Routing with a Markovian Metric to Promote Local Mixing

Yunnan Wu[*], Saumitra M. Das[†], Ranveer Chandra[*]

[*]*Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399.*
`yunnanwu,ranveer@microsoft.com`
[†]*School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907.*
`smdas@purdue.edu`

November 8, 2006

Technical Report
MSR-TR-2006-158

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Routing with a Markovian Metric to Promote Local Mixing

Yunnan Wu*, Saumitra M. Das†, Ranveer Chandra*.
*Microsoft Research, One Microsoft Way, Redmond, WA 98052. {yunnanwu,ranveer}@microsoft.com
†School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907. smdas@purdue.edu

***Abstract*— Routing protocols have traditionally been based on finding shortest paths under certain cost metrics. A conventional routing metric models the cost of a path as the sum of the costs on the constituting links. This paper introduces the concept of a *Markovian metric*, which models the cost of a path as the cost of the first hop plus the cost of the second hop conditioned on the first hop, and so on.**

The notion of the Markovian metric is fairly general. It is potentially applicable to scenarios where the cost of sending a packet (or a stream of packets) over a link may depend on the previous hop of the packet (or the stream). Such scenario arises, for instance, in a wireless mesh network equipped with *local mixing*, a recent link layer advance. This scenario is examined as a case study for the Markovian metric. The local mixing engine sits between the routing and MAC layers. It maintains information about the packets each neighbor has, and identifies opportunities to mix the outgoing packets via network coding to reduce the transmissions in the air. We use a Markovian metric to model the reduction of channel resource consumption due to local mixing. This leads to routing decisions that can better take advantage of local mixing. We have implemented a system that incorporates local mixing and source routing using a Markovian metric in Qualnet. The experimental results demonstrate significant throughput gain and resource saving.

I. Introduction

Network coding refers to a scheme where a node is allowed to generate output data by mixing (i.e., computing certain functions of) its received data. The broadcast property of the wireless medium renders network coding particularly useful. Consider nodes v_1, v_2, v_3 on a line, as illustrated in Figure 1. Suppose v_1 wants to send packet $\boldsymbol{x}_1$ to v_3 via v_2 and v_3 wants to send packet $\boldsymbol{x}_2$ to v_1 via v_2. A conventional solution would require 4 transmissions in the air (Figure 1(a)); using network coding, this can be done using 3 transmissions (Figure 1(b)). The key here is that a single broadcast transmission of $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$ (the bitwise XOR of the two packets) presents $\boldsymbol{x}_2$ to node v_1 who knows $\boldsymbol{x}_1$, and $\boldsymbol{x}_1$ to node v_3 who knows $\boldsymbol{x}_2$. This technique was termed *physical piggybacking* by Wu et al. [1] because the two packets are combined into one, without even increasing the size of the packet. It looks as if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are getting a shared ride in the air.

It is not hard to generalize Figure 1 to a chain of nodes. For packet exchanges between two wireless nodes along a line, the consumed channel resource could potentially be halved with physical piggybacking. Wu et al. further showed a simple distributed implementation that can realize such advantages in practice. Specifically, each wireless router can examine its local buffer and mix a left-bound packet with a right-bound packet (here "left" and "right" are in the relative sense). Such a mixture packet can be demixed by the left and right neighbors.

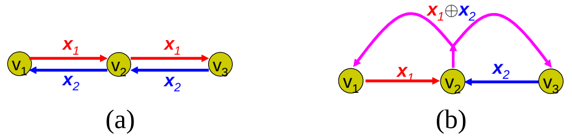

Fig. 1. (a) The conventional solution requires 4 transmissions to exchange two packets between v_1 and v_3 via a relay node v_2. (b) Using network coding, two packets can be exchanged in 3 transmissions [1].

Generalizing [1], Katti et al. [2] recently presented a framework for taking advantage of physical piggybacking to improve the efficiency of unicasting in multi-hop wireless networks. In their approach, each node snoops on the medium and buffers packets it heard. A node also informs its neighbors which packets it has overheard. This allows nodes to know roughly what packets are available at each neighbor (i.e., "who has what?"). Knowing "who has what" in the neighborhood, a node examines its pending outgoing packets and decides how to form output mixture packets, with the objective of most efficiently utilizing the medium.

These prior studies result in a link layer enhancement scheme in the networking stack. As illustrated in Figure 2, the local mixing engine sits above the MAC layer (e.g., 802.11) and below the network layer. Given the routing decisions, the local mixing engine tries to identify opportunities for physical piggybacking. Experimental results in [2] demonstrate the usefulness of local mixing in improving the link layer efficiency. The gain of this technique, however, critically depends on the traffic pattern in the network. This motivates the following question: Can we make intelligent routing decisions that maximize the benefits offered by the local mixing engine?

In this paper we focus on wireless mesh networks (i.e., static multi-hop wireless networks), which find useful applications

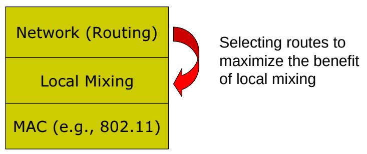

Fig. 2. The big picture. The local mixing engine sits between the network layer and the MAC layer and thus presents an enhanced link layer to the network layer. This paper develops routing solutions that can better take advantage of the local mixing engine.

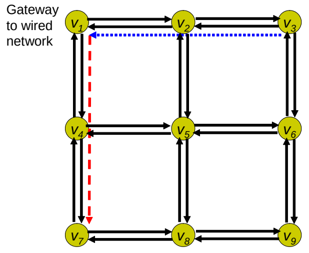

Fig. 3. An example mesh networking scenario. There are 9 mesh access points and v_1 is a gateway to the wired network. The connectivity graph is shown in the figure. Assume currently there are two long-term background flows, $v_3 \rightarrow v_2 \rightarrow v_1$ and $v_1 \rightarrow v_4 \rightarrow v_7$. Suppose we want to find a good routing path from v_1 to v_9.

in metro-area public Internet access, community wireless networks, and transient networks (e.g., disaster relief). State-of-the-art routing protocols for wireless mesh networks have traditionally been based on finding shortest paths under certain cost metrics. The simplest path metric is the hop count along the path. Later on, various link quality metrics have been proposed for static wireless mesh networks. These metrics include for example, the per-hop round-trip time (RTT), the expected transmission count (ETX) [3], and the expected transmission time (ETT) [4].

A natural thought is to modify the link metrics to take into account the effect of the local mixing engine in reducing the transmissions over the air. This, however, is not straightforward. Consider the example setting illustrated in Figure 3. There are two long-term flows in the network, $v_3 \rightarrow v_2 \rightarrow v_1$ and $v_1 \rightarrow v_4 \rightarrow v_7$. We want to find a good routing path from v_1 to v_9. Due to the existence of the local mixing engine, the route $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_9$ is a good solution because the packets belonging to this new flow can be mixed with the packets belonging to the opposite flow $v_3 \rightarrow v_2 \rightarrow v_1$, resulting in improved resource efficiency. To encourage using such a route, can link $v_2 \rightarrow v_3$ announce a lower cost? There are some issues in doing so, because a packet from v_5 that traverses $v_2 \rightarrow v_3$ may not share a ride with a packet from v_3 that traverses $v_2 \rightarrow v_1$, although a packet from v_1 that traverses $v_2 \rightarrow v_3$ can.

We see from this example that in the presence of the local mixing engine, assessing the channel resource incurred by a packet transmission requires some context information about where the packet arrives from. For example, we can say that given the current traffic condition, the cost for sending a packet from v_2 to v_3 that previously arrives from v_1, is smaller. The key observation here is the need to define link cost based on some context information. More generally, this motivates the concept of a *Markovian metric*.

A Markovian metric introduces context information into the cost modelling. The cost of sending a packet (or a stream of packets) across a link is now allowed to depend on where the packet (or the stream) arrived from. The cost of a path is modelled as the cost of the first hop plus the cost of the second hop conditioned on the first hop, and so on. Due to this decomposition structure, the dynamic programming principle still applies and thus finding the shortest path with a Markovian metric can still be done in polynomial time. In a practical network, support for the Markovian metric can be added easily into an existing routing framework that uses a conventional routing metric.

The concept of the Markovian metric is explained in a general context in Section II. After reviewing local mixing in Section III, in Section IV we examine how to design a specific Markovian metric to maximize the benefit of local mixing. Section V presents the experimental results.

II. Markovian Metric

A conventional routing metric models the cost of a path as the sum of the costs on the individual links. A Markovian metric introduces context information into the cost modelling. The cost of sending a packet (or a stream of packets) across a link is now allowed to depend on where the packet (or the stream) arrived from.

Definition 1 (Markovian Metric): Consider a path $\mathcal{P} = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_k$. A *Markovian metric* models the cost of a path as the sum of the conditional costs on the links:

$$\begin{aligned} \mathsf{cost}(\mathcal{P}) \triangleq & \mathsf{cost}(v_0 \rightarrow v_1) + \mathsf{cost}(v_1 \rightarrow v_2 | v_0 \rightarrow v_1) + \ldots \\ & + \mathsf{cost}(v_{k-1} \rightarrow v_k | v_{k-2} \rightarrow v_{k-1}). \end{aligned} \tag{1}$$

Here $\mathsf{cost}(b \rightarrow c | a \rightarrow b)$ denotes the cost of sending a packet from b to c, conditioned on that the packet arrived at b via a.

The conventional routing metric can be viewed as a special case of the Markovian metric where all the conditional link costs are equal to their unconditional counterparts. The decomposition relation (1) is reminiscent of the decomposition of the joint probability distribution of random variables forming a Markov chain into a product of the conditional probabilities. Thus, a Markovian metric to an unconditional metric is like a Markov chain to a memoryless sequence of random variables.

A. The Dot Graph Representation

Suppose we are given a set of unconditional link costs W_{uncon} and a set of conditional link costs W_{con}. For ease in notation, we use $w_{i,j}$ to denote an unconditional link cost $\mathsf{cost}(v_i \rightarrow v_j)$ and $w_{i,j,k}$ to denote a conditional link cost $\mathsf{cost}(v_j \rightarrow v_k | v_i \rightarrow v_j)$. We now discuss a graphical representation of these costs, which we call the *dot graph*.[1] Denote the original graph by G and the resulting dot graph by $\dot{G}$. For the example network in Figure 3, the graphical representation of the link costs is illustrated in Figure 4. In this example, we assume each unconditional link cost is 1 and there are two conditional costs, $w_{1,2,3} = 0.5$ and $w_{7,4,1} = 0.5$. For instance, here $\mathsf{cost}(v_2 \rightarrow v_3 | v_1 \rightarrow v_2) < \mathsf{cost}(v_2 \rightarrow v_3)$

[1] A dot graph is a generalization of the *line graph*, a well known representation in graph theory. Given a graph $G = (V, E)$, the line graph $L(G)$ is a graph whose node set is E and whose edge set comprises the set of all ordered edge pairs (e, e') such that e's end point is equal to e''s start point. The dot graph, however, may contain significantly less edges than the line graph, if there are only a few conditional link costs.

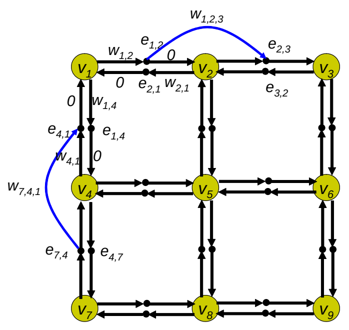

Fig. 4. The dot graph representation of a collection of conditional and unconditional link costs, for the example graph in Figure 3.

because a packet from v_2 to v_3 that arrived from v_1 can be mixed with the existing traffic in the flow $v_3 \to v_2 \to v_1$.

First, we introduce a dot for each directed link in the original graph, which "splits" the original link into two halves. Note that there is a one-to-one correspondence between the links in the original graph G and the dots in $\dot{G}$. With slight abuse of notation, we refer to these dots as the names for the links in G; for example, the dot that splits the link from v_1 to v_2 is referred to as $e_{1,2}$. Therefore, $\dot{G}$ has $|V(G)| + |E(G)|$ nodes.

Second, for each conditional link cost $\text{cost}(v_j \to v_k | v_i \to v_j)$ in the given set W_{con}, we draw an edge from the dot e_{ij} to the dot e_{jk}. These edges, together with the edges generated by splitting the original links, constitute the edge set of the dot graph. To distinguish from the edges in the original graph, we call an edge in the dot graph a *wire*. Therefore, $\dot{G}$ has $2|E(G)| + |W_{\text{con}}|$ wires.

Third, we associate a cost label with each wire in $\dot{G}$. The cost of a wire from a physical node $v_i \in V(G)$ to a dot $e_{i,j} \in V(\dot{G})$ is the given unconditional cost of the link, w_{ij}. The cost of a wire from a dot $e_{i,j} \in V(\dot{G})$ to a physical node $v_j \in V(G)$ is 0. The cost of a wire from a dot $e_{i,j} \in V(\dot{G})$ to another dot $e_{j,k} \in V(\dot{G})$ is $w_{i,j,k}$, the given conditional cost of the link.

In general, we allow the coexistence of a conditional cost and unconditional cost for the same link, e.g., $\text{cost}(b \to c | a \to b)$ and $\text{cost}(b \to c)$. We assume the conditional link cost is always less than or equal to its corresponding unconditional link cost. This is without loss of generality because we can always define the unconditional cost on a link as the maximum of the corresponding conditional link costs. The meaning is intuitive: The unconditional link cost represents a conservative estimate of the cost incurred; given further context information, the cost may be lower. For example, in Figure 4, $w_{1,2,3} = 0.5 < w_{2,3} = 1$; intuitively, there is a "short cut" from $e_{1,2}$ to $e_{2,3}$.

B. Minimum Cost Routing Using a Markovian Metric

Intuitively, the dot graph models the existence of short cuts at various places in the network. It is easy to see that a path in the dot graph $\dot{G}$ maps into a route in the original network and the cost of the route is just the total cost along the path in $\dot{G}$. For instance, consider a path $\mathcal{P}_1$ from v_1 to v_9 in $\dot{G}$:

$$v_1 \to e_{1,2} \to e_{2,3} \to v_3 \to e_{3,6} \to v_6 \to e_{6,9} \to v_9.$$

This corresponds to the physical route $v_1 \to v_2 \to v_3 \to v_6 \to v_9$. The cost of the path is: $w_{1,2} + w_{1,2,3} + w_{3,6} + w_{6,9} = 3.5$. In comparison, consider the path $\mathcal{P}_2$:

$$v_1 \to e_{1,4} \to v_4 \to e_{4,7} \to v_7 \to e_{7,8} \to v_8 \to e_{8,9} \to v_9.$$

This corresponds to the physical route $v_1 \to v_4 \to v_7 \to v_8 \to v_9$, which has a cost of 4. Therefore, $\mathcal{P}_1$ is better than $\mathcal{P}_2$.

More generally, to find the minimum cost route between two physical nodes, we just need to apply a shortest path algorithm over the dot graph. For example, with Dijkstra's algorithm, the complexity is $O(|V(\dot{G})|^2)$. Note that we can remove the unnecessary dots in $\dot{G}$; specifically, we can introduce a dot $e_{i,j}$ only if there is a need to express a related conditional link cost, i.e., when W_{con} includes a cost $w_{i,j,*}$ or $w_{*,i,j}$. (Here we use the symbol $*$ as a wildcard.) By doing so, the number of vertices in the dot graph can be reduced to $O(|V(G)| + \min\{|E(G)|, 2|W_{\text{con}}|\})$.

Proposition 1 (Min-Cost Routing w/ Markovian Metric): Given a set of unconditional link costs W_{uncon} and a set of conditional link costs W_{con}, the minimum cost routing from a source node s to a sink node t can be found by running a shortest path algorithm over the dot graph. This can be done in complexity $O\left((|V(G)| + \min\{|E(G)|, 2|W_{\text{con}}|\})^2\right)$.

C. Adaptive Routing in a Practical Network

In a practical network, routing decisions are made to reflect the changes in the topology and sometimes the traffic as well. The dot graph representation makes it particularly easy to see how to modify the existing routing protocols for a Markovian metric system. Essentially, a physical node v_i needs to play several characters in a distributed routing algorithm, including those of its neighboring dots who do not physically exist. In particular, we could divide the computation responsibility as follows. Let each physical node v_i be responsible for the outgoing wires of v_i and its incoming dots $e_{*,i}$. For example, in Figure 4, physical node v_2 implements the computation involving wires $e_{*,2} \to v_2$, $v_2 \to e_{2,*}$, $e_{1,2} \to e_{2,3}$.

Routing protocols can be classified as either proactive or reactive. Proactive protocols attempt to maintain up-to-date routes within the network, so that a route is readily available when a packet needs to be forwarded. Reactive protocols, on the other hand, do not maintain up-to-date topological information about the network. When a source needs to find a route to a destination node, the source initiates a route discovery procedure, typically done by flooding. In the following we examine how to support a Markovian metric in representative routing algorithms, starting with the proactive protocols.

1) Link State Routing: Example protocols in this category include OLSR [5] and LQSR [3]. In link state routing, each router measures the cost to each of its neighbors, constructs a packet including these measurements, sends it to all other routers, and computes the shortest paths locally. In essence, the complete topology and the link costs are experimentally measured and distributed to each router. Then each router can locally run a shortest path algorithm to decide the routes.

To support a Markovian metric, minimal changes are needed in a link state routing system. Each router can just measure the unconditional and conditional costs for the links/wires it is responsible for and broadcast the measurements to all other routers. Take node v_2 in Figure 4 as an example. Here v_2 needs to measure the unconditional costs to each neighbor, as well as the conditional costs of the form $\mathsf{cost}(e_{i,2} \to e_{2,j})$.

2) Distance Vector Routing: An example protocol in this category is DSDV [6]. In distance vector routing (also known as the Bellman-Ford algorithm), each router maintains a table (i.e., a vector) giving the best known cost to reach each destination and which interface to use to get there. There tables are updated by exchanging information with the neighbors.

Let us start with a first-cut solution. Once every T milliseconds each router sends each neighbor a list of its estimated minimum cost to reach each destination. Since each physical node v_i is also playing the roles of its incoming dots, a first-cut implementation will aggregate the tables from v_i and its incoming dots. For example, consider node v_2 in Figure 4. It is also responsible for playing the roles of $e_{1,2}$, $e_{3,2}$, $e_{5,2}$. Thus the aggregated table will include one minimum cost from $e_{i,2}$ to v_j, for $i = 1, 3, 5$ and all other destinations. Denote $\mathsf{cost}(e_{i,2} \rightsquigarrow v_j)$ the estimated minimum cost to reach destination v_j. Note that in this case,

$$\mathsf{cost}(e_{3,2} \rightsquigarrow v_j) = \mathsf{cost}(e_{5,2} \rightsquigarrow v_j) = \mathsf{cost}(v_2 \rightsquigarrow v_j).$$

Similar scenarios may happen whenever the conditional cost cannot lead to a lower route to the given v_j. In these scenarios, sending both $\mathsf{cost}(e_{3,2} \rightsquigarrow v_j)$ and $\mathsf{cost}(e_{5,2} \rightsquigarrow v_j)$ is wasteful. To remove such redundancy, we include $\mathsf{cost}(v_2 \rightsquigarrow v_j)$, and $\mathsf{cost}(e_{*,2} \rightsquigarrow v_j)$ only if it is lower than $\mathsf{cost}(v_2 \rightsquigarrow v_j)$. This results in an improved implementation.

3) Source-Routed On-Demand Route Discovery: Example protocols in this category include DSR [7] and DSR with ETX [8]. Here a source node that wishes to discover a route to a destination broadcasts a route request packet. This route request contains the address of the destination, the source node's address, and a unique identifier. When a node forwards a route request, it appends not only its own address, but also information about the related link costs. Specifically, suppose the route discovery packet has traversed a path $v_0 \to v_1 \to \ldots \to v_k$ to v_k. Then v_k sends a route request that contains:

$$\mathsf{cost}(v_0 \to v_1),\ \mathsf{cost}(v_1 \to v_2 | v_0 \to v_1),\ \ldots,\\ \mathsf{cost}(v_{k-1} \to v_k | v_{k-2} \to v_{k-1}), \qquad (2)$$

and also $\mathsf{cost}(v_k \to v_j | v_{k-1} \to v_k)$ for neighbor v_j (because v_j may not know how to compute this). If the physical medium is broadcast medium, then v_k can send a single broadcast packet that contains (2) and $\mathsf{cost}(v_k \to v_j | v_{k-1} \to v_k)$ for all neighbors v_j. When a node v_k receives a request it has already forwarded, it forwards it again only if the accumulated cost to a neighbor v_j is better than the best which it has already forwarded with the request ID. The link metrics are included in the route replies sent back to the source.

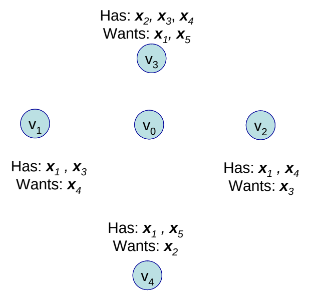

Fig. 5. The local mixing problem is about optimizing the formation of mixture packets at a local wireless router, knowing "who has what" and "who wants what" in a neighborhood.

4) Hop-by-Hop On-Demand Route Discovery: An example protocol in this category is AODV [9]. Similar to the source-routed on-demand discovery, here the route request initiated by the source is flooded through the network. However, each route request no longer contains the entire route. Instead, each route request received by v contains only the cumulative cost from s to v. Each node v maintains for each incoming neighbor u the minimum accumulated cost from s via u to v, denoted by $\mathsf{cost}(s \rightsquigarrow u \to v)$. When a node v receives a request, it forwards the request to a neighbor v' only if the total cost to v' is reduced. The routing table entries to the destination will be created after the destination sends back a route reply to the source, following the sequence of best previous hops.

III. Local Mixing: A Review

The gist of the packet exchange example in Figure 1 is as follows: At certain moment, v_1 has $\boldsymbol{x}_1$; v_2 has $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$; v_3 has $\boldsymbol{x}_2$. Thus a mixture packet $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$ can be demixed into $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively at v_3 and v_1. In the following we use the name *source packet* to refer to a packet such as $\boldsymbol{x}_1$ which was originally generated by a source node, and the name *mixture packet* to refer to a packet such as $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$.

More generally, we may have a situation illustrated in Figure 5. A wireless router knows the source packets each neighbor has (i.e., "who has what"). It also knows "who wants what" because these are the packets in its output queue that it is supposed to forward to the neighbors. Then it can decide locally how to optimize the formation of mixture packets. A heuristical approach for generating the mixture packets is used in [2], which takes the packet at the head of the output packet queue, and steps through the packet queue to greedily add packets to the mixture, while ensuring the neighbors can successfully demix. For example, in Figure 5, there are five packets in the output queue, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5$; assume a lower indexed packet is an earlier packet. Then the greedy procedure will use three transmissions: $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$, $\boldsymbol{x}_3 \oplus \boldsymbol{x}_4$, $\boldsymbol{x}_5$. However, there is a better solution: $\boldsymbol{x}_1 \oplus \boldsymbol{x}_3 \oplus \boldsymbol{x}_4$, $\boldsymbol{x}_2 \oplus \boldsymbol{x}_5$. A mathematical abstraction of the optimized formation of the mixture packets – the *local mixing problem* – is studied by Wu et al. [10] from an information theoretic point of view. Under the assumption that each neighbor discards the received packets that are polluted by sources it does not have or want, the optimal mixing is characterized.

Why would a node have packets meant for others? In Figure 1, node v_1 has packet $\boldsymbol{x}_1$ because it is the previous hop of $\boldsymbol{x}_1$. More generally, due to the broadcast nature of the wireless medium, neighboring nodes may overhear packets. For example, in Figure 5, packet $\boldsymbol{x}_1$ may follow a path $\ldots v_4 \to v_0 \to v_3 \ldots$; v_1 and v_2 may have overheard $\boldsymbol{x}_1$ when v_4 sent it to v_0.

How does a node get to know "who has what"? First, note that a node can obtain some partial information about its neighbors' data availability in a passive fashion. For example, node v_2 may infer that node v_1 holds packet $\boldsymbol{x}_1$ if v_1 recently received packet $\boldsymbol{x}_1$ or a mixture packet involving $\boldsymbol{x}_1$ from v_1, or if v_2 recently heard v_1 acknowledging the receipt of packet $\boldsymbol{x}_1$. This suffices for packet exchanges such as Figure 1.

Passive inference does not incur any additional overhead. However, using passive inference alone, a node may only obtain a limited view of the neighbors' data availability. Katti et al. [2] extended this by proposing two techniques to obtain more information about local data availability: (i) Let each node explicitly announce the packets it currently has to its neighbors; (ii) let a node guess whether a neighbor has overheard a packet using information about the channel reception. In the former, each node can periodically compose *reception reports* to announce the packets it has overheard. The reception reports may also be piggybacked with ordinary packets. To implement guessing, nodes conduct measurement about the packet success probabilities to its neighbors and exchange the measurement results in the neighborhood. Such measurement and report functionality may already be needed by a routing protocol based on the expected transmission count (ETX) [8]. The guessing technique of [2] can be explained via Figure 5. Suppose v_4 sends a source packet $\boldsymbol{x}_1$ to v_0 without mixing; suppose v_0 knows that v_1 can receive a packet from v_4 with probability 0.8. When v_0 received $\boldsymbol{x}_1$ sent by v_4, v_0 can infer that v_1 has overheard the packet with probability 0.8. Guessing may result in a more up-to-date knowledge about "who has what"; however, if the guess is wrong, the neighbor may fail to demix a packet intended for it.

A mixture packet may be intended for more than one receiver. Due to the limited collision avoidance mechanism for broadcast in 802.11, the mixture packet is sent as a unicast packet addressed to one of the receivers [2]. A consequence of this is that the sender cannot be sure whether the other intended receivers received the packet reliably. We call such an issue the "missing ACKs" problem. The missing ACKs problem can be addressed by explicitly generating ACKs in addition to the ACK in 802.11 MAC [2]. Nodes can keep track of the packets that were sent but have not yet been acknowledged and retransmit packets after time-out.

Next, we briefly review the key data structures and operations in implementation.

Each packet has a variable length header that includes: (i) the IDs of the source packets being mixed and their respective receivers, (ii) some piggybacked ACKs, (iii) some piggybacked reception reports. If no data packets were sent after a certain amount of time, then a dedicated control packet containing ACKs and reception reports is broadcast.

Each node maintains three separate buffers, `OverheardBuffer`, `ReceivedBuffer`, `SentBuffer`, holding respectively the source packets that the node overheard, received, or sent. Upon receiving a packet, the packets in these three buffers are used for demixing. Reception reports describe new content in the `OverheardBuffer`. ACKs describe new content in the `ReceivedBuffer`.

Each node maintains a `WhoHasWhatTable` whose entries are of the form "node v_i has source packet $\boldsymbol{x}_j$ with probability p". Upon receiving a packet, the `WhoHasWhatTable` is updated according to the local mixing header. If the received packet is a source packet, guessing is also performed based on the measured channel reception probabilities.

After a packet is sent, the ingredient source packets are moved from the output queue into the `SentBuffer`. In addition, timer events are inserted so that the sent packets will be moved back to the output queue for retransmission if the ACKs does not arrive after a certain time threshold.

IV. Markovian Metric for Local Mixing

In this section we examine how to use a Markovian metric to maximize the benefit of local mixing. The central issue here is to properly define the link costs and compute them. Let us begin with the unconditional link metrics. A popular link quality metric in the literature is the expected transmission count (ETX) [8]. This metric estimates the number of transmissions, including retransmissions, needed to send a unicast packet across a link. It is obtained by measuring the loss probabilities of broadcast packets between pairs of neighboring nodes.

The ETX metric can be viewed as a characterization of the amount of resource consumed by a packet transmission. With the local mixing engine, several packets may share a ride in the air. Naturally, the passengers can share the airfare. In effect, each participating source packet is getting a discount. Such discount, however, cannot be accurately modelled by an unconditional metric such as ETX, because the applicability of the discount depends on the previous hop of the packet. We propose a conditional link metric called the *expected resource consumption* (ERC), which models the cost saving due to local mixing. Consider a packet sent in the air. If it is a mixture of k source packets, then each ingredient source packet is charged $\frac{1}{k}$ the resource consumed by the packet transmission. The resource consumed by the transmission could be measured in terms of, e.g., air time, or consumed energy.

A. Computation of Expected Resource Consumption (ERC)

We now explain how to compute the ERC. Each node maintains a `WireInfoTable`. Each row of the table contains the measured statistics about a wire, say $e_{i,j} \to e_{j,k}$, which crosses the current node v_j. The packets forwarded by the current node can be classified into categories associated with the wires. For each wire category, we collect the total number of packets sent and the total resource consumed in a sliding time window. The total resource consumption is obtained by adding the resource consumption for each sent packet. A simple charging model is used in our current implementation. For example, if a source packet across wire $e_{i,j} \rightarrow e_{j,k}$ is sent in a mixture of 3 packets, we set the resource consumption of this source packet as $1/3$ of the ETX of link $e_{j,k}$.[2]

To implement the sliding window computation efficiently, we quantize the time axis into discrete slots of equal length. We use a sliding window of N slots. For each wire, we maintain a circular buffer of N bins; at any time, one of the N bins is active. At the start of each slot, we shift the active bin pointer once and clear the statistics in the new active bin. Each time a packet is transmitted in the air, we update the statistics in the current active bin accordingly.

To evaluate the conditional link metric for a certain wire $e_{i,j} \rightarrow e_{j,k}$, we first obtain the ERC for each slot, say n, as:

$$\mathsf{erc}_n := \frac{\text{Resource consumed by pkts sent in slot } n}{\text{\# of packets sent in slot } n}. \quad (3)$$

Then we compute the ERC for the wire as the weighted average of the ERCs for the slots:

$$\mathsf{ERC} := \sum_{n=0}^{N-1} \alpha_n \mathsf{erc}_n; \;\; \alpha_n = \alpha^{N-1-n} \left(\frac{1-\alpha}{1-\alpha^N} \right). \quad (4)$$

Here the parameter α is the forgetting factor for old observations. Old observations receive lower weights.

What if few or no packets were sent during a certain slot across the wire? We propose to conduct experiments using probing packets. The experiments are done via simulation, without disturbing the existing traffic. Specifically, we maintain a virtual output queue in addition to the output queue. Whenever we insert an actual packet into the output queue, we insert it into the virtual output queue as well. Whenever the MAC layer asks for an output packet, we also invoke the mixing algorithm on the virtual output queue to generate a mixture packet. (In fact, for the virtual output queue, we just need to decide on how to mix; no actual mixing is needed.) The result of the mixing decision is only used to update the statistics for the wires being tested.

B. Interpretation of the ERC Metric

The ERC link metric estimates the local resource consumption while considering the effects of local mixing. The resulting Markovian path metric thus estimates the total resource consumed by the route. Resource efficiency is an important performance metric for resource-constrained wireless networks, e.g., sensor networks. It is also well aligned with the overall system throughput in interference limited wireless networks, because resource saved in one flow may be used to improve the throughput for other flows.

The ERC metric emphasizes the maximization of network-wide utility more than the maximization of the individual flow utility (e.g., latency). By using a weighted combination of two Markovian metrics, one capturing the network-wide utility and the other the individual flow utility, we can adjust the balance between the two objectives. For example, occasionally there may be a longer route with a lower resource consumption than a shorter route; in this case, we may define a Markovian metric as the weighted combination of the ERC metric and the ETT (expected transmission time) metric:

$$\mathsf{cost}(e_{jk}|e_{ij}) \triangleq \beta \mathsf{ERC}(e_{jk}|e_{ij}) + (1-\beta)\mathsf{ETT}(e_{jk}), \quad (5)$$

where the ETT metric is computed by dividing the ETX metric by the raw link data rate (see [4]). It is up to the applications or certain network rule-makers to decide how to balance these two considerations.

C. Route Stabilization via Randomized Route Holding

In order to model the resource reduction due to local mixing, the ERC takes the traffic load into account. Could this cause oscillation in the routing decisions? We provide some intuitive reasoning below. Recall that the discounts offered by the local mixing engine exist only when the flows cross in certain ways. Stated alternatively, the advertised discounts have restrictions and hence only a few qualifying flows may find them attractive. Since the discounts benefit all the flows whose packets are being mixed, there is incentive for flows to route in a certain cooperative manner that are mutually beneficial. Presumably, if the flows try such a mutually beneficial arrangement for some time, they will confirm the discounts and tend to stay in the arrangement. Such an arrangement is analogous to the Nash equilibrium in game theory, where no player wants to deviate from its current strategy given all other players' strategies. However, a complication is that there can be more than one equilibrium. We want the flows to make dynamic decisions that eventually settle down to one equilibrium. To facilitate this, we propose the following strategy. To prevent potential route oscillations, we require each flow to stay for at least T_{hold} duration after each route change, where T_{hold} is a random variable. The randomization of the mandatory route holding time T_{hold} is used to avoid flows from changing routes at the same time. In addition, after the mandatory route holding duration, the node switches to a new route only if the new route offers a noticeably smaller total cost.

V. Evaluation of Markovian Metric Routing

We implemented the local mixing engine, a link-state source routing protocol, the support for Markovian metric routing, and the support for the ERC link matric in Qualnet 3.9.5, a widely used event-driven simulator for wireless networks. The resulting integrated system will be referred to as Markovian Metric Source Routing (MMSR) in the following. We implemented local mixing for MMSR as a shim layer between the routing and MAC (802.11) layers. The routing layer exposes the packet delivery probabilities to the local mixing layer (for guessing "who has what"). The local mixing layer exposes the ERC statistics to the routing layer.

The basic routing framework can be viewed as a simplified version of LQSR [3], a source-routed link-state protocol that supports link quality (unconditional) metrics. Similar to LQSR, every 2 seconds, each node sends a message that carries

[2]We could also use ETT [4] in lieu of ETX. The ETT metric is equal to the ETX metric divided by the raw link rate. When the radios operate at the same physical link rate, the two metrics are essentially equivalent.

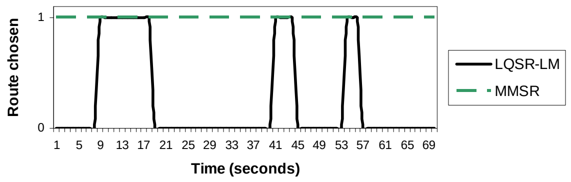

Fig. 6. Scenario 1: Mixing performance. State 1 denotes an optimal mixing route whereas 0 denotes a suboptimal mixing route.

information about the links from this node; this message floods throughout the network. Such a message, called the `WireInfo` message, describes the unconditional and conditional link costs. Here ETX is used as the unconditional link cost, and ERC is used as the conditional link cost. To compute ERC, we use $N = 10$ slots, each of length 0.5s. If there are 25 or more packets crossing a wire, then the ERC of the slot is computed as the average according to (3). Otherwise, a simplified rule (instead of maintaining the virtual output queue) is applied in our current implementation. Specifically, for a wire $e_{ij} \rightarrow e_{jk}$, if there are less than 25 packets crossing, then we examine the number of unmixed packets y in the reverse wire $e_{kj} \rightarrow e_{ji}$. If $y \geq 25$, then we set the ERC as 0.5 of the ETX (50% discount); otherwise, we set the ERC as the ETX (no discount). The ERCs of the slots are combined with exponentially delaying weights, using $\alpha = 0.8$ in (4).

A conditional link cost, say $\mathsf{cost}(e_{jk}|e_{ij})$, is included in the message only if it is at least 5% less than the ETX for e_{jk}. Each node maintains a `LinkCache` that stores its current picture of the dot graph. The `LinkCache` is updated upon receiving each `WireInfo` message. Since the conditional link costs are load-dependent and may not be periodically reported, stale conditional link cost entries are removed after time-out.

The route stabilization techniques of Section IV-C are used. The mandatory holding time T_{hold} is drawn uniformly from $[1, 3]$ (sec). When a source packet is just generated, if the current mandatory holding time has elapsed, then the node looks for the optimal route in the dot graph. If the cost of the new route is at least 5% lower than that of the old route, then the new route is adopted. Otherwise, the old route is still used.

The simulations use the 802.11a MAC and a realistic signal propagation model. All radios operate at a nominal physical layer rate of 54Mbps. We use CBR flows in the evaluation.

A. Impact on Local Mixing: MMSR vs. LQSR+LM

We first demonstrate the performance of local mixing in MMSR compared to LQSR+LM, which uses the unconditional ETX metric for routing. This evaluation is performed in a 9-node grid network topology.

1) Mixing performance evaluation: Consider the network shown in Figure 3 with an existing flow $v_3 \rightsquigarrow v_1$. After 3 seconds, v_1 initiates a flow to v_9. There are many possible routes that this flow can take, but only one ($v_1 - v_2 - v_3 - v_6 - v_9$) is optimal in terms of the resource consumption. Figure 6 shows how often this optimal route is chosen by LQSR+LM and MMSR, respectively. As the results show, MMSR causes the flow $v_1 \rightsquigarrow v_9$ to choose the mutually beneficial route

Scenario	Mixed (MMSR)	Mixed (LQSR+LM)
S1	20,366	1,593
S2	39,576	24,197

TABLE I
GAIN FROM MMSR COMPARED TO LQSR+LM.

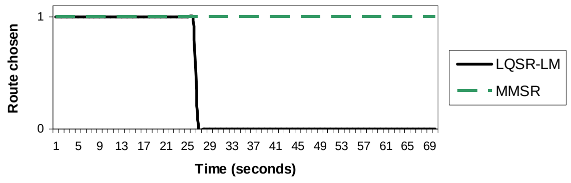

Fig. 7. Scenario 2: Mixing performance. State 1 denotes an optimal mixing route whereas 0 denotes a suboptimal mixing route.

$v_1 - v_2 - v_3 - v_6 - v_9$ and this route is chosen 100% of the time, resulting in maximized mixing. Intuitively, the existing flow $v_3 \rightsquigarrow v_1$ creates a discount in terms of the conditional ERC metric in the opposite direction, which attracts v_1 to choose route $v_1 - v_2 - v_3 - v_6 - v_9$. Once the flows start in both directions, they stay together and mix because both see discounts. As shown in Table I, MMSR increases the number of mixed packets in this scenario by $12\times$ in comparison to LQSR+LM.

Next, we examine whether mixing can take place effectively over longer routes. We thus now consider an existing flow $v_9 \rightsquigarrow v_1$ and examine the choice made by a new flow $v_1 \rightsquigarrow v_9$ originating 3 seconds later. The results are shown in Figure 7. Notice that LQSR+LM only chooses the optimal route for less than half of the time, whereas MMSR quickly locks on to the opposite route due to the conditional discounts. In this case, mixing is possible at 3 separate nodes and thus the number of mixed packets increase.

Note that MMSR is crucial when specific small opportunities exist for mixing (e.g. at one node in scenario S1). In this case it is unlikely that LQSR+LM will be able to exploit such opportunities. In scenario S2, mixing is possible at many hops and by pure random choice, LQSR+LM can also locate a non-trivial number of mixing opportunities. However, MMSR still provide a 63% increase in mixed packets over LQSR+LM.

2) Stabilizing routes: As mentioned in Section IV-C, an important but subtle requirement for a Markovian metric is to avoid potential route oscillations. Occasionally, if two flows that can potentially mix (such as $v_9 \rightsquigarrow v_1$ and $v_1 \rightsquigarrow v_9$) start right at the same time and choose different routes, they could be attracted to each other repeatedly and potentially oscillate. As described in Section IV-C, we deal with this by holding each route for a random amount of time. Figure 8 shows the route evolution when both flows start exactly together. Due to the randomized route holding strategy, the oscillation is resolved quickly (in 7 seconds in this case), allowing the two flows to mix effectively. Note that such pathological cases are unlikely to occur frequently; nonetheless, the randomized route holding strategy ensures correct operation even in such pathological cases.

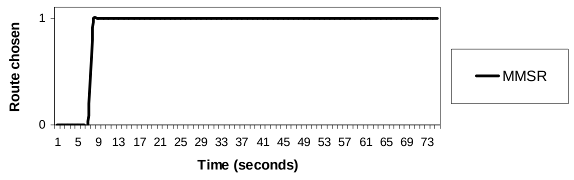

Fig. 8. MMSR randomization provides robustness to oscillations. State 1 denotes an optimal mixing route whereas 0 denotes a suboptimal mixing route.

3) Summary: In summary, we find that MMSR reliably chooses the optimal mixing route and the ERC discounting system, together with the randomized route holding strategy, facilitate flows to settle down into a mutually beneficial equilibrium in a distributed and dynamic manner.

B. Impact on Overall Performance

We next study how MMSR compares to LQSR+LM as well as basic LQSR in terms of the overall performance.

1) Grid Network Scenario: We continue with the 9-node grid network scenario and evaluate the performance with three flows: (1)$v_9 \rightsquigarrow v_1$, (2) $v_1 \rightsquigarrow v_9$, and (3) $v_3 \rightsquigarrow v_1$. Each flow begins randomly between 50–60 seconds into the simulation. We evaluate the performance of LQSR, LQSR+LM and MMSR for this scenario for different input loads. The results are depicted in Figure 9. It is observed that LQSR cannot sustain the throughput imposed by the input flows to the network as the load increases. MMSR provides significant throughput gains compared to LQSR (up to 47%) and LQSR+LM (up to 15%). This is because not only does MMSR allow subsequent flows to mix with existing flows, it explicitly tries to maximize mixing. In contrast, without the Markovian metric routing, flows mix essentially by chance. In the example, the flow 1-2-3-6-9 is mixed with 9-6-3-2-1 with MMSR, due to the mutually beneficial discounts enjoyed by both flows.

Figure 10 gives the amount of *resource* saved by using MMSR. MMSR consistently provides reduction of packet transmissions of over 10,000 packets across a wide variety of traffic demands. Indeed, as discussed in Section IV-B, MMSR functions by directly trying to minimize the resource usage. Another observation from Figure 10 is that the saved transmissions reduce as network load increases. This is counter-intuitive since more packets should indicate more mixing opportunities. However, this occurs because of the *capture* effect in the 802.11 MAC layer which is amplified at high loads. Due to this, packets from only one node (the capturing node) fill queues for large durations of time without allowing other traffic to come in. This reduces mixing opportunities at high load. This problem can potentially be addressed through a better MAC layer design that avoids capture.

We also considered a pathological case to stress MMSR, where all three flows start exactly at the same time. We found that even in this case, throughput gains were still significant, although lower. The convergence time of MMSR is quite fast (on the order of 4–7 seconds); as a result, there is some time for long-lived flows to fully benefit from mixing.

Our results exemplify that local mixing itself cannot provide

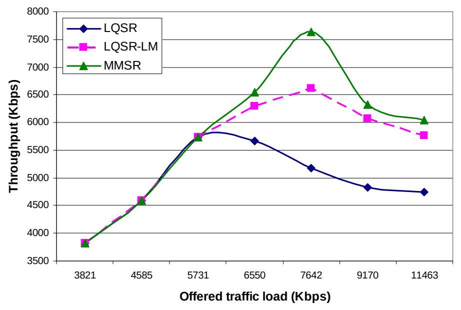

Fig. 9. Throughput comparison of MMSR, LQSR+LM and LQSR.

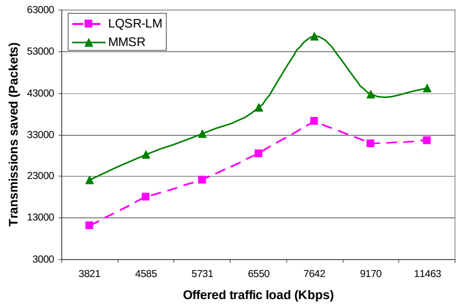

Fig. 10. Transmissions saved through mixing in MMSR and LQSR+LM.

the best achievable performance and a protocol such as MMSR can help local mixing achieve its potential.

2) Office Mesh Scenario: We now consider how MMSR performs in a wireless office scenario [11]. Figure 11 gives the topology, which is the actual topology of the mesh testbed at Microsoft Research (with 42 nodes). We set nodes 1–7 as servers offering different services and the remaining 35 nodes as clients. Note that a client may contact servers other than the closest one, because many servers provide a specialized service (e.g., Email/Source Control/Domain Controller). Eriksson et al. [11] observed that the traffic in an office network is predominantly download (i.e., server $\rightsquigarrow$ client). Taking this observation into account, we simulated a traffic pattern consisting of 7 download connections, one from each server. Each connection starts and ends at random times in the simulation.

The results are depicted in Table II. The results show that MMSR outperforms LQSR+LM by 16% and LQSR by 17% in throughput. MMSR also saved 15% of the transmissions (i.e., MMSR sent m source packets using $85\%m$ transmissions).

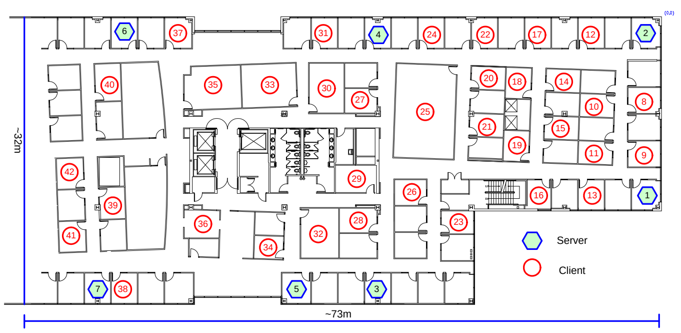

Fig. 11. The topology of the mesh testbed.

Scenario	Avg. Client Throughput	% Transmissions Saved
LQSR	865 Kbps	-
LQSR+LM	877 Kbps	3%
MMSR	1014 Kbps	15%

TABLE II
MMSR PERFORMANCE IN AN OFFICE MESH SCENARIO.

Note that the extent of gain depends on the traffic patterns and the scale of the network. A larger network with more hops would provide enhanced mixing and limit interference between flows. Additionally, if clients generate both uploads and downloads this can also provide additional gains from local mixing. At any rate, MMSR can exploit mixing opportunities if and when they occur.

C. Summary of Results

In summary, our evaluation of MMSR has shown that Markovian metrics are useful and practically applicable. The results showed that a Markovian metric can significantly improve the benefit of an advanced link-layer technique: local mixing. We found that MMSR can tolerate pathological cases (that can potentially cause oscillations) with a good convergence time, while in typical scenarios adapting flows quickly to routes that have good mixing opportunities. Finally, evaluations in an office mesh scenario also confirmed the benefit offered by MMSR and hence the Markovian metric.

VI. CONCLUSION

In this paper we introduced the notion of *Markovian metric*. A Markovian metric models the cost of a path as the sum of the individual conditional link costs, in a way analogous to the decomposition of the joint probability of a Markov chain into a product of conditional probabilities. We introduced the *dot graph* to represent the conditional and unconditional link costs. We showed that minimum cost routing with a Markovian metric can be done by finding a shortest path in the dot graph; hence it has polynomial complexity. We showed how to support a Markovian metric in representative routing protocols.

As a case study, we demonstrated how to use a Markovian metric to make routing decisions that better take advantage of local mixing. We proposed the expected resource consumption (ERC) as a conditional link metric that models the cost reduction due to local mixing. Markovian metric routing using the ERC link metric maximizes the total resource consumption of a path, leading to improved system efficiency. With such a Markovian metric, flows tend to self-organize themselves toward an equilibrium arrangement that can benefit each other. We proposed techniques that can facilitate flows to settle down into an equilibrium.

The local mixing engine, on its own, can improve the link layer efficiency; it identifies mixing opportunities on the fly and takes advantage of them if they are present. Routing with a Markovian metric makes local mixing more useful as it creates more mixing opportunities. This can translate to notable resource saving and throughput gain, as confirmed by simulations.

As a next step, we plan to develop a prototype system on a mesh testbed and conduct more extensive experiments. We plan also to investigate mobile scenarios.

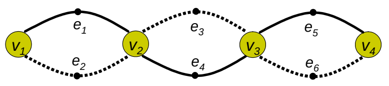

Fig. 12. An example multi-radio mesh network.

Outlook: The potential of Markovian metric extends beyond promoting local mixing. For example, we envision it to be useful for routing in a multi-radio mesh network. Using multiple radios is a promising technique for improving the capacity of mesh networks. However, similar to the case for local mixing, maximizing the benefit of multiple radios calls for intelligent routing decisions that are aware of multiple radios and best take advantage of them.

Consider the example multi-radio mesh network illustrated in Figure 12. Here each node is equipped with two radios, which are tuned to two orthogonal channels. This creates two orthogonal (bidirectional) links between two neighbors; in Figure 12 we denote these two orthogonal links by a solid line and a dashed line, respectively. Suppose we want to find a route from v_1 to v_4. If there is low background traffic, then a route that alternates between the two radios is a good route, because the forwarding and receiving of packets can happen in parallel at a node for the flow, instead of competing with each other. A Markovian metric fits naturally in this scenario. To encourage using different radios in adjacent hops, we can set the the conditional cost of $\mathsf{cost}(e_3|e_1)$ to a lower value than the unconditional cost of transmitting on link e_3.

Furthermore, a Markovian metric can potentially provide a unified solution for a wireless network equipped with multiple radios *and* local mixing.

ACKNOWLEDGMENTS

The authors thank their colleagues J. Padhye, P. A. Chou and V. Padmanabhan for their help and the useful discussions.

REFERENCES

[1] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *Proc. 39th Annual Conf. Inform. Sci. and Systems (CISS)*, Baltimore, MD, Mar. 2005, [Online] http://research.microsoft.com/∼yunnanwu.

[2] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *SIGCOMM*. Pisa, Italy: ACM, Sept. 2006.

[3] R. Draves, J. Pahdye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *SIGCOMM*, Sept. 2004.

[4] ——, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom*. ACM, 2004.

[5] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Oct. 2003, Internet Engineering Task Force (IETF), RFC3626.

[6] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector protocol (DSDV) for mobile computers," *ACM SIGCOMM: Computer Communications Review*, vol. 24, no. 4, Oct. 1994.

[7] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, ch. 5, pp. 153–181.

[8] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "High throughput path metric for multi-hop wireless routing," in *MobiCom*. ACM, 2003.

[9] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," July 2003, IETF, RFC3561.

[10] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," in *Proc. Information Theory and Applications Workshop*. San Diego, CA: Univ. of California, San Diego, Feb. 2006.

[11] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye, "Feasibility study of mesh networks for all-wireless offices," in *MobiSys*. Upsalla, Sweden: ACM/USENIX, June 2006.