

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 x_1 to v_3 via v_2 and v_3 wants to send packet 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)) [1]. 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. Wu et al. [1] 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.

Generalizing [1], Katti et al. [2] recently presented a framework for exploiting network coding and the broadcast medium 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



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].

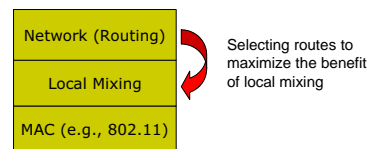


Fig. 2. The big picture. The local mixing engine sits between the network layer and the MAC layer and thus presents an enhanced link layer to the network layer. This paper develops routing solutions that can better take advantage of the local mixing engine.

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 mixing opportunities. 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). 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. 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].

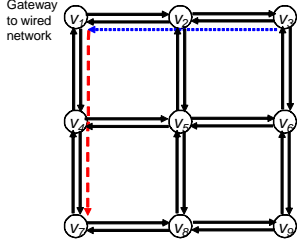


Fig. 3. An example mesh networking scenario. There are 9 mesh access points and v_1 is a gateway to the wired network. The connectivity graph is shown in the figure. Assume currently there are two long-term background flows, $v_3 \rightarrow v_2 \rightarrow v_1$ and $v_1 \rightarrow v_4 \rightarrow v_7$.

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. This motivates the concept of a *Markovian metric*.

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. 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.

Definition 1 (Markovian Metric):

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

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

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

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.

A. The Dot Graph Representation

Properly defining the conditional and unconditional link costs and computing them is a central problem in the applications of Markovian metric; a concrete example can be found in Section III. In this section we assume that 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 $\text{cost}(v_i \rightarrow v_j)$ and $w_{i,j,k}$ to denote a conditional link cost $\text{cost}(v_j \rightarrow v_k | v_i \rightarrow v_j)$. We now discuss a graphical representation of these costs, which we call the *dot graph*. 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 $\text{cost}(v_2 \rightarrow v_3 | v_1 \rightarrow v_2) < \text{cost}(v_2 \rightarrow v_3)$ because a packet from v_2 to v_3 that arrived from v_1 can be mixed with the existing traffic in the flow $v_3 \rightarrow v_2 \rightarrow v_1$.

To produce \dot{G} , we add to G 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 $\text{cost}(v_j \rightarrow v_k | v_i \rightarrow v_j)$ in the given set W_{con} , we draw an edge from the dot $e_{i,j}$ to the dot $e_{j,k}$. 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, $w_{i,j}$. 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.

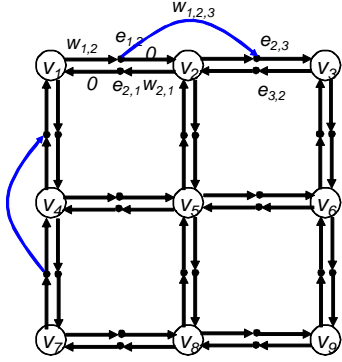


Fig. 4. The dot graph representation of a collection of conditional and unconditional link costs, for the example graph in Figure 3.

In general, we allow the coexistence of a conditional cost and unconditional cost for the same link, e.g., $\text{cost}(b \rightarrow c|a \rightarrow b)$ and $\text{cost}(b \rightarrow c)$. 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} :

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

This corresponds to the physical route $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_9$. 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 :

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

This corresponds to the physical route $v_1 \rightarrow v_4 \rightarrow v_7 \rightarrow v_8 \rightarrow v_9$, 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 $O((|V(G)| + \min\{|E(G)|, 2|W_{\text{con}}|\})^2)$.

C. Adaptive Routing in a Practical Network

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 its own outgoing wires of v_i and the outgoing wires of its incoming dots $e_{*,i}$. For example, in Figure 4, physical node v_2 implements the computation involving wires $e_{*,2} \rightarrow v_2$, $v_2 \rightarrow e_{2,*}$, $e_{1,2} \rightarrow e_{2,3}$. We next examine how to support a Markovian metric in two types of representative routing algorithms.

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.

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 along 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 $\text{cost}(e_{i,2} \rightarrow e_{2,j})$.

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. These 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 $\text{cost}(e_{i,2} \rightsquigarrow v_j)$ the estimated minimum cost to reach destination v_j . Note that in this case,

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

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

III. 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) [7]. 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 a 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 $e_{i,j} \rightarrow e_{j,k}$, which crosses the current node v_j . The packets forwarded by the current node can be classified into categories associated with the wires. A packet that is received along $e_{i,j}$ and sent along $e_{j,k}$ falls into the category “ $e_{i,j} \rightarrow e_{j,k}$ ”. 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 $e_{j,k}$. (We could also use ETT [4] in lieu of ETX.)

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. In the simulations, we use $N = 10$ slots, each of length 0.5s.

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:

$$\text{erc}_n := \frac{\text{Resource consumed by pkts sent in slot } n}{\# \text{ of packets sent in slot } n}. \quad (2)$$

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

$$\text{ERC} := \sum_{n=0}^{N-1} \alpha_n \text{erc}_n; \quad \alpha_n = \alpha^{N-1-n} \left(\frac{1-\alpha}{1-\alpha^N} \right). \quad (3)$$

Here the parameter α is the forgetting factor for old observations. Older observations receive lower weights. In the simulations, we use $\alpha = 0.8$.

The above measurement method generates an estimate of the *current ERC*, which is the ERC seen by a flow whose packets are currently being mixed. In addition to the current ERC, we collect another statistic called the *marginal ERC*, which is the ERC meant for a flow that has not been actively using the wire. If the existing flows already use up most of the mixing opportunities, then the marginal ERC will not have a high discount. Both the current ERC and the marginal ERC are reported.

To compute the marginal ERC, a simple rule is applied in our current implementation. Specifically, for a wire $e_{ij} \rightarrow e_{jk}$ in a given time slot, 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 marginal ERC as 0.75 of the ETX (25% discount); otherwise, we set the marginal ERC as the ETX (no discount).

B. 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.

IV. 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 metric in Qualnet 3.9.5,

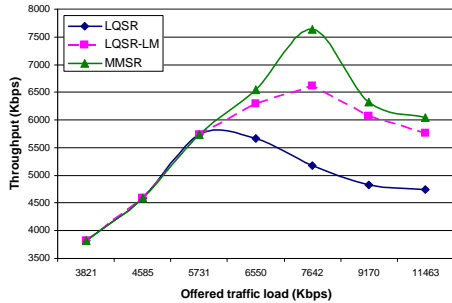


Fig. 5. Throughput comparison of MMSR, LQSR+LM and LQSR.

a widely used event-driven simulator for wireless networks. The basic routing framework can be viewed as a simplified version of LQSR [3]. 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.

We have conducted extensive evaluations to examine the performance of MMSR. In a number of micro-benchmarks, 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. 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. Due to space constraint, the detailed simulation parameters and results are described in a technical report [8]. In the following we present an example performance comparison result.

We use 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 5. 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 6 gives the amount of *resource* saved by using MMSR; the y -axis is the number of original source packets minus the number of actual transmissions. MMSR consistently provides reduction of packet transmissions of over 10,000 packets across a wide variety of traffic demands.

V. 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

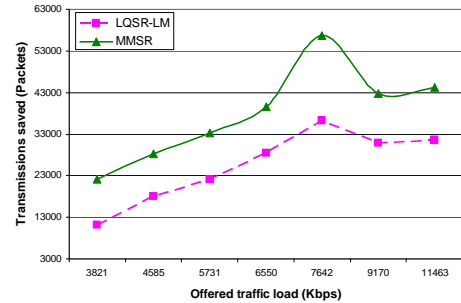


Fig. 6. Transmissions saved through mixing in MMSR and LQSR+LM.

decomposition of the joint probability of a Markov chain into a product of conditional probabilities. 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.

ACKNOWLEDGMENTS

The authors thank their colleagues J. Padhye, P. A. Chou, and V. Padmanabhan for their help and the useful discussions, and D. Florencio for his suggestion about marginal ERC.

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. Padhye, 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. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "High throughput path metric for multi-hop wireless routing," in *MobiCom*. ACM, 2003.
- [8] Y. Wu, S. M. Das, and R. Chandra, "Routing with a markovian metric to promote local mixing," Microsoft Research, Redmond, Tech. Rep. MSR-TR-2006-158, Nov. 2006, [Online] <ftp://ftp.research.microsoft.com/pub/tr/TR-2006-158.pdf>.