Context-based Routing: Techniques, Applications and Experience

Saumitra Das[1], Yunnan Wu[2], Ranveer Chandra[2], Y. Charlie Hu[1]

[1]Purdue University, [2]Microsoft Research Redmond

Abstract

Routing protocols in multi-hop networks are typically based on finding low cost paths by modeling the cost of a path as the *sum of the costs* on the constituting links. However, the insufficiency of this approach becomes more apparent as new lower layer technologies are incorporated. For instance, to maximize the benefit of multiple radios, ideally we should use routes that contain low interference among the constituting links. Similarly, to maximize the benefit of network coding, we should ideally use routes that create more coding opportunities. Both of these are difficult to accomplish within the conventional routing framework because therein the links are examined in isolation of each other, whereas the nature of the problem involves interdependencies among the links.

This paper aims at revealing a unifying framework for routing in the presence of inherent link interdependencies, which we call "context-based routing". Through the case studies of two concrete application scenarios, network coding–aware routing and self-interference aware routing in multi-radio systems, we highlight the common key pillars for context-based routing and their interplay: a context-based path metric and a route selection method. We implement context-based routing protocols in Windows XP and Linux and evaluate them in detail. Experiments conducted at 2 testbeds demonstrate significant throughput gains.

1 Introduction

Routing in wireless mesh networks is a well studied problem. A common practice is to model the cost of a path as the sum of the costs on the constituting links, where the link cost reflects the link quality e.g., the per-hop round-trip time (RTT) [1], the expected transmission count (ETX) [3], and the expected transmission time (ETT) [5]). Routing then aims at finding the path offering the lowest total cost. However, the inadequacy of this widely used approach becomes evident as new lower layer technologies appear.

For example, a promising technique for improving the capacity of wireless mesh networks is to use multiple radios[1]. With multiple radios, more concurrent communications can be packed in spectrum, space and time. To maximize the benefit of multiple radios, ideally we should use routes that contain low interference among the constituting links. Another example is link layer network coding, a

recent advance [7] that exploits the broadcast nature of the wireless medium. Network coding, on its own, can improve the link layer efficiency. The gain of this technique, however, critically depends on the traffic pattern in the network, hence the routing decisions. To maximize the benefit of network coding, ideally we should use routes that create more mixing opportunities. In these scenarios, the conventional routing framework does not perform well because therein the links are examined in isolation of each other. To fully leverage the lower layer advances, we need advanced routing techniques that can properly model the inherent interdependencies of the links and then identify good routes.

Related Work: For multi-radio systems, some progress has been made in selecting interference aware routes. Draves et al. [4] proposed the WCETT metric, which penalizes a path that uses the same channel multiple times, thus modeling link interference to some extent. However, WCETT assumes that all links on a route that use the same channel interfere, which does not incorporate the phenomenon of spatial reuse and may miss high-throughput routes that reuse channels carefully at links sufficiently apart (see Figure 4). Moreover, [4] uses Dijkstra's shortest path algorithm. Although Dijkstra's algorithm is optimal for the conventional path metric, i.e. a sum of link costs, it is not optimal for WCETT. Another proposal for interference aware routing is [15, 16]. They proposed the MIC metric and showed how to find the optimal route under this metric in polynomial time. However, to ensure finding the optimal path in polynomial time, the metric is forced with some constraints (decomposability, fixed memory) that may not match well with the nature of the underlying link interdependencies. This can cause issues in modeling the costs of routes (see Figure 5), and as a result, the path found can be inefficient.

Network coding–aware routing has been studied in [10, 13] from theoretical perspectives. These papers presented theoretical, flow-based formulations for computing the throughput with network coding and network coding–aware routing, assuming centralized control in routing and scheduling. To characterize the gain of network coding, these theoretical studies showed that it is necessary to examine two consecutive hops jointly, because of the link interdependency.

Overview: These previous studies have offered valuable insight in dealing with link interdependencies. In this paper, we aim at revealing a unifying framework for routing in the presence of inherent link interdependen-

---

[1]Several vendors already provide 6 radio routers [2, 9].

cies, which we call "context-based routing". We proceed by studying two concrete application scenarios: network coding–aware routing, and self-interference aware routing. Through these case studies, we highlight the common key pillars for context-based routing and their interplay, and provide a more systematic treatment of these pillars. In particular, we show how to overcome the shortcomings of the aforementioned approaches for self-interference aware routing in multi-radio systems and also apply the techniques more generally to other scenarios.

The first pillar is the concept of *context-based link metrics*, which can model the interactions among different links in a route. A context-based link metric examines the cost of each link based on what links are used prior to the current link. Having such context allows us to evaluate the 'goodness' of routes by considering the impact of the links on each other. More specifically, the conditional link costs allow us to conveniently characterize effects such as: (i) the self-interference caused by other links of the same flow, and (ii) the reduction of channel resource consumption due to the use of network coding.

The second pillar is an optimized route selection module, which can search for good paths under a context-based link metric. The interdependencies among the links in a context-based link metric make it challenging to search for a good path. To make things worse, sometimes the context-based link metric is by nature globally coupled, meaning that the cost of a link can depend on as far as the first link. We propose a general context-based path pruning method (CPP). To model the context of a partial path while avoiding the exponential growth in the number of partial paths, CPP maintains a set of paths that reach each node, corresponding to different *local contexts*. Each local context can be viewed as a concise summary of the dominating history of all the partial paths that end with that local context, based on the observation that older history usually has less impact on the future. The best path under each local context is maintained and considered further for possible expansion into an s–t path.

The essence of the proposed routing solution (CRP or Context Routing Protocol) lies in (i) properly modeling the link interdependencies (via the context-based metric) and (ii) handling the ensuing algorithmic challenges in route optimization (via CPP). The use of local contexts in pruning is synergistic with the use of contexts in cost modeling. Together they present a general context-based routing paradigm, which is applicable in scenarios where modeling link-interdependencies is critical.

We have implemented CRP on Windows XP and Linux. We have also tested our system through Qualnet simulations and on two different testbeds. Our evaluations show that it provides TCP throughput gains up to 130% and on average around 50% over state-of-the-art approaches for multi-radio networks and gains of up to 70% over state-
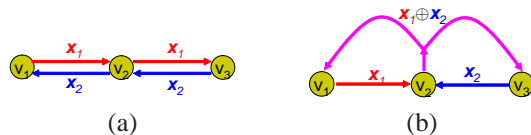

Figure 1: Network coding example.

of-the-art approaches for wireless network coding.

2 Context-Based Routing Metrics

The common defining feature of context-based metrics is that the cost of a link is context-dependent (dependent on what links are already part of the route). To make things concrete, we consider two specific systems: multi-radio multi-channel networks and single-radio networks equipped with network coding and show how context-based metrics can better match the nature of the problems.

2.1 ERC: Context-based metric for exploiting network coding

In *network coding*, 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)) [14]. 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.

Katti et al. [7] recently presented a framework for network coding in wireless networks. in which 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. The network coding engine sits above the MAC layer (e.g., 802.11) and below the network layer. Given the routing decisions, the network coding engine tries to identify mixing opportunities. 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 network coding engine?
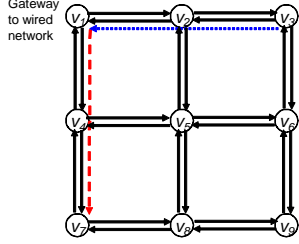
Figure 2: An example mesh network. There are 9 mesh access points and v_1 is a gateway to the wired network.

A natural thought is to modify the link metrics to take into account the effect of the network coding engine in reducing the transmissions over the air. This, however, is not straightforward. Consider the example setting illustrated in Figure 2. 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 network coding 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.

Thus, in the presence of the network coding 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. Specifically, for this application, we model the cost of a path $\mathcal{P} = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_k$ as the sum of the conditional costs on the links:

$$\mathsf{cost}(\mathcal{P}) \stackrel{\Delta}{=} \mathsf{cost}(v_0 \rightarrow v_1) + \mathsf{cost}(v_1 \rightarrow v_2 | v_0 \rightarrow v_1) + \ldots$$
$$+ \mathsf{cost}(v_{k-1} \rightarrow v_k | v_{k-2} \rightarrow v_{k-1}). \quad (1)$$

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 from a. The central issue is to properly define the link costs and compute them. Let us begin by reviewing a conventional (unconditional) link metric. A popular link quality metric in the literature is the expected transmission count (ETX) [3]. This metric estimates the number of transmissions, including retransmissions, needed to send a unicast packet across a link. The ETX metric is a characterization of the amount of resource consumed by a packet transmission.

We now describe how to define a conditional link metric to model the resource saving due to network coding. With network coding, 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 modeled 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 network coding. 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.

Computation of Expected Resource Consumption (ERC) We now explain how to concretely compute the ERC. Each node maintains a `WireInfoTable`. Each row of the table contains the measured statistics about a wire. A wire is a virtual link that connects an incoming and outgoing link from a node (connected the past and future hop), e.g. $e_{i,j} \rightarrow e_{j,k}$, 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 [5] 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. 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: $\mathsf{erc}_n := \frac{\text{Resource consumed by pkts sent in slot } n}{\text{\# of packets sent in slot } n}$. Then we compute the ERC for the wire as the weighted average of the ERCs for the slots:

$$\text{ERC} := \sum_{n=0}^{N-1} \alpha_n \text{erc}_n; \quad \alpha_n = \alpha^{N-1-n} \left(\frac{1-\alpha}{1-\alpha^N} \right). \quad (2)$$

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).

Dealing with oscillation ERC takes the traffic load into account. Could this cause oscillation in the routing decisions? 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. 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.

In summary, using context conveniently represents paths that benefit from network coding. We now show another example of a context-based metric for networks with multiple-radios.

2.2 SIM: Context-based metric for exploiting multiple radios

Consider a multi-hop wireless network equipped with multiple radios. Similar to [4], we assume each radio is tuned to a fixed channel for an extended duration; a route specifies the interfaces to be traversed. We define a context-based SIM (self-interference aware) metric, as a weighted sum of two terms:

$$\text{SIM}(\mathcal{P}) \triangleq (1-\beta) \sum_k \text{ETT}(e_k) + \beta \max_k \text{ESI}(e_k | \mathcal{P}_{k-1}),$$

where e_k is the k-th edge along the path $\mathcal{P}$.

The first term is the sum of the expected transmission time along the route. The ETT [4] metric aims at estimating the average transmission time for sending a unit amount of data. It is defined as: $\text{ETT} \triangleq \frac{\text{PktSize} \cdot \text{ETX}}{\text{Link Bit-Rate}}$. ETX [3] is computed as $\text{ETX} \triangleq 1/(1-p)$, where p is the probability that a single packet transmission over link e is not successful.

ESI of the Bottleneck Link The second term is the estimated service interval (ESI) at the bottleneck link, which reflects how fast the route can send packets in the absence of contending traffic. Note that the max operation is used here instead of the sum operation. This can be explained by a pipeline analogy. In a pipeline system consisting of several processing stages, the long term throughput is determined by that of the bottleneck stage, rather than the average throughput of the stages. Sending packets along a wireless route is similar. The long term throughput of a flow is determined by that of the bottleneck link.

The ESI of a link is defined as:

$$\text{ESI}(e_k | \mathcal{P}_{k-1}) \triangleq \text{ETT}(e_k) + \sum_{j<k} p_{jk} \text{ETT}(e_j). \quad (3)$$

Here p_{jk} is a binary number that reflects whether e_j and e_k interfere. Characterizing the interference relations among the links is itself a research challenge. For instance, one method is to make use of actual interference measurement; see, e.g., Padhye et al. [11] and the references therein. Consider two links, $A \rightarrow B$ and $C \rightarrow D$, using the same channel. As suggested in [11], for 802.11 networks, the primary forms of interference are four cases: (i) C can sense A's transmission via physical or virtual carrier sensing, and hence refrains from accessing the medium, (ii) A can sense C's transmission, (iii) transmissions of A and C collide in D, (iv) transmissions of A and C collide in B. Based on this, we adopt a simplified approach in the experiments of this paper. We treat the two links as interfering if A has a link to C or D with sufficiently good quality, or C has a link to A or B with sufficiently good quality.

The ESI expression (3) leaves out the interference caused by other contending traffic. This is a simplification in modeling. The ESI expression (3) considers the self-interference from the previous hops of the route, by adding up the expected transmission times of the previous links. The intuition is that the packets at the link needs to share the channel with the interfering links on the route. One might ask why we do not add the ETTs from the subsequent links on the path, even though both previous and

subsequent links can create interference. The following theorem provides an answer.

Theorem 1 (Interpretation of Bottleneck ESI)
Assuming ideal scheduling, sufficiently long flow, absence of contending traffic, and an ideal binary interference model dictated by a conflict graph, the end-to-end throughput of $1/\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$ is achievable.

Proof: Under the assumptions in the claim, finding the optimal end-to-end throughput essentially amounts to finding an optimal interference-free scheduling of the uses of the constituting links. If we can schedule each link to transfer B bits in T seconds, then the throughput B/T can be achieved. It is well known that this problem can be viewed as a continuous version of the graph coloring problem on the conflict graph.

In greedy coloring algorithms, nodes in a graph are visited one by one. Each node tries to reuse some existing colors if possible. If not, the node selects a new color. With this procedure, it is easy to see that the graph can be colored in $\Delta(U)+1$ colors, where $\Delta(U)$ is the maximum degree of a vertex. Notice that the greedy coloring algorithm always look at the already colored nodes, but not future nodes. Hence in fact the upper-bound can be tightened to one plus the maximum number of already-colored neighbors for the nodes. We now apply a greedy-coloring like algorithm for scheduling the links on a route. This is illustrated in Figure 3 for a path with 6 links. We visit the links on the route sequentially, from the first hop to the last hop. For each link e_k, find one or more intervals with a total length of $\mathsf{ETT}(e_k)$. Similar to greedy coloring, when assigning the intervals to a link, we only need to examine the previous links, but not future links. With this greedy scheduling process, we can finish the assignment in a total duration of $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$. If we repeat this scheduling pattern for a sufficiently long time, then we can deliver one packet end to end every $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$ (sec). Hence the throughput is achievable. ∎

The above theorem shows that the bottleneck ESI corresponds to a theoretically achievable throughput. Conversely, if a link e_k interferes with a set $\mathcal{F}$ of previous links, then typically links in $\mathcal{F} \cup \{e_k\}$ would be expected to mutually interfere (hence forming a clique in the conflict graph). If that indeed is the case, then we cannot deliver more than one packet end to end every $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$ (sec). This argument shows that the maximum throughput is roughly around $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$.

Note that can be applied essentially with no change to other multi-radio network architectures such as in [8].
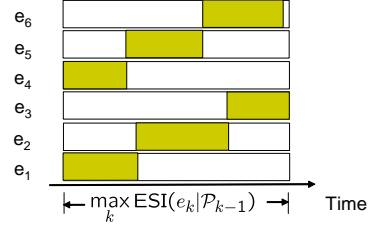

Figure 3: An interference-free scheduling of links. Assume e_k interferes with $e_{k-1}, e_{k-2}, e_{k+1}, e_{k+2}$. A shaded region for a link e_k indicates that e_k is using the medium.
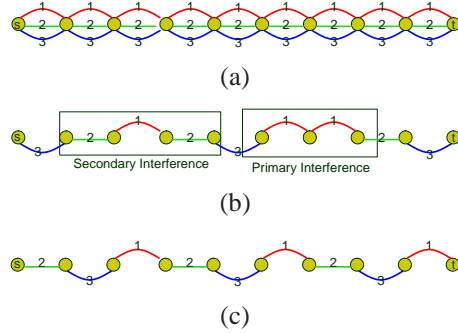

Figure 4: (a) Chain topology. Each node has 3 radios on orthogonal channels 1,2,3. (b) Route selected by using the WCETT metric. (c) Route selected by using the SIM metric.

Related Work: Comparison with Other Metrics The bottleneck ESI models self-interference. The previously introduced metrics, WCETT and MIC, also consider self-interference. We now compare the proposed SIM metric with WCETT and MIC.

The WCETT metric proposed by Draves et al. [4] is defined as:

$$\mathsf{WCETT}(\mathcal{P}) \overset{\Delta}{=} (1-\beta) \cdot \sum_{e_k \in \mathcal{P}} \mathsf{ETT}(e_k) + \beta \cdot \max_{\text{Channel } j} X_j,$$

$$X_j \overset{\Delta}{=} \sum_{e_k \text{ is on channel } j} \mathsf{ETT}(e_k). \qquad (4)$$

Here e_k denotes the k-th hop on the path $\mathcal{P}$ and β is a weighting factor between 0 and 1. The WCETT metric is a weighted sum of two terms. The first term is the sum of the ETT along the path. The second term aims at capturing the effect of self-interference. A path that uses a certain channel multiple times will be penalized.

Although WCETT considers self-interference, it has some drawbacks. Consider the following example. Figure 4(a) shows a chain topology, where each node has three radios tuned to orthogonal channels 1,2,3. We assume two links within 2 hops interfere with each other if they are assigned the same channel. We further assume all links have similar quality. An ideal route in this setting is then a route that alternates among the three channels, such as the one illustrated by Figure 4(c), because it can completely avoid self-interference. Figure 4(b) shows a possible solution returned by WCETT. This route suffers

from primary interference as well as secondary interference. This can be explained by the way WCETT models self-interference. From (4), it is seen that WCETT views a path as a set of links. In this example, it tries to maximize channel diversity by balancing the number of channels used on the path. Thus the path in Figure 4(b) and the path in Figure 4(c) appear equally good under WCETT.

In essence, WCETT takes a pessimistic interference model that all links on the same channel in the route interfere with each other. Indeed, if we use this interference model, then SIM reduces to WCETT. The proposed SIM metric models self-interference can differentiate different ordering of the links in the path and find the optimal path for this example. The benefit is that SIM potentially allows better channel reuse.

Another self-interference aware metric is the metric of interference and channel-switching (MIC), proposed by Yang et al. [15, 16]. The MIC metric is defined as:

$$\mathsf{MIC}(\mathcal{P}) \triangleq \alpha \sum_{e_k \in \mathcal{P}} \mathsf{IRU}(e_k) + \sum_{\text{Node } i \in \mathcal{P}} \mathsf{CSC}_i, \qquad (5)$$

$$\mathsf{IRU}(e_k) \triangleq \mathsf{ETT}(e_k) \times N_k, \qquad (6)$$

$$\mathsf{CSC}_i \triangleq \begin{cases} w_1 & \text{if previous hop is on a different channel,} \\ w_2 & \text{otherwise.} \end{cases}$$

$$0 \le w_1 < w_2. \qquad (7)$$

Here $\alpha > 0$ is a weighting factor to balance the two terms. In (6), N_k is the number of neighbors that interferes with the transmission of link e_k. The IRU (interference-aware resource usage) term aims at reflecting the inter-flow interference. A link that interferes with more neighbors will be penalized. The CSC (channel-switching cost) aims at reflecting the self-interference, since it penalizes consecutive uses of the same channel.

The total CSC term (7) in the MIC metric models the self-interference by considering the immediate previous hop. As an extension, the authors also considered the extension of the MIC metric to model self-interference for more than one hops [16]. However, a potential limitation with the MIC metric and its multi-hop extension is that it has a fixed limited memory span. Consider the example in Figure 5. Suppose the links between nodes a and b have very low costs. Under the MIC metric, it is possible that the route $A \to B \to D \to B \to C$ has a lower cost than the route $A \to B \to C$. Since the path is selected by optimizing the metric, it is unclear whether the MIC metric can rule out the possibility of selecting a pathological path, which has self-interference that are not modeled in the MIC expression.

Compared with the MIC metric, the SIM metric considers the possible interference with all previous hops. This avoids the pathological case as shown in Figure 5. In addition, whereas the link CSCs are summed up in (5), SIM
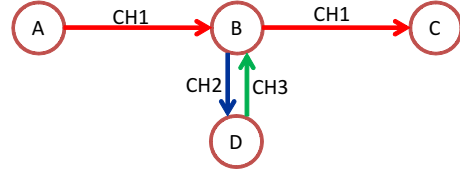

Figure 5: Under the MIC metric, it is possible that the route $A \to B \to D \to B \to C$ has a lower cost than the route $A \to B \to C$. Edges are labeled with the channels.

uses the maximum ESI. The use of the maximum operation can better reflect the fact that the throughput is determined by the bottleneck link.

3 Context-Based Path Pruning

The previous section showed two concrete examples of how context helps in making routing metrics more powerful. In addition to defining a good context-based path metric, another challenge is to find the optimal (or near-optimal) route under the path metric. A context based routing protocol needs both a context-based metric and a way to find good paths under such metrics. In this section, we develop such a path finding method called CPP (Context-based Path Pruning) that can find paths under any generic context-based metric. As a starting point, we first consider a link state routing framework. 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. Hence for link state routing, what is needed is a centralized algorithm for computing the shortest paths. The CPP method will be explained as a centralized algorithm. But we note that it can also be applied in some other distributed settings, such as distance vector protocols and on-demand route discovery; these extensions are omitted in the interest of space.

Let us begin by reviewing the (simpler) problem of finding the optimal path under a decomposable path metric, where each link has a nonnegative cost and the cost of a path is the sum of the costs of the constituting links. This problem is well understood. For example, the classical shortest path algorithm by Dijkstra can be applied to find the optimal path with complexity $O(|V|^2)$, where $|V|$ denotes the number of nodes in the network. Dijkstra's algorithm maintains upper-bound labels $f(v)$ on the lengths of minimum-cost s–v paths for all $v \in V$. The label of each node is either *temporary* or *permanent*, throughput the execution of the algorithm. At each iteration, a temporary label with least total cost is made permanent and the remaining temporary labels are updated. Specifically, if v^* has the minimum total cost among the temporary nodes, then we update the cost of every other temporary node w as:

$$f(w) := \min \left\{ f(w), f(v^*) + c(v^* w) \right\}. \qquad (8)$$

Dijkstra's algorithm operates on an optimality principle: The shortest path from s to t via v is the shortest path from s to v concatenated with the shortest path from v to t. Thus, each node only needs to remember the cost of the best s–v path. Such optimality principle no longer holds for a metric such as SIM, ERC or WCETT; instead, an s–v path $\mathcal{P}_1$ may have a larger cost than an s–v path $\mathcal{P}_2$ but may eventually lead to a lower cost toward the final destination node t.

We propose a context-based path pruning (CPP) technique, as a heuristic method for optimizing a context-based metric. To model the potential impact of past hops on future hops, we maintain a *set* of paths that reach each node v, instead of a single s–v path with minimum cost. A natural question is: How many paths should we store at each node as we search for a good s–t path? Storing all paths would apparently result in an exponential complexity. To keep the complexity manageable, we must constrain the amount of memory at each node. Our key observation is that the effect of self-interference has a localized nature. Normally, given what happened in the near past (i.e., first few previous hops), the older history is unlikely to have a significant impact on the future. This observation motivates us to organize the memory at each node according to several *local contexts*. Each local context can be viewed as a concise summary of the dominating history of all the partial paths that end with that local context, based on the observation that older history usually has less impact on the future. During the execution of the proposed algorithm, each node stores the best path under each possible local context. The best paths stored are considered further for potential expansion into an s–t path.

Degrees of freedom exist in defining the local contexts. For example, we can define the local context of a path as the sequence of links in the last l hops. As another example, we can define the local context of a path as the sequence of channels taken by the links in the last l hops. These would both serve as reasonable summaries of the past that would impact the future.

Algorithm 1 shows a Dijkstra-style instantiation of the CPP method. We maintain a set T of temporary paths and a set P of permanent paths. In each step, we choose the temporary path with minimum cost. Such a path, say $\mathcal{P}^*$, is made permanent. Then we consider the possible ways of extending $\mathcal{P}^*$ toward an s–t path. For each extension $\mathcal{P} = \mathcal{P}^* + e$, we determine its local context and search for a path with the same local context in T and P. If a path with the same local context already exists, then such existing path is compared with $\mathcal{P}$ and the winner is retained. If a path with the same local context does not exist, then $\mathcal{P}$ is added to T.

There is an alternative way to understand the CPP method. For each physical node v, introduce one vertex

---

Algorithm 1 A Context-based Path Pruning Method

INPUT: A function that can evaluate the cost of a path.
$T := \{s\}$; /* The set of temporary paths. */
$P := \emptyset$; /* The set of permanent paths. */
while $T \neq \emptyset$ **do**
  choose the path $\mathcal{P}^*$ from T with the minimum cost;
  $T := T - \mathcal{P}^*$; $P := P + \mathcal{P}^*$;
  for each valid extension of $\mathcal{P}^*$, say $\mathcal{P} = \mathcal{P}^* + e$, **do**
    $c := \texttt{LocalContext}(\mathcal{P})$;
    if $T \cup P$ contains a path $\mathcal{Q}$ with local context c **then**
      replace $\mathcal{Q}$ by $\mathcal{P}$ if it has a lower cost than $\mathcal{P}$;
    else
      $T := T + \mathcal{P}$;
    end if
  end for
end while
OUTPUT: For each node v, find the best local context $c^*(v)$ resulting in minimum cost. For each context c of each node v, store the best link reaching it with minimum cost. To recover a route from v to s, back-track from $c^*(v)$ along the best links toward s.
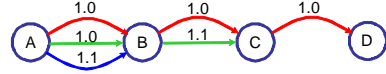

Figure 6: An example graph. There are three orthogonal channels, CH1, CH2, CH3, from top to bottom. The edges are labeled with their ETT.

v_c for each local context c applicable to v and interconnect the vertices according to original connectivity. Denote such a *context-expanded graph* by G_c. We can interpret Algorithm 1 as applying a revised version of the Dijkstra's algorithm to the expanded graph G_c. More specifically, since here the path metric is not decomposable, the cost update step (8) needs to be revised. Instead of using (8), node v^* first reconstructs the current best path from the source, say $s \overset{\mathcal{P}}{\leadsto} v^*$. Then each neighbor node w of v^* is updated using the following path-based update rule:

$$f(w) := \min\left\{ f(w), \mathsf{cost}\left(s \overset{\mathcal{P}}{\leadsto} v^* \to w \right) \right\}, \quad (9)$$

where $\mathsf{cost}(\cdot)$ returns the path metric.

We now illustrate this context-expanded graph using an example shown in Figure 6 for multi-radio networks. Here there are there orthogonal links from A to B, two links from B to C, and one link from B to C. The ETT metrics for the links are shown on the links. Consider $\beta = 0.5$ and the simplified interference model mentioned in Section 2. In this example, Dijkstra's algorithm using the path-based update rule (9) will return $A \overset{\text{CH1}}{\longrightarrow} B \overset{\text{CH2}}{\longrightarrow} C \overset{\text{CH1}}{\longrightarrow} D$, with a total cost of $0.5 * (1.0 + 1.1 + 1.0) + 0.5 * \max\{1.0 + 1.0, 1.1\} = 2.55$.

Figure 7 shows the expanded graph for the case the local contexts are defined by the previous hop's channel ID. Taking the original node B as an example, there are three nodes in G_c, which are associated with three channels to reach B. To connect Algorithm 1 with running Dijkstra's algorithm using the path-based update rule (9) over the expanded graph, we can view each node as storing the
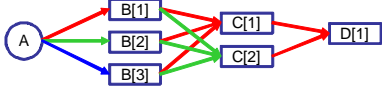
Figure 7: Expanded graph when the local contexts are defined by the previous hop's channel ID. $X[i]$ corresponds to the local context where the incoming link is on channel i.
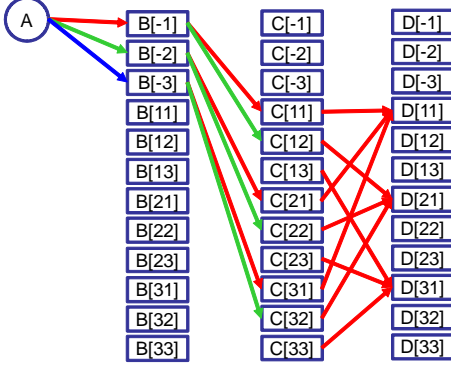

Figure 8: The expanded graph for the case the local contexts are defined by the previous two hops' channel IDs. Here $X[ij]$ corresponds to the context where this node is reached via a link in channel i, followed by a link in channel j. $X[-j]$ refers to the local context where this node is directed reached from the source via a link in channel j.

optimal path reach it from the source; the optimal path can be obtained by backtracking along the best links that reach each node.

In this case, the optimal route found is $A \xrightarrow{\text{CH2}} B \xrightarrow{\text{CH1}} C \xrightarrow{\text{CH1}} D$, with a total cost of $0.5 * (1.0 + 1.0 + 1.0) + 0.5 * \max\{1.0 + 1.0, 1.0\} = 2.5$.

If the local contexts are defined by the previous two hops' channel IDs, then the resulting expanded graph would be the one shown in Figure 8(c). This will yield the optimal route $A \xrightarrow{\text{CH3}} B \xrightarrow{\text{CH2}} C \xrightarrow{\text{CH1}} D$, with a total cost of $0.5 * (1.1 + 1.1 + 1.0) + 0.5 * \max\{1.1, 1.1, 1.0\} = 2.1$.

Handling the ERC metric is simpler for CPP since ERC simple has one-hop of memory. Thus the local context is simply defined as the previous hop on which the packet arrived instead of specifying the channel. This shows that CPP is a generic technique which can be used for many context-based metrics by defining the local contexts in accordance with the definition of the metric being used.

Optimality If the path metric indeed has a fixed memory span (say, l hops) such as in ERC, then CPP with the local context defined by the l-hop links is guaranteed to find the optimal solution (because no pruning step is suboptimal). In our case, the SIM path metric has a memory span that could potentially involve the entire path history. Even if the path metric has a longer memory span than the length of the local contexts, the CPP method can still be applied as an effective heuristic method.

Related Work: Comparison with the Route Selection Method in [15, 16] Yang *et al.* [15, 16] proposed a method for finding the optimal route under the MIC metric (5) and its multi-hop generalization in polynomial time complexity. The method hinges upon the fact that the path metric is decomposable into a sum of link costs, where the cost of a link depends only a fixed number of previous hops; in other words, the path metric is additive and has a fixed memory span. In [15, 16], a virtual graph is constructed, by introducing virtual vertices and edges to represent different states; each edge has an associated cost. Then finding an optimal route under the decomposable, finite-memory metric in the original problem becomes the problem of finding an optimal path in the virtual graph under a memoryless metric, where the cost of a path is simply the sum of the costs of the constituting edges. Therefore, for the additive path metric with a fixed memory span, the optimal solution can be found in polynomial time.

In comparison, the CPP method is applicable for all path metrics, not only when the path metric is decomposable and has finite-memory. When the path metric is decomposable and has finite-memory, CPP also finds the optimal answer. When the path metric is not decomposable, CPP can still be used, as a heuristic method. This comes from the fact that CPP operates by always examining the partial-paths from s and applying the path metric to evaluate their costs, which is apparent from Algorithm 1, as well as the path-based update rule in (9). Observe also that the edges in Figures 7 and 8 do not have associated costs, in contrast to the virtual graph in [15, 16]. Although the local context used in CPP only has limited information about the path history, the path metric has the information of the complete path and can properly take into account any local or global link interdependency. This mix of "local" memory and "global" path evaluation metric is a distinct feature of the CPP method, rendering it generally applicable and efficient.

Complexity As we mentioned earlier, Algorithm 1 can be essentially viewed as applying Dijkstra's algorithm with the path-based update rule (9) over the expanded graph. Note though that in Algorithm 1, the involved vertices and links are constructed on the fly, without explicitly maintaining the expanded graph. Due to the connection between Algorithm 1 and Dijkstra's algorithm, we can easily conclude that the time complexity of the Algorithm 1 is $O(C^2)$, where C is the total number of local contexts at all nodes. If we define the local context of a path as the sequence of channels taken by the links in the last l hops, then C is upper-bounded by $(K + K^2 + \ldots + K^l) * |V(G)|$, where K is the number of channels in the system and $V(G)$ is the set of nodes in the original network. For practical purposes, we specifically

propose to use $l = 2$; see Figure 6 for an example definition of the local contexts. This would lead to a specific complexity of: $O\left(\left((K + K^2) \cdot |V|\right)^2\right)$.

4 Performance

This section documents our experience with the Context Routing Protocol using simulations and a real deployment. CRP was implemented both for WindowsXP as well as for Linux.

4.1 Evaluation Methodology

Simulation Setup : We implemented CRP in QualNet 3.9.5, a widely used simulator for wireless networks. CRP was implemented with support for multiple interfaces, ETX and packet pair probing for ETT calculations, periodic dissemination of link metrics, the WCETT, SIM and ERC metrics and the CPP route selection method. The simulations use the 802.11a MAC and a two-ray fading signal propagation model. All radios operate at a nominal physical layer rate of 54 Mbps and support autorate. We use both UDP and TCP flows in the evaluation. The simulations use 2-hop context length for SIM and 1-hop context length for ERC. We also implemented a complete version of COPE network coding [7] in QualNet as a shim layer based on the protocol description.

Testbed Setup We implemented CRP in Windows XP and Linux and tested it on two separate wireless testbeds. CRP for Windows was implemented by extending the MR-LQSR protocol [5] implementation as a loadable Windows driver that sits at layer 2.5. CRP appears like a single virtual network adapter to applications by hiding the multiple physical interfaces bound to it. This allows unmodified applications to run over CRP. Routing operates using 48 bit virtual Ethernet addresses of the MCL adapter in each node. This choice for our CRP implementation allows a direct and fair comparison with the WCETT metric since it is based on the same underlying codebase. The CRP code uses 2-hop channel ids as context. CRP for Linux was implemented by extending the SrcRR routing protocol from the RoofNet project. We also used the publicly available COPE implementation from the authors of the protocol to implement network coding. This implementation was available only for Linux.

The CRP protocol was deployed on two testbeds: (1) A 14 node wireless testbed running Windows XP (testbed A), and (2) A 20 node wireless testbed running Mandrake Linux (testbed B). Network A is located on a floor of an office building in Redmond and is illustrated in Figure 9. 10 of the nodes are small form factor HP desktops each with 3 DLink AWG132 802.11a/b/g USB cards while 4 nodes are Toshiba tablet PCs with one Netgear WAG511 802.11a/b/g PCMCIA card and 2 DLink
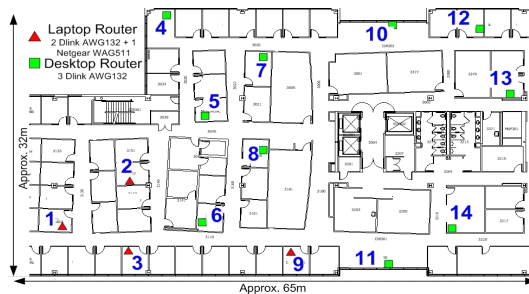

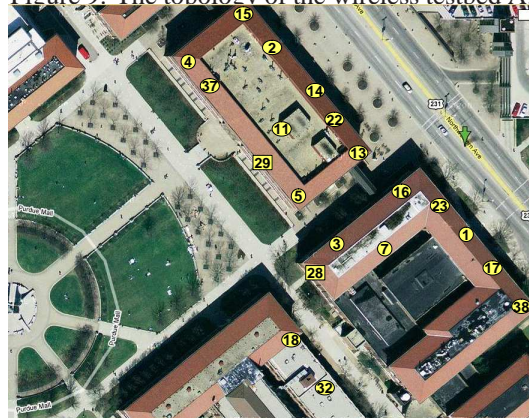Figure 9: The topology of the wireless testbed A.

Figure 10: The topology of the wireless testbed B.

AWG132 802.11a/b/g cards. The driver configurations are modified to allow multiple cards from the same vendor to co-exist in a machine. The radios on each node have their own SSID and are statically assigned the 802.11a frequencies 5180, 5240 and 5300 GHz. Thus, the 42 radios form three 14-node networks each with its own frequency and SSID glued together through the CRP virtual adapter. Each radio works in ad-hoc mode and performs autorate selection. The OS on each machine implements TCP-SACK. Finally, the nodes are static and use statically assigned private IPv4 addresses. The results are expected to not be significantly affected by external interference since no other 802.11a network was in the area. Network B is located across 3 academic building at Purdue University and is shown in Figure 10. The machines are small form factor HP desktops each with an Atheros 802.11a/b/g radio and run Linux.

4.2 Simulation Results

We first demonstrate the performance of CRP in a controlled and configurable simulator setting.

4.2.1 CRP in a Frequency Reuse Scenario

The first scenario demonstrates how CRP can exploit frequency reuse opportunities in a correct way. We consider a chain of 10 nodes separated by 300m, each with 3 radios on 3 orthogonal channels. The transmission power used causes interference to nodes even two hops away which
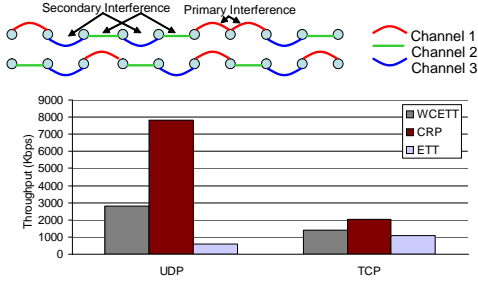
Figure 11: Performance under frequency reuse.

Figure 12: Performance under heterogeneous nodes.

Figure 13: Performance in a large network.

is typical in real networks. Figure 11 shows the routes selected by using WCETT (on top) and CRP (below).

Notice the route selected by WCETT: while it maximizes channel diversity, i.e. each channel is used exactly equally (3 times), it cannot optimize the ordering of the channel use (as explained in Section 2.2) while CRP chooses the optimal route. How does this route selection impact performance? Figure 11 also shows the UDP and TCP throughput achieved using ETT, WCETT and CRP on this chain topology. The results show that CRP route selection has a significant impact on performance improving UDP throughput by 166% over WCETT and over 1000% compared to ETT. For TCP, CRP improves the throughput by around 44% and 91% with respect to WCETT and ETT respectively. Thus, CRP is able to select a path with high raw capacity. However, the gains are reduced due to TCP's known inability to reach the available bandwidth effectively in multi-hop wireless networks (which is also amplified by the long chain). An interesting aspect to note here is that CRP can choose this route even by running normal Dijkstra but using the SIM metric, i.e. the benefit comes from the context in the metric. However, the next section shows when SIM by itself is not enough and demonstrates the usefulness of context-based path selection.

4.2.2 CRP in a Heterogeneous Node Scenario

In this scenario, we consider a simple 3 node topology as shown in Figure 12. Node A and B have 2 radios tuned to channel 1 and 2 while node C has one radio tuned to channel 1. Even in such a simple topology, WCETT and ETT are unable to choose the good route which can simply be decided by observation. This is because of the 'forgetfulness' of Dijkstra. Once the route to B is decided, there is no context with which to select the next hop accurately. Thus, both ETT and WCETT in MR-LQSR choose the path with both hops on channel 1 while CRP chooses the path with channels 1 and 2. Figure 12 shows that the CRP route improves the TCP throughput by 108% and UDP throughput by 102% over both WCETT and ETT. Note that the performance improvement is similar for TCP and UDP since in a two-hop scenario, TCP is better able to
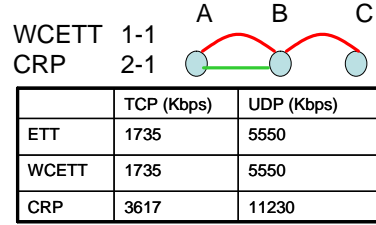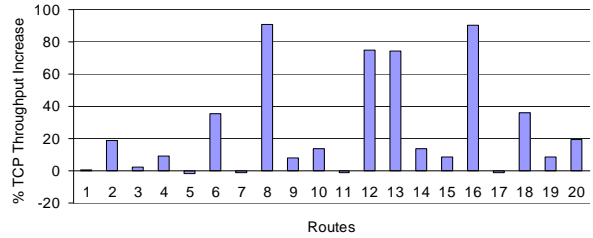
utilize the available bandwidth.

4.2.3 CRP in a Large Network

We now evaluate CRP in a large network of 50 static nodes randomly placed in a 2000 m x 2000 m area. We selected 20 random non-overlapping source destination pairs in the network and initiated a TCP connection between each pair for 60 seconds (so as to observe steady state behavior). Each node had 3 radios on orthogonal channels.

The results in Figure 13 shows the percentage increase in TCP throughput of CRP over WCETT. We find that 7 paths have 20% or more gain in TCP throughput with the gains being as high as 90% in some cases. Typically whenever the performance of CRP and WCETT are equal, it is for two reasons: (1) The path is 3 hops or shorter in length. Since WCETT assigns channels in equal proportions, it always chooses a route similar to CRP (each link has a different channel), (2) Although rare, sometimes when the path is longer than 3 hops, by random chance the WCETT ordering of channels turns out to be optimal.

4.2.4 CRP in Network Coding Scenarios

We now evaluate CRP for a single-radio network with network coding in QualNet. Consider the network shown in Figure 2 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. CRP causes the flow $v_1 \rightsquigarrow v_9$ to choose the mutually beneficial 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

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

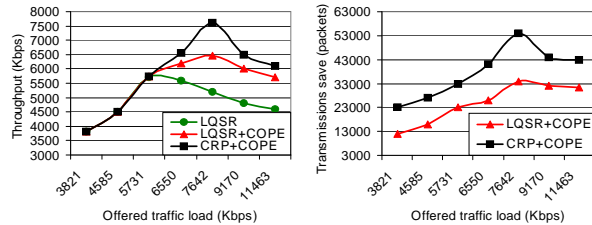Table 1: Gain from CRP+COPE compared to LQSR+COPE.

Figure 14: Throughput comparison and transmissions saved with CRP+COPE, LQSR+COPE and LQSR.

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 1, CRP increases the number of mixed packets in this scenario by $12\times$ in comparison to LQSR+COPE.

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+COPE and CRP+COPE for this scenario for different input loads. The results are depicted in Figure 14. It is observed that LQSR cannot sustain the throughput imposed by the input flows to the network as the load increases. CRP provides throughput gains compared to LQSR (up to 47%) and LQSR+COPE (up to 15%). This is because not only does CRP allow subsequent flows to mix with existing flows, it explicitly tries to maximize mixing. In contrast, without CRP, flows mix essentially by chance. In the example, the flow 1-2-3-6-9 is mixed with 9-6-3-2-1 with CRP, due to the mutually beneficial discounts enjoyed by both flows.

Figure 14 also gives the amount of *resource* saved by using CRP. CRP consistently provides reduction of packet transmissions of over 10,000 packets across a wide variety of traffic demands. Another observation from Figure 14 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.

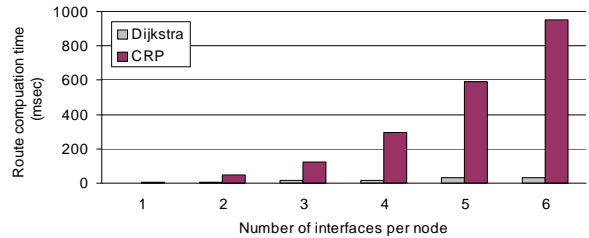Thus, the simulation results show that gains from CRP

Figure 15: Microbenchmark: route computation time.

are observed in two cases for multi-radio networks: (1) When frequency reuse is possible, and (2) When there are heterogeneous nodes in the network, i.e. not all nodes have the same number of interfaces. While these scenarios are common and important to address (typical wireless networks are bound to have some frequency reuse opportunities and networks may have all different types of nodes with different number of radios and characteristics), our testbed evaluation in the next section provides insight into other scenarios where CRP is helpful. In systems with network coding, CRP performance exemplifies that network coding itself cannot provide the best achievable performance and a protocol such as CRP can help network coding achieve better performance.

4.3 Testbed Evaluation

We now document our experience with the performance of CRP in a real wireless network deployment. We first perform a series of microbenchmarks to see its performance in known settings and then evaluate its performance as a running system.

4.3.1 Computational Complexity

CRP improves Dijkstra route selection at the expense of more computation and complexity. While the complexity is not a hurdle (the protocol is already implemented), it is important to find the computational complexity to ascertain the protocols applicability to real mesh networks which may have embedded devices with slow processors.

We configured our CRP implementation's graph cache with a graph topology of 100 nodes (taken from a typical neighborhood layout) and varied the number of interfaces per node. We then evaluated the time taken by CRP versus Dijkstra in computing the shortest paths to all nodes in the network. We find that our implementation, while significantly more computationally expensive than Dijkstra, can find routes in the extreme case of 100 node networks with 6 radios each in 900 ms. The current MCL implementation infact calls Dijkstra only once per second, caching routes in between. However in our current testbed with 14 nodes each with 42 radios, the computation is performed in around 10-20 ms. Thus, even if we use embedded mesh routers whose CPU speeds are 5-6
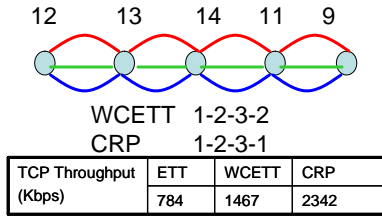
Figure 16: Microbenchmark: frequency reuse.

	12	13	14	11	9
WCETT		1-2-3-2			
CRP		1-2-3-1			

TCP Throughput (Kbps)	ETT	WCETT	CRP
	784	1467	2342

		8	5	7
WCETT	1-1			
CRP	2-1			

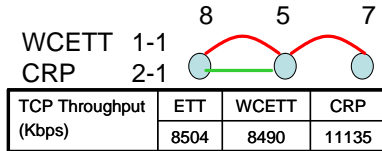| TCP Throughput (Kbps)	ETT	WCETT	CRP
	8504	8490	11135

Figure 17: Microbenchmark: heterogeneous nodes.

times slower than our testbed nodes, we expect to typically find routes under 100 ms. Additionally, one could proactively find such routes in the background to hide this delay as well.

4.3.2 Frequency Reuse Scenario

We now evaluate the real performance gains from good choice of routes in our testbed. Note that unlike the simulator which does not simulate co-channel interference[2], this gives us a better idea of the potential performance benefits.

We selected one of the 4 hops routes in our testbed between nodes 12 and 9 and performed multiple 1-minute TCP transfers taking the median performance. The route and channels chosen by WCETT and CRP are shown in Figure 16. WCETT reuses channel 2 on the last link 11→9 essentially because the channel 2 radio on node 11 had a lower ETT. Thus, WCETT took a local view not realizing this link would cause secondary interference with link 13→14. However, CRP chose the interface on channel 1 on the last link which although locally had a higher ETT, did not cause self-interference. Thus this route got a higher SIM score and was selected by CRP providing a performance gain of 60% and close to 200% with respect to WCETT and ETT respectively. This example clearly shows how using context benefit translates into real world gain.

4.3.3 Heterogeneous Node Scenario

We next consider the simple topology of 3 to see how much gain a CRP provides over a WCETT route in the real world. As shown in Figure 17, we configured node 8 and 5 with two interfaces on channel 1 and 2 while node 7 had one interface on channel 1.

WCETT chose channel 1 on both links due to the the use of Dijkstra and ETT also chose this route because the ETT of channel 2 on node 8 was around 1ms higher than channel 1. However, CRP is able to identify the correct route and chooses a route with channel 2 and channel 1. This allows CRP to have a TCP throughput 31% higher than both WCETT and ETT. Note that the gain is lower than that observed in simulation because some co-channel interference can limit gains. From measurements, we know such interference exists despite our attempts to place the cards at some distance from each other on each machine.

4.3.4 Large Homogeneous Node Scenario

We now evaluate the gain from CRP in a running network topology. Each testbed node in this scenario has 3 radios. We performed multiple 1-minute TCP transfers between node 1 and all the other testbed nodes with WCETT and CRP. The % throughput gain of CRP over WCETT observed from node 1 to the other nodes (numbered on the x-axis) is shown in Figures 18.

The results show that CRP can improve WCETT throughput in 7 out of 13 paths by upto 100% and more than 30% on most paths. Some of the results are expected while some are non-intuitive. Among the expected results are: (1) The paths to nodes 3, 6, 4 and 11[3] have no gain since they are less than 4 hops in length and WCETT can assign costs correctly if the number of hops are less than the number of radios per node. (2) Nodes 5 and 7 are reached via 4-hop routes and the SIM metric can now choose a better path to provide gain.

Among the unexpected results are: (1) Paths to nodes 8 and 9 have gain despite their being 3 hops away. We found this was caused by ETT variation among the interfaces of node 6 which rendered 2 interfaces almost useless due to high ETTs and effectively caused a bottleneck link which required careful route selection. Thus, random variations in link quality can create channel bottlenecks requiring the careful route selection of CRP even on short-hop paths. (2) Routes to node 10 have low gain despite reuse opportunities since the link performance is constrained by the weak link 8→10. However, going to nodes 12 and 13 through 10 gives gain since the throughput is now constrained more by ordering of channels selected than the weak link. (3) Finally, although 14 is 4 hops away, there is no gain because WCETT chooses the good route selected by CRP purely by chance.

---

[2]QualNet physical layer code is binary only so we could not change it to simulate co-channel interference

[3]Node 4 can be reached with 2 hops due to it being in an open area and the waveguide effect of hallways.
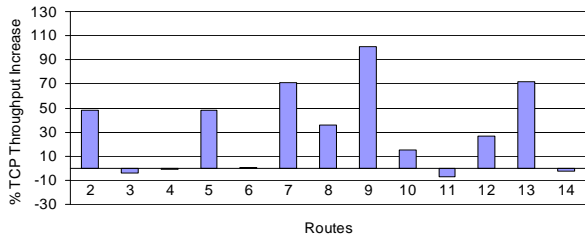
Figure 18: Testbed macrobenchmark: Basic scenario. TCP throughput gain on paths from node 1 to all other nodes.

Figure 19: Macrobenchmark: heterogeneous node scenario. TCP throughput gain on paths from node 1 to all other nodes.

4.3.5 Large Heterogeneous Node Scenario

We also evaluated the gain in a heterogeneous network scenario where all nodes do not have the same number of interfaces. We configure nodes 6, 10 and 11 to have one interface disabled. The percentage throughput gain observed from node 1 to the other nodes (numbered on the x-axis) is shown in Figure 19.

The results again show that CRP can provide gain in the presence of heterogeneous nodes. However, the gain from CRP in this scenario can both increase or decrease due to separate reasons. It can increase compared to the homogeneous scenario due to the increased importance of careful path selection. However, it can also reduce compared to the homogeneous scenario since we reduced the network capacity by reducing the number of radios and so some paths may not support high throughput or may not have the degree of freedom to provide an interference-free channel ordering any longer. We can see examples of both these effects. As an example of the first case, the routes to nodes 5 and 8 went through node 6 with an interface constraint and the routes to nodes 10, 11 and 14 are affected by interface constraints at nodes 10 and 11 resulting in an increased gain compared to the homogeneous scenario. However, since the route to nodes 7 and 9 did not use node 6 this time, they both reduced to a 3 hop route and there was no gain as in the homogeneous scenario. As an example of the second case, there is small or reduced gain for routes to nodes 12 and 13 since these routes contain two nodes with interface constraints making it impossible to find a non-interfering channel ordering.

Finally, some nodes closeby are not affected by nodes with fewer radios and there is no difference in performance (e.g. nodes 2, 3, 4). During this experiment node

Scenarios	UDP Gain	TCP Gain
22→3, 3→22	1.66×	1.29×
18→5, 5→18	1.53×	1.24×
11→7, 7→11	1.78×	1.31×
37→13, 2→5	1.71×	1.24×

Table 2: Median throughput gain from CRP+COPE compared to LQSR+COPE for both UDP and TCP. For each scenario we initiated the flows 15 times.

2 did not have ETT fluctuations so no gain was observed compared to the homogeneous scenario. In addition, the route to node 6, despite an interface bottleneck, was chosen correctly by WCETT purely by chance.

We performed experiments from other nodes and sources and observed a similar overall trend: around half of the paths had gain and the maximum gain observed was 152%. We chose to explain the result in more detail from a single vantage point due to lack of space.

4.3.6 Network Coding Scenario

We now perform experiments with our Linux CRP implementation on top of COPE in testbed B. We take multiple examples of the Alice-and-Bob topologies from our testbed and demonstrate how much gain is provided by simple using SrcRR+COPE versus using CRP+COPE. This testbed has only one radio on each node we simply use the ERC metric in CRP. Table 2 shows the gains achieved using CRP for different topologies tested in the testbed. The left hand column lists the two flows initiated in the network in each case and the node numbers refer to Figure 10.

The table shows significant gains from CRP+COPE over SrcRR+COPE in the range of 1.7× for UDP and 1.27× for TCP. In the first scenario, SrcRR chooses the routes 22-13-3 and 3-5-22 which does not provide gain from network coding while causing interference among the two flows. CRP entices flow 3-22 onto the same route due to the ERC discounts and allows network coding to occur at node 13 for the duration of the transfer. In the second scenario under SrcRR, flow 18-5 uses the route 18-28-5 while flow 5-18 uses the low quality direct route 5-18. CRP advertises discounts on the 28-18 link for traffic from 5 and this causes the two flows to mix at node 28 resulting in gain. In the third scenario, SrcRR choose the routes 11-5-3-7 and 7-16-13-11/7-16-5-11 for the two flows resulting in no coding of packets while CRP brings the two flows onto a mutually beneficial route traversing nodes 5 and 3 in both directions. Finally in the fourth scenario, CRP allows nodes 37 and 2 to choose 11 as a next hop once ERC discounts become visible. Node 5 and 13 can overhear the packets from 37 and 2 respectively giving rise to network coding gain. Without CRP the two flows tend to take the routes 37-11-5 and 2-14-22

13

resulting in no coding gain. Thus, we can see the benefits that CRP provides to the COPE system in allows the network coding engine more opportunities to code packets. Note that if a lot of flows are present in the network (such as the scenarios evaluated in [7]) the benefits from CRP would be less significant since a large number of packets present in the network can provide enough coding opportunities without intelligent route selection. However, CRP can provide COPE with coding opportunities even in more sparse or lightly loaded networks.

Note that we currently do not aim at solving the load balanced routing problem in handling multiple flows. We believe a better practical strategy is for link traffic to affect the link metric and then use a method, such as CRP, to find a route. Thus CRP should potentially be useful for load-balanced routing proposals.

5   Other Uses for Context-Based Routing

Multi-radio coding networks : It is possible to combine multiple context-based metrics to model multiple phenomena. As an example, we are exploring route selection in a wireless network with multiple radios as well as network coding. However there are other orthogonal challenges since multiple radios remove broadcast opportunities through channel assignment while network coding requires broadcast. In addition, the coding algorithms in COPE may need modifications. Thus these networks will require CRP as well as new channel assignment and coding algorithms.

Multicast : While this paper applied context-based routing to multi-radio unicast, another interesting problem we are exploring is whether context can be used to optimize multicast trees in multi-radio networks.

Optical Networks : The problem of choosing lightpaths in an WDM optical network has some synergy with a wireless network equipped with multiple radios. Some costlier devices can also perform wavelength conversion. Thus, the cost of choosing an outgoing wavelength can depend on context, i.e. the incoming wavelength and the cost of conversion. It is interesting to see whether context-based path selection can potentially be applied in such networks.

6   Conclusion

In this paper, we investigated *context-based routing* through the case studies of network coding–aware routing and self-interference aware routing in multi-radio systems. We proposed two specific context-based path metrics for the two application scenarios and provided theoretical justifications. We propose a context-based path selection method, CPP, which serves as an efficient route selection method under context-based metrics. The ERC

metric and the CPP method jointly offer an effective solution for network coding–aware routing. The SIM metric and the CPP method jointly offer an effective solution for self-interference aware routing, which can find good routes when frequency reuse is possible, heterogeneous nodes exist in the network or transient bottlenecks occur and hence is an efficient general purpose routing protocol for multi-radio meshes. The effectiveness of our approach is demonstrated through both simulations and a deployed implementation. These two case studies can in fact be viewed in a common framework for context-based routing, which could be a generally applicable technique whenever link interdependencies exist and potentially has further applications.

References

[1] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proc. of BroadNets*, 2004.

[2] Bel Air Networks. http://www.belairnetworks.com.

[3] D. S. J. D. Couto, D. Aguayo, J. C. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.

[4] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proc. of ACM SIGCOMM*, 2004.

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

[6] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic, 1996.

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. Xors in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, August 2006.

[8] P. Kyasanur and N. Vaidya. Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. In *MC2R 10(1):31–43, Jan*, 2006.

[9] Meru Networks. http://www.merunetworks.com.

[10] B. Ni, N. Santhapuri, Z. Zhong, and S. Nelakuditi, "Routing with opportunistically coded exchanges in wireless mesh networks," In *Proc. WiMesh'06*, Sept. 2006.

[11] J. Padhye, S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of Link Interference in Static Multi-hop Wireless Networks. In *Proceedings of IMC*, 2005.

[12] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proc. of ACM SIGCOMM*, August 1994.

[13] S. Sengupta, S. Rayanchu, and S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *Proc. IEEE INFOCOM*, May 2007.

[14] Y. Wu, P. Chou, and S.-Y. Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. In *Proc. of CISS*, 2005.

[15] Y. Yang, J. Wang, and R. Kravets. Designing routing metrics for mesh networks. In *Proc. of WiMesh*, 2005.

[16] Y. Yang, J. Wang, and R. Kravets. Interference-aware load balancing for multihop wireless networks. In *Technical Report UIUCDCS-R-2005-2526, Dept. of Comp. Science, UIUC*, 2005.