance, as discussed in Section 3.3. It then adds a small random factor to break ties.

Each of these three metrics has a slightly higher delivery ratio than the last with significantly reduced overhead. The tradeoff is somewhat longer roundtrip times. The plots in the following sections all use DVR as the delay metric.

## 4.2 Network Utilization

Figure 3 shows the effect of differing levels of network utilization on LFBL and AODV. As in the previous section, all 100 nodes move using a random waypoint model with no pause time and speeds between 0 and 30 meters per second. We vary the number of bidirectional flows in the network, where no two flows share an endpoint. Thus, with 24 simultaneous flows, just short of half of the nodes in the network are actively transmitting data.

From these results, it is clear that AODV was not designed to be run on top of a primitive MAC protocol like the simple CSMA protocol used here, as its packet delivery ratios are under 20 percent. Versus AODV over 802.11, LFBL has significantly higher packet delivery ratios at all utilization levels and appears to degrade more gracefully. In particular, there is an enormous spike in both the roundtrip delay and the overhead of AODV over 802.11 when the number of flows reaches 24, as well as a marked drop in the delivery ratio. No such spikes or drops appear to be present in LFBL at these utilization levels.

## 4.3 Physical Mobility

Figure 4 shows the effect of differing amounts of physical mobility on LFBL and AODV. For these simulations, mobile nodes move at a constant velocity of 30 meters per second using a random waypoint model. Any remaining nodes do not move. Eight simultaneous, bidirectional data flows are present for this simulation.

As intended, LFBL is largely unaffected by the amount of mobility, maintaining a delivery ratio of well over 90 percent with only a small increase in overhead as the network becomes more mobile. Once again, AODV clearly does not interact well with the simple CSMA MAC protocol. AODV over 802.11 has lower overhead than LFBL in this scenario when the percentage of mobile nodes is lower, as well as somewhat shorter round trip times in the high mobility cases. However, AODV has a delivery ratio under 60 percent in the presence of mobility, and its record would appear to be far poorer with responses than requests – the gap in the total amount of data successfully received is significantly higher.

## 4.4 Logical Mobility

As previously noted, LFBL is designed to support not only physical mobility of nodes, but logical mobility of application-level identifiers (appIDs) as well. Figure 7 is a simple demonstration of this capability. For this simulation, as with the others, 100 nodes were randomly placed. We assigned 92 of them a random waypoint mobility model with zero pause time and a velocity between 0 and 30 meters per second. However, the remaining eight nodes were kept stationary. Using this setup, we ran two experiments. In both experiments, two of the mobile nodes served as requesters, each requesting a different appID. For the first experiment (dark blue bars), we assigned one stationary node as a responder for each of the two appIDs, creating two standard bidirectional flows. For the second experiment

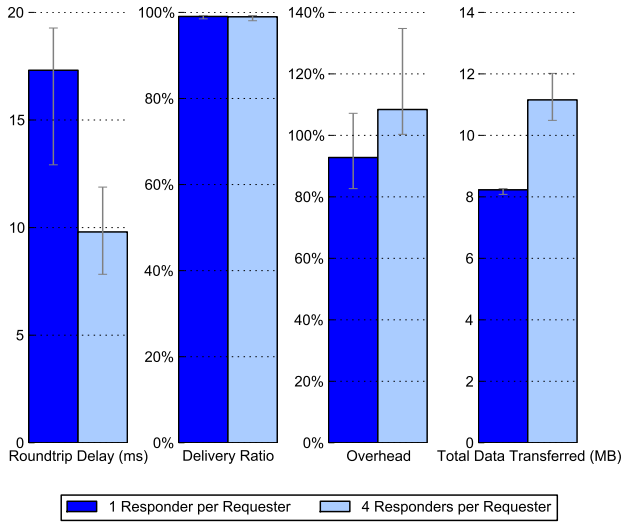


**Figure 7: Effect of multiple available responders.**

(light blue bars), we assigned four of the (randomly placed) stationary nodes to respond to the *same* appID for each of the two appIDs being requested. The expected result is a behavior similar to anycast routing, where, as the requesters move away from one responder and towards another, the closer responder will take over.

The results in Figure 7 support the expected behavior. When the number of responders is higher, the roundtrip delay drops significantly. This indicates that shorter paths were used throughout each simulation run, despite the mobility of the requesters. The increase in total data transferred despite an identical delivery ratio and fixed request rate indicates that the requesters occasionally received responses from multiple responders. However, the increase is only 30 percent, meaning that, out of four possible responders, each requester only received 3 extra responses for every 10 requests. If all responders responded to all requests, each requester would receive 3 extra responses for every *one* request. This small number of extra responses is expected during initialization and handover phases.

# 5. DISCUSSION: SUPPORTING APPLICATIONS

Applications generally require a discovery phase that resolves application-level *identifiers* to the IP addresses of actual hosts. In the Internet, the discovery phase would likely involve a query to the DNS infrastructure to resolve a DNS name, but in unstructured, highly dynamic wireless networks, this will generally have to be a flooded query. Routing protocols for wireless networks also tend to involve a flooding discovery phase in order to determine the physical location of the node with a particular IP address.

Put another way, there are three identifying traits for any communication endpoint in a network:

- An application-level identifier, such as a URI, DNS name, etc.
- A node identifier, such as an IP address, a MAC address, etc.
- The physical location of a node in the network topology, the form of which depends on the routing protocol in use.

Two discovery processes are required here: one to determine the node identifier for a given application-level identifier (normally the job of a separate application-level service), and a second to determine the physical location of a given node identifier (the job of the routing protocol). LFBL allows applications to conflate these two processes by combining its own discovery phase with the application's. The requesting application provide an identifier to LFBL, and LFBL leaves it up to the application at each receiving node to decide if it wants to reply. This enables deployment not only of traditional Internet-like modes of communication, but new types of applications.

For example, named-content-based communication [4] can greatly benefit from the LFBL capability to deal with high dynamics. A node can initiate interest for content by sending an LFBL request for the content identifier. Any node that has the content, either because it is the originator of the content or because it has a cached copy of the content, can respond to the initiator of the request. LFBL will maintain the ability to reach the *content*, rather than any particular node providing the content, despite dynamics caused by the mobility of the requester, the mobility of the responders, the mobility of the forwarders, or the changing availability of the content due to the opportunistic nature of caching.

# 6. RELATED WORK

## 6.1 Opportunistic Routing

The basic idea of making forwarding decisions at the receiver rather than the sender has been explored in the past within the context of opportunistic routing. Next, we present a selected number of publications in this space.

ExOR [1]: In ExOR, forwarding decisions are made at the receiver, but the computation of the routing tables is done in a traditional way, using a link state routing protocol. ExOR was designed for wireless mesh networks and as such routing updates caused due to topology changes are not frequent events. Adapting ExOR in a mobile environment becomes challenging,

given the high overheard incurred due to frequent topology changes. In contrast, LFBL has been designed from the beginning with high dynamics in mind, which led us to a radically different design in terms of route computations.

MORE [2]: MORE is a natural extension of an opportunistic routing protocol that takes advantage of network coding techniques. While network coding inherently can mask failed paths, caused due to link quality or topology changes, MORE still does not address the problem of routing in highly dynamic networks. As in the case of ExOR, MORE was designed for wireless mesh networks, and as such it does not have some of the advantages of LFBL when it comes to maintain routing state.

ROMER [10]: ROMER makes us of opportunistic routing in order to deal with the link quality fluctuations in wireless mesh networks. Packets are forwarded on paths that dynamically adjust based on the end-to-end quality. In addition, ROMER enables control transmission of redundant packets over alternative paths in order to improve the overall system resiliency. Again, LFBL differs in the way that routing state information is collected and maintained. ROMER requires full topology information, while LFBL needs one to keep track of the distance for each active endpoint.

SSR [3]: SSR shares many similarities with LFBL, mainly it receiver based forwarding decision making process and its ability to make use of the wireless broadcast channel for routing updates. On the other hand, SSR was also designed for static wireless networks, with the end-to-end hop count being the main optimization goal. As such, the SSR protocol designed has been highly coupled to the hop count metric. As our evaluation shows hop count is not a desirable metric under high dynamics, making SSR less applicable for mobile wireless networks.

### 6.2 Sensor Networks

GRAB [8, 9]: GRAB makes use of a similar metric based forwarding scheme. On the other hand, it does not take advantage of the opportunistic routing techniques. Instead, all eligible nodes forward packets, and it uses a feedback scheme to control the forwarding redundancy. Given that GRAB was designed for static sensor networks it cannot efficiently deal with topology changes.

### 6.3 Geographical Routing

GPSR [5]: GPSR makes use of geographical information in order to route packets. That enables nodes to route without the need of maintaining the network topology information. On the other hand, GPSR requires a mapping of node identifiers to location identifiers, a mapping service non essential for LFBL. Furthermore, GPSR does not take advantage of opportunistic forwarding, which results to nodes having to maintain state for every neighbor.

## 7. CONCLUSION

*Physical mobility* of nodes and *logical mobility* of application identifiers are two of the main causes of dynamics in multi-hop wireless networks. When we endeavored to address the latter, expecting that existing ad-hoc routing protocols were capable of handling the former, we were surprised to find that dealing with high dynamics in general was still an elusive goal. We attributed this failure of existing routing protocols mainly to their dependence on network topology. While current ad-hoc networks were designed to deal with changes in the network topology, their main pitfall in dealing with those dynamics was that they required the full or partial network topology for the computation of best routes. As such frequent changes in the topology had a direct impact on the performance of those protocols. Based on the above observations we set to design a new *routing* protocol for highly dynamic multi-hop wireless networks, capable of dealing both with physical mobility of nodes and logical mobility of application identifiers.

The result is Listen First, Broadcast Later (LFBL), a new multi-hop wireless protocol comprising of a distributed *forwarding* capability with essentially no *routing* protocol. The main tenets of LFBL are that all communications are done through the broadcast wireless medium, and minimal state information is maintained only for active endpoints. No unicast communications are used, no per-neighbor state is maintained, and no network topology information is required. These design choices have the following profound implications. Nodes can maintain their forwarding tables in a completely distributed manner without the need of explicit signaling by listening to the broadcast medium. At the same time, they are able to gracefully adapt to dynamics, by leveraging new paths without introducing any overhead caused by topology changes. Other features of LFBL, such as its capability to combine multiple discoveries for the various levels of identifiers, i.e. application, node and location identifiers, make LFBL suitable for various modes of communication, ranging from traditional Internet-style communication to recently proposed named-content-based communication [4].

We have experimentally evaluated LFBL and compared it against AODV, a representative wireless ad-hoc routing protocol. LFBL outperforms AODV using four different metrics and in a diverse set of simulation scenarios. Our results show that, under high dynamics, LFBL delivers around 5 times more packets compared to AODV, while having comparable overhead, introduced mainly due to redundant packets. This and other results validate our design choices in terms of the

capability of LFBL to deal with highly dynamic environments, caused due to physical mobility of nodes and logical mobility of application identifiers. In the future we plan to use LFBL to enable new types of application protocols for wireless multi-hop networks that follow more content and service-centric communication models.

## 8. REFERENCES


[1] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. *ACM SIGCOMM Computer Communication Review*, 35(4):144, 2005.

[2] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 169–180, New York, NY, USA, 2007. ACM.

[3] G. Chen, J. Branch, and B. Szymanski. Self-selective routing for wireless ad hoc networks. volume 3, pages 57 – 64 Vol. 3, aug. 2005.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12, New York, NY, USA, 2009. ACM.

[5] B. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, page 254. ACM, 2000.

[6] V. Paxson and M. Allman. Computing tcp's retransmission timer, 2000.

[7] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. *Mobile Computing Systems and Applications, IEEE Workshop on*, 0:90, 1999.

[8] F. Ye, A. Chen, S. Lu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Tenth Internation Conference on Computer Communications and Networks*, pages 304–309. Citeseer, 2001.

[9] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11(3):285–298, 2005.

[10] Y. Yuan, H. Yang, S. Wong, S. Lu, and W. Arbaugh. ROMER: Resilient opportunistic mesh routing for wireless mesh networks. In *The 1st IEEE Workshop on Wireless Mesh Networks (WiMesh*. Citeseer, 2005.