

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

Michael Meisel (UCLA), Vasileios Pappas (IBM Research), Lixia Zhang (UCLA)

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 predetermined 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 data. Under simulation, LFBL significantly outperforms AODV.

I. INTRODUCTION

Today, state-of-the-art, dynamic wireless data networks still use a protocol stack that was designed for the relatively static wired Internet. This includes traditional, IP-based routing at the network layer and unicast, point-to-point transmissions at the MAC layer.

In this paper, we present Listen First, Broadcast Later (LFBL), a forwarding protocol for wireless networks that does not require predetermined routes, IP addressing, or a unicast-capable MAC layer. This combination allows LFBL to seamlessly handle two types of high dynamics: *physical* mobility of nodes and *logical* mobility of content. Handling physical mobility means that the performance of LFBL is not dependent on the type or frequency of topological changes in the network. Handling logical mobility means that data or service requests need not be addressed to a specific node in advance – they can be fulfilled by any node that can provide the requested data.

The basic operation of LFBL 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. If it hears no such transmission, it forwards the packet itself.

While the idea of receiver-centric forwarding is not new in and of itself [9], [1], [3]), previous work in this area often does not support high dynamics, is still tied to IP-based addressing, and relies on the availability of unicast support at the MAC layer. As a result, the performance of these protocols

cannot natively support logical mobility and their performance depends on the level of physical mobility in the network. For example, ExOR [1], perhaps the canonical opportunistic routing protocol, relies on extensive measurements of the network’s static topology distributed via traditional routing protocols.

Traditional routing protocols for highly dynamic wireless networks, such as the Ad-Hoc, On-Demand Distance Vector protocol (AODV) [8], are also dependent on the level of physical mobility in the network, since they must set up explicit routes between endpoints. This requires them to track the state of at least some of the links in the topology and reconfigure their routes when the state of these links change. They also do not natively support logical mobility.

In contrast, LFBL completely does away with the notion of route selection. For each end-to-end flow, potential LFBL forwarders make decisions based only on their distance from the endpoint – a single scalar value. This distance can be determined by any number of metrics. In essence, LFBL removes the need for *any* explicit knowledge about the intervening network topology when calculating the cost of end-to-end paths. Our evaluation shows that this fundamental 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 properties:

- LFBL is topology-agnostic, that is, nodes maintain 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 not only physical mobility of nodes, but logical mobility of data as well – packets are addressed using data names rather than IP addresses.

II. 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.
- **Broadcast everything and learn from listening.** In alignment with our goal of a unicast-free MAC layer,

LFBL uses only broadcast packets. LFBL takes advantage of any overheard packet, and the control information it places within, to maintain the small amount of state that it does keep. LFBL generates no separate control packets.

- **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.
- **Drop IP addresses in favor of named data.** From an application’s perspective, IP addresses lose much of their meaning in a highly dynamic network. Not only must IP addresses be assigned and reassigned to the constantly changing set of active nodes in the network, applications must also maintain mappings between their own, application-level data names and these IP addresses – mappings which can be in constant flux. LFBL can use application-level data names directly in forwarding, providing seamless support for logical mobility.

A. End-to-End Communication

From end to end, communication in an LFBL network is composed of two phases: a request phase and a data 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 contrast, LFBL uses a data-centric approach. Requesters simply request the data they are interested in by name, and each receiving node decides whether it can respond to the request. Thus, applications need no additional discovery phase to map between content and node.

The *data phase* begins when a response reaches the requester. In this phase, the requester and responder exchange data via normal LFBL forwarding (as described below), without flooding. The data phase continues until the requester either (a) ceases sending requests or (b) is no longer receiving responses to its requests. In the latter case, the requester returns to the request phase.

B. Forwarding

LFBL inserts its own header into each data packet, which contains information used to maintain end-to-end connectivity. In both the request and data phases, this information overheard by all nodes in transmission range of an active path. During the data phase, any of these nodes can potentially become forwarders at any time without any explicit coordination or path setup.

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 mentioned previously, all forwarding decisions in LFBL are carried out at the receiver side. Senders simply broadcast

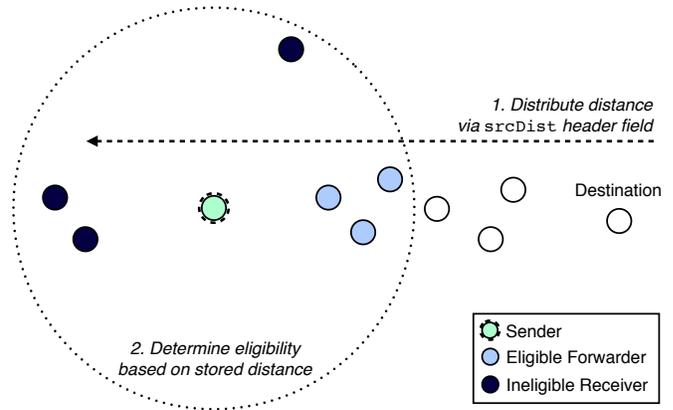


Fig. 1. Eligible forwarders.

packets. Upon reception, a receiver first decides 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.

1) *Am I an Eligible Forwarder?:* Since the goal of a forwarder is to move packets toward 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 between the source of the packet and 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 of a number of distance metrics may be used (see Section IV-A).

Once a node knows its distance from another node, it can use this information when forwarding packets back *toward* 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 (a) received the transmission, and (b) are closer to the destination. (See Figure 1.)

2) *How Long Should I Listen?:* Since there will generally be more than one eligible forwarder for each transmission, before forwarding, each node chooses some amount of time to wait and see if other nodes perform the forwarding task. This time is called the node’s *listening period*. It 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 IV-A.

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

seqnum	A monotonically increasing sequence number assigned by the source node.
acknum	A cumulative acknowledgment number also set by the source node.
srcId	A unique endpoint identifier for the source node, generally its MAC address.
dstId	A unique endpoint identifier for the destination 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), acknowledgment (ACK), or a combined acknowledgment and request (ACK+REQ).
dataName	The name of the data being requested, provided, or acknowledged.

TABLE I
FIELDS IN THE LFBL HEADER.

a collision. Adding a random factor to the listening period proves to be highly effective for this purpose according to our simulations (see Section IV).

III. 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 I.

A. 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 neighbors. The state that LFBL nodes *do* keep is described below.

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 identifier 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 .

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 $d_{N \rightarrow S \rightarrow R}$, 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 IV-A.

Once N has calculated $d_{N \rightarrow S \rightarrow R}$, 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 $d_{N \rightarrow S \rightarrow R}$ 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 $d_{N \rightarrow S \rightarrow R}$ 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 $d_{N \rightarrow S \rightarrow R}$, 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 round-trip times [7].

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.

2) *Responder Map*: This map is only used by active requesters. When a requester receives a response for data with name x from a responder R , it stores a mapping from x to R in its responder map. R is identified by some unique *endpoint identifier*, generally its MAC address, which the requester retrieves from the `srcId` field in the LFBL header. This allows the requester to address its next request directly to R instead of having to fall back to flooding. Entries in the responder map time out after they have remained unused for a configurable period of time, or when requests to an endpoint identifier retrieved from the map go unanswered for a configurable period of time.

B. The Request Phase

A new request phase starts when a node wants to request some data. The requester places the name of the data in the `dataName` field of the outgoing LFBL header. If both (a) the requester has the endpoint identifier for a previous responder in its responder map, and (b) it has an entry for the responder's endpoint identifier 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 III-C. 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.

To avoid collisions and ensure accurate distance measurements during flooding, we use a technique inspired by Ye et al. [9] 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 III-C). 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 III-A1).

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.

C. 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.

1) *Eligible Forwarders*: As we briefly discussed in Section II-B, 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 – that is, one that causes too many receivers to assume that they are eligible forwarders – will cause unnecessary contention for the channel and thus extra overhead. An overly pessimistic or insufficiently granular distance metric will provide too few eligible forwarders, so that if none among this small set receive the packet, forward progress will stop and the packet will be lost. In Section IV-A, 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.

2) *Listening Periods*: As mentioned in Section II-B, 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, a node 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 ℓ of the hop the packet just traversed, according to the network’s distance metric.

The logic is as follows. If a node N is on the best path between the sender and the destination, the sender must have used N ’s distance to update its own distance table. In other words, the sender would have set its distance to d plus the distance from N to the sender. If the network’s distance metric is symmetric and stable, then d should be *exactly* $\text{dstDist} - \ell$. 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 $\text{dstDist} - \ell$, 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 $\text{dstDist} - \ell$, we assume that N is *on* the best path.

This information can be used to adjust the node’s listening period in a number of ways. In Section IV-A, 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 ($d \leq \text{dstDist} - \ell$). 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 ($\max(0, d - (\text{dstDist} - \ell))$) and the variance of its distance over time. The point of including the variance in this metric is to penalize nodes with an unstable distance value, favoring more stable paths. (The details of how the variance is calculated is explained in Section III-A1.) To these two values, we also add a random factor to break ties and avoid collisions.

Regardless of the delay metric used, a node N uses the listening period to listen for other eligible forwarders, to see if any of its neighbors forward the packet first. If N hears a neighbor forward a packet with the same `srcId` and `seqnum`, it next examines the `dstDist` field. If the forwarder is closer to the destination than N , N cancels its pending forward.

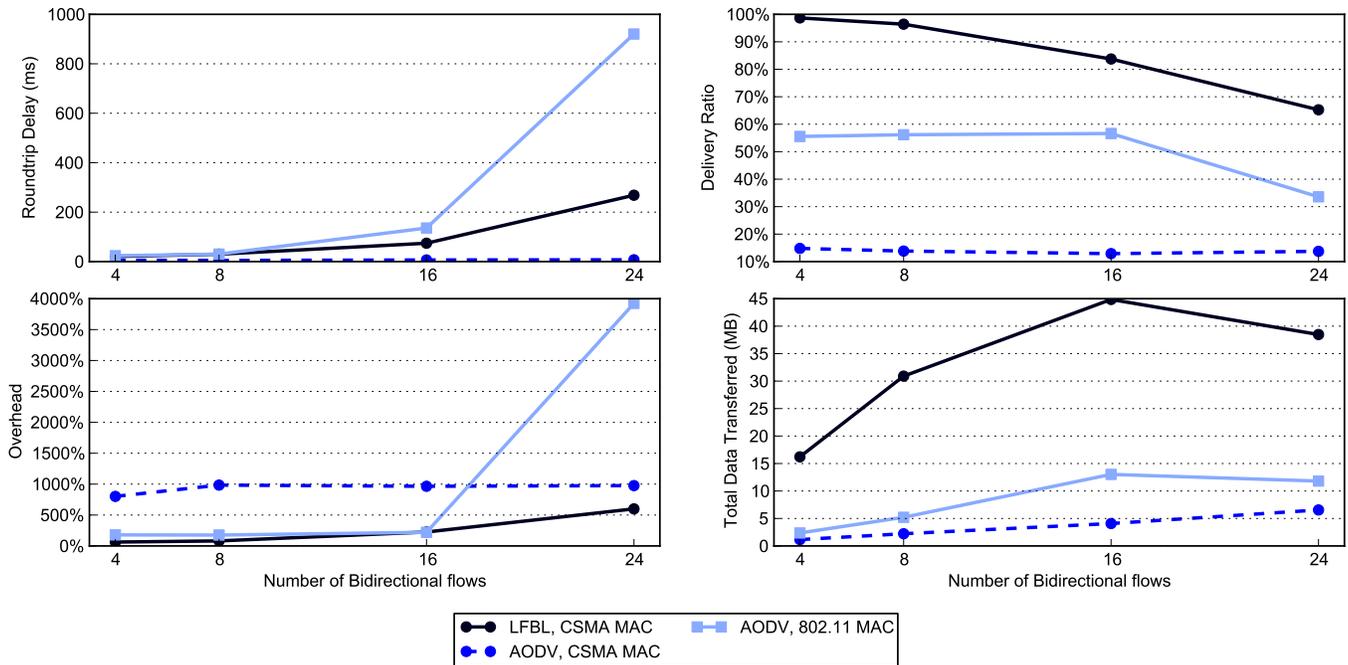


Fig. 2. Effect of the number of simultaneous data flows in the network.

However, if the forwarder is further away from the destination than N , N should compute a new listening period and restart its timer, as it may still be needed to make forward progress toward the destination.

Note that both interference from other transmissions and the hidden terminal problem can 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.

D. Implicit Request Handoff

One last feature of LFBL that helps to support logical mobility, where data either move between nodes or is 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 dataName. If the requester receives more than one response to its request, it can pick the one it prefers by choosing which responder to send an acknowledgment to. A responder which ceases to receive acknowledgments for its responses will eventually give up.

IV. EVALUATION

To obtain the simulation results presented in this section, we implemented LFBL in the QualNet network simulator. 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.¹ All request packets are 36 bytes long and all response packets are 1400 bytes long.

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 total number of packets received (by any node, requester or responder) divided by the

¹It is important to note that 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. LFBL was designed with real, bidirectional network applications in mind. As a result, we use only bidirectional flows in our evaluation. This is the major factor contributing to AODV's poor performance, as compared to other work that has evaluated that protocol.

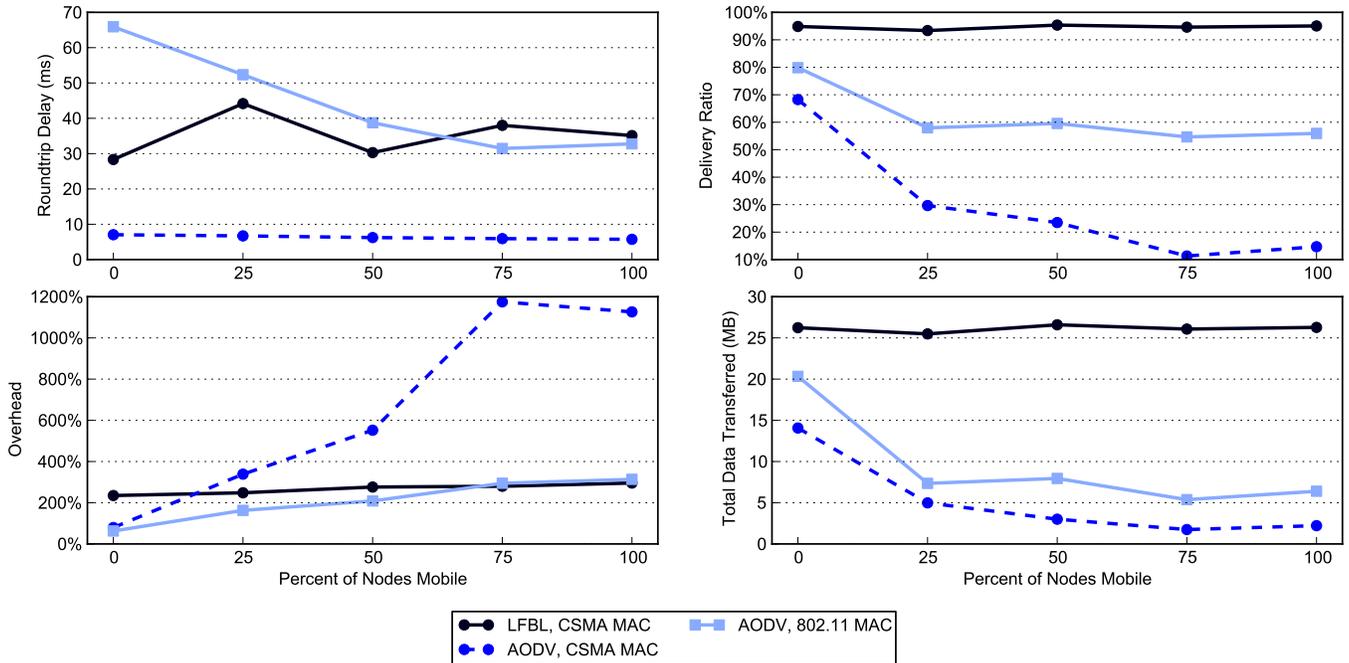


Fig. 3. Effect of the number of mobile nodes in the network.

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 one. That is to say, it is the portion of transmissions used for something other than the successful delivery of application-layer data (either requests or responses). Total data transferred is the sum of all bytes received by all requesters over the entire duration of the simulation.

A. Distance and Delay Metrics

Though LFBL does not depend on any particular distance metric, the choice of distance metric can significantly affect its performance. The leftmost two bars in each group in Figure 4 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 significantly better than the hop count. Figure 4 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 4: random, slotted random, and distance + variance + random (DVR). See Section III-C for descriptions of these metrics. For the random delay metric, each eligible forwarder simply selects a random delay between zero and 4 milliseconds (ms). For the slotted random delay metric, nodes select one of two, 2-ms slots based on the distance metric. Within each slot, the node selects a random delay between zero and 2 ms. For the DVR delay metric, the node assigns itself a delay penalty based on its distance metric and its computed variance. 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.

B. Network Utilization

Figure 2 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 are present in LFBL at these utilization levels.

C. Physical Mobility

Figure 3 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

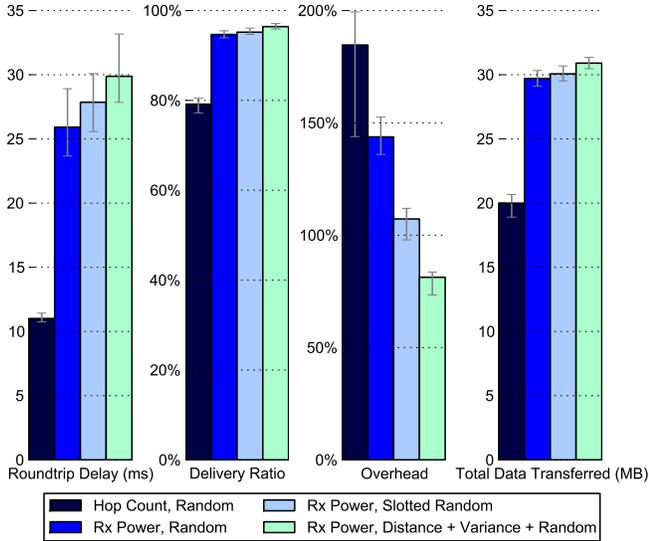


Fig. 4. A comparison of different distance and delay metrics.

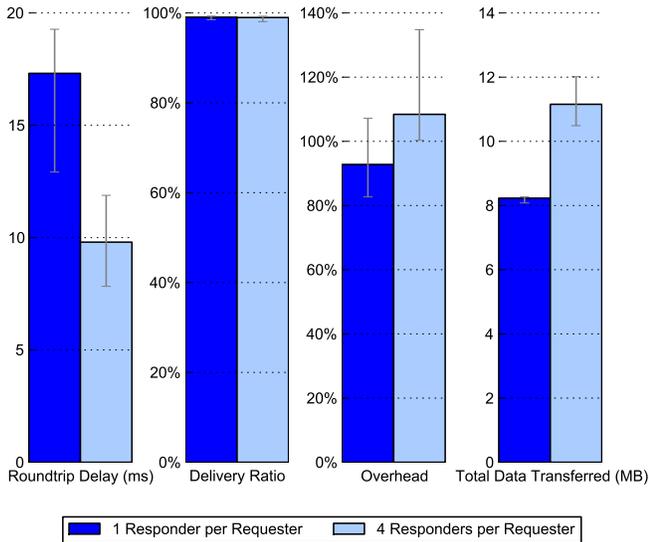


Fig. 5. Effect of multiple available responders.

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 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 the gap in the total amount of data successfully received between the two protocols is significantly higher. This suggests that AODV fares far poorer with the larger response packets than it does with requests.

D. Logical Mobility

As previously noted, LFBL is designed to support not only physical mobility of nodes, but logical mobility of data as well. Figure 5 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 dataName. For the first experiment (dark blue bars), we assigned one stationary node as a responder for each of the two dataNames, creating two standard bidirectional flows. For the second experiment (light blue bars), we assigned four of the (randomly placed) stationary nodes to respond to the *same* dataName for each of the two dataNames 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 5 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.

V. RELATED WORK

The basic idea of making forwarding decisions at the receiver rather than the sender has previously been explored in the context of opportunistic routing. However, none of the previous approaches addresses forwarding in highly dynamic networks. In ExOR [1], forwarding decisions are made at the receiver, but only after using a traditional link state routing protocol to determine link weights. ExOR was designed for wireless mesh networks and as such routing updates caused due to topology changes are not frequent events. Adapting ExOR to a mobile environment would be challenging, given the high overhead incurred by frequent topology changes.

MORE [2] is an opportunistic routing protocol that takes advantage of network coding techniques. While network coding can inherently mask path failures 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 can incur high overhead to maintain routing state.

ROMER [11] makes use of opportunistic routing 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. ROMER enables control transmission of redundant packets over alternative paths in order to improve overall system resiliency. Again, ROMER, having been designed for mesh networks, requires full topological information.

SSR [3] has much in common with LFBL, particularly its receiver-based forwarding algorithm. However, 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 design has been highly coupled to the hop count metric. Our evaluation shows that the hop count metric performs poorly under high dynamics. SSR also requires explicit acknowledgement of forwarded packets, incurring extra overhead. Furthermore, unlike LFBL, SSR is dependent on IP addressing and does not support *logical* mobility.

In the realm of sensor networks, GRAB [9], [10] also makes use of a distance-metric-based forwarding scheme. However, GRAB does not address the issue of collision avoidance or forwarder prioritization. Instead, all eligible nodes forward packets at each hop. GRAB uses a hop-count-based feedback scheme to reduce forwarding redundancy. Given that GRAB was designed for static sensor networks, it has no mechanism to deal with mobility.

In the geographical routing space, GPSR [5] uses geographical information to enable nodes to route without maintaining information about the network topology. However, GPSR requires a mapping of node identifiers to location identifiers, which may not be readily available in all networks. Furthermore, GPSR does not take advantage of opportunistic forwarding, meaning that nodes having to maintain state for every neighbor.

VI. CONCLUSION

Physical mobility of nodes and *logical mobility* of data 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 handling high dynamics in general was still an elusive goal. We attributed this failure of existing routing protocols mainly to their precomputation of best paths – a process that requires knowledge of the network topology. As such, frequent changes in the topology have a direct impact on the performance of those protocols. Based on these observations, we set out to design a new architecture for highly dynamic multi-hop wireless networks, capable of dealing both with physical mobility of nodes and logical mobility of data.

The first major product of our efforts is Listen First, Broadcast Later (LFBL), a new, multi-hop wireless protocol comprised of a distributed *forwarding* algorithm with no explicit routing protocol. The main tenets of LFBL are: nodes store a minimal amount of state, all communications are done using the natural broadcast capability of the wireless medium, all forwarding decisions are made by the receiver, and all addressing is data-centric. 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 for explicit signaling, by simply 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 to update topology information. Furthermore, LFBL supports logical mobility via named-data-based communication, removing the need for applications to pick a specific destination IP address before they can make requests for content or services. This architectural change can save applications from having to discover, store, and maintain up-to-date mappings between logical identifiers and IP addresses. We hope to fully realize this capability in future work by combining LFBL with Named-Data Networking (NDN, aka CCN) [4], as described in a recent paper [6].

We have experimentally evaluated LFBL and compared it to AODV, a representative wireless, ad-hoc routing protocol. LFBL outperforms AODV across four different metrics in a diverse set of simulation scenarios. Our results show that, under high dynamics, LFBL delivers nearly five times more packets than AODV with comparable overhead. This and other results validate our design choices – LFBL performs extremely well in highly dynamic environments, independent of the levels of physical and logical mobility.

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] M. Meisel, V. Pappas, and L. Zhang. Ad Hoc Networking via Named Data. In *Proceedings of the Fifth ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*, September 2010.
- [7] V. Paxson and M. Allman. Computing tcp's retransmission timer, 2000.
- [8] 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.
- [9] F. Ye, A. Chen, S. Lu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Tenth International Conference on Computer Communications and Networks*, pages 304–309. Citeseer, 2001.
- [10] 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.
- [11] 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.