

# Interference-aware Fast Path Adaptation in Mesh Networks

Vishnu Navda<sup>†</sup>, Samrat Ganguly<sup>\*</sup>, Anand Kashyap<sup>†</sup>, Samir Das<sup>†</sup>

<sup>†</sup> Department of Computer Science, SUNY at Stony Brook, NY, USA, {vnavda,anand,samir}@cs.sunysb.edu

<sup>\*</sup> NEC Laboratories America, Princeton, NJ, USA, samrat@nec-labs.com

## Abstract

Existing multihop wireless routing protocols are incapable of reacting quickly to transient degradation of link quality rendering them unsuitable for supporting real-time applications. This limitation is imposed on the metric-based routing protocols in order to achieve route stability and low protocol overhead. We design Deflect, a lightweight and opportunistic mechanism that works underneath the routing layer. Deflect enables fast local adaptation of end-to-end routes in response to sudden drops in link delivery ratio or node failures. Deflect nodes snoop on transmissions to monitor link quality, and deflect traffic on other nodes to circumvent a poor-quality link. The design of Deflect achieves the following goals: a) zero message overhead mechanism for monitoring link quality at a very short time scale; b) fast switching of traffic through a node in neighborhood of the poor link; c) transparency to end-to-end routing protocol. We show that Deflect can react to link quality degradation within a few hundred milliseconds, compared to tens of seconds for an end-to-end routing protocol. Experiments in presence of controlled interference with Deflect show that the average measurement error is less than 5%. Overall, Deflect is shown to provide significant improvement in packet loss and delay compared with traditional metric-based routing.

## 1 Introduction

Our work is motivated by the necessity for wireless mesh networks to support real-time applications such as streaming/interactive voice and video. Examples include providing VoIP coverage in enterprises, and supporting streaming video from surveillance cameras to monitoring locations in communities. To support these applications, a mesh network must provide a steady level of quality of service guarantees. However, the wireless link carrying the streaming data can undergo sudden quality degradation due to interference from other wireless transmitters and external noise sources. This may cause frequent bursts of frame losses [18] resulting in a severe impact on the perceived quality of real-time applications.

For practical reasons, existing routing protocols for

wireless ad hoc or mesh networks, such as DSDV [23], OLSR [3], AODV [22], DSR [10], LQSR [6], SrcRR [5], respond conservatively to sudden degradations in link quality. The limitation in the sensitivity of the routing protocols to link quality is difficult to overcome due to the following reasons. All link quality metrics used in routing, such as ETX [5, 6] require active probing,<sup>1</sup> but frequent probing results in high overhead and can interfere with the actual traffic. Further, a time window of 5 to 10 seconds [6] is required to gather enough probe samples to allow reasonable confidence in the statistics and avoid load sensitivity. Finally, frequent updates of routing metric across the network also results in high control overhead. Additional latency is also incurred in propagating routing updates. The resulting time scale of recovery from poor links can cause significant interruptions in streaming applications – a 10 second loss of a link can result in a loss of 250 frames from a typical video stream.

Our goal in this work is to design a mechanism that enables fast adaptation of the end-to-end path by temporarily circumventing the poor links. To be effective, such an adaptation mechanism must act at a short time scale so that it is able to a) *quickly detect the degradation of the link quality*, b) *switch to a path circumventing the poor link*, and c) *react quickly with negligible control overhead*.

We present a lightweight mechanism, called **Deflect**, for fast path adaptation. Deflect is decoupled from the end-to-end route construction and can complement many existing routing protocols. Deflect’s goal is to protect the end-to-end route from short-term link quality degradation or outage before the end-to-end route recovery takes over. Deflect achieves this goal by (a) a passive, *zero-overhead*, measurement driven approach for monitoring link qualities and (b) switching to alternate routes only in the local neighborhood by very low-overhead signaling. Both these strategies enable rapid path adaptation in the order of hundreds of milliseconds in commodity 802.11-based deployments. Deflect preserves the end-to-end number of hops, preventing sudden changes in end-to-end delays.

---

<sup>1</sup>ETX (expected transmission count) measures the expected number of transmissions needed to send a unicast packet over an 802.11 link. Since in 802.11, unicast packets use a link-layer acknowledgment, ETX is measured as the inverse of the product of delivery ratios of probe packets in both directions for the link.

Earlier protocols such as Divert [18] and ExOR [2] have been proposed that exploit link or path diversity similar to Deflect, but Divert is limited to single-hop WLANs, while ExOR is applicable only for bulk-transfer data and not real-time traffic.

In the rest of the paper we design and evaluate Deflect and demonstrate its performance in an indoor 802.11-based mesh network. First, we describe the Deflect protocol, which is based on the novel concept of *cooperative relay* nodes. A cooperative relay node silently monitor links in its vicinity and, if and when appropriate, volunteer to bypass them in a fashion transparent to the routing protocol. We argue that such relays will be plentiful in a dense mesh network. We then describe the detailed design of Deflect particularly with regard to how link qualities are monitored and estimated, and how paths are switched. Finally, we present evaluations to analyze the advantages of using Deflect.

## 2 Motivation and Overview

In this section, we first present a set of experiments demonstrating the inadequacy of routing protocols using a probing-based metric like ETX to recover from short-term link quality fluctuations. Then, we present an overview of the design of Deflect and then do a study to determine the frequency with which Deflect may be applicable.

### 2.1 Limitations of routing protocols

To evaluate the responsiveness of routing protocols and routing metrics to adapt to bursty background traffic, we set up a series of experiments in our 20 node indoor mesh testbed that uses 802.11b radios. The actual details of the testbed architecture is not relevant here. It is described later in Section 4. These measurements are done on a section of testbed with 5 nodes. Two nodes, say  $S$  and  $D$ , act as traffic source and destination respectively, with three 2-hop paths available between them, via three other nodes, say  $A, B, C$ . We run a link-state protocol OLSR [3] for routing and use ETX [5] as the path metric. By default, ETX uses a 10 sec time window with probes at the rate of 1 probe/sec.

We also set up background traffic to deliberately cause link quality fluctuations. This traffic is generated by node  $A$  in the form of 10 sec long bursts of back-to-back broadcast frames alternating with 10 sec silence periods. This causes channel contention at nodes  $B$  and  $C$ . By default, the ETX metric used for routing is calculated by using the statistics of the last 10 ETX probe packets, where the probes are sent at the rate of 1 probe/sec. This measurement window controls the responsiveness of ETX with respect to changing conditions. Similar parameter setting has been used in literature in the past [6].

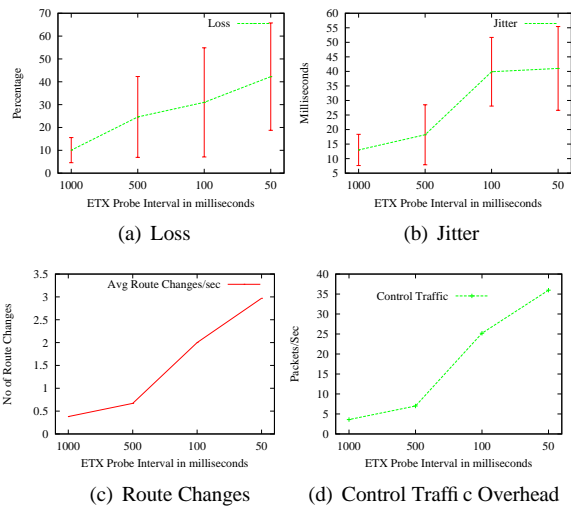


Figure 1: Effect of ETX probe interval on loss, jitter and route changes and route control traffic overhead.

Figure 1 depicts various performance metrics obtained during a 100 sec run of 1000 Kbps CBR traffic from  $S$  to  $D$ , averaged over several such runs. With the default probing frequency, the CBR flow incurs about 10% packet losses due to the interference traffic. Such losses are due to the lack of responsiveness in the metric calculation as mentioned before – route changes lag interference changes by several seconds. However, increasing probing frequency (500ms, 100ms and 50ms) does not improve the performance. Packet losses actually increase with probe frequency as probes themselves interfere with data traffic (Figure 1(a)). For the same reason, the delay jitter also increases (Figure 1(b)). As far as routing is concerned, route change frequency increases (Figure 1(c)) with probe frequency, as now routing becomes more reactive, increasing control overhead as well (Figure 1(d)). Higher control overhead is another reason for increasing loss and jitter.

Above experiments show that existing routing protocols cannot adapt quickly to handle transient losses. There is a need for fine grained path adaptation to support real-time media over mesh networks. We conclude that any active probing method is not suitable to handle short term variations due to self-interference, control overhead and route instability.

### 2.2 Basic Idea of Deflect

The basic idea in Deflect is that a node can volunteer to replace a neighboring relay node on a routing path if it determines – via passive measurements based on snooping – that the overall delivery rate will improve. Consider Figure 2 for an illustration. Assume that  $A - B - C - D - E$

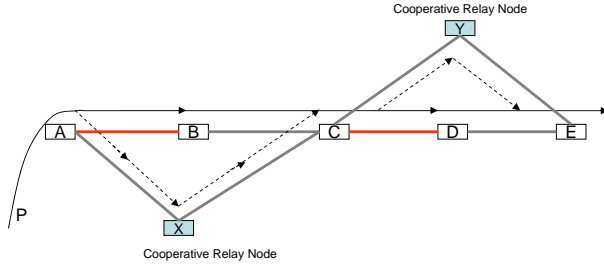


Figure 2: Path adaptation overview.

is a route computed by the routing protocol based on long term observation of the ETX metric [5] of each link of the network. Although alternative routes such as  $A - X - C - Y - E$  exist, let us assume that they have a higher cost as determined by the metric. Deflect provides a low overhead mechanism for fast switching to the alternate route depending upon the condition of links  $AB$ ,  $BC$ ,  $CD$ , and  $DE$ . The advantage of Deflect comes with the limitation that in the interest of quick response, it can adapt the path only using nodes in the neighborhood, and cannot do any end-to-end rerouting. Reconstruction of an overall better end-to-end path considering complete topology of the mesh network is still the responsibility of the routing protocol. But this activity incurs substantial overhead and can only be done over a longer time-scale.

In Deflect, a node such as  $X$  is called a *cooperative relay* for a node such as  $B$  that is the original relay node on a path.  $X$  monitors the performance of the  $AB$  and  $BC$  links (in terms of their delivery ratios) by overhearing the packets transmitted from  $A$  and from  $B$ . Now, if  $X$  determines, based on this estimation, that the product of the delivery ratios of  $AB$  and  $BC$  is worse than that of  $AX$  and  $XC$ , it signals node  $A$  so that  $A$  temporarily flips the next hop route through  $X$ . At this time,  $B$  can start playing the role of  $X$ . Similar relay switching can happen at any hop in an end-to-end path, for example, again at  $Y$ . Clearly, the scheme depends on the existence of such cooperative relay nodes  $X$  and  $Y$ . We show in Section 2.3 that in a dense, large-scale deployment network such cooperative relays are expected to be quite common. The scheme also depends on the accuracy of estimation of the link qualities for  $AB$ ,  $BC$ ,  $AX$  and  $XC$ . The estimation procedure is described in Section 3.1.

Note again that Deflect keeps the overhead low by restricting route changes only in the radio neighborhood of the original route computed by the routing protocol. It depends on available route diversity in this neighborhood. It does not change the end-to-end path length in number of hops, thus avoiding any sudden change in the delay

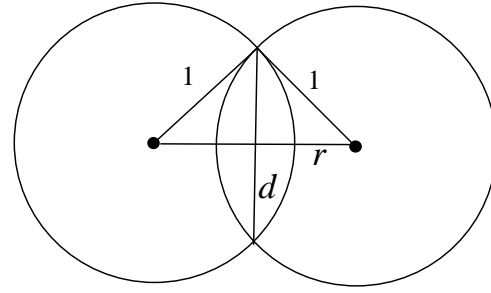


Figure 3: Unit disk graph model illustrating probability of existence of two relay nodes within communication range. The relay nodes must be within the intersection region of the two circles.

and major packet reordering problems. It does increase packet reordering; but – as our performance evaluation later shows – this is minor enough that can be easily handled by a small reorder buffer.

### 2.3 Existence of Cooperative Relays

The success of Deflect obviously depends on the existence of cooperative relays. In this section, we show that a large number of such cooperative relays do exist in dense networks. Specifically, we argue that for a given *pair* of two-hop paths (say  $A - B - C$  and  $A - X - C$ ) with common end points ( $A$  and  $C$ ), the relay nodes ( $B$  and  $X$ ) hear each other with high probability. This can be understood by using an idealized unit-disk graph model.

Assume that if two nodes are within radio communication range unity, they have a perfect link between them. Otherwise, they do not have a link. If the distance between the end points of a two-hop path is  $r$ , then  $1 < r < 2$ . The potential relay nodes must lie within the intersection of the unit circles centered at the end points. The maximum distance between any two points selected in this region is given by  $d = \sqrt{4 - r^2}$ , as shown in Figure 3.

We do a Monte Carlo simulation to determine the probability that the distance between any two relay nodes is less than unity. Figure 4 shows the likelihood of the distance between two relay nodes being less than 1 with increasing  $r$ . Averaging over  $r$  gives the probability that any two relay nodes for common endpoints can hear each other. This probability comes to 0.97, which is quite high. It can be argued that even though the two relay nodes ( $B$  and  $X$ ) can hear each other, they may be so close to each other geographically, that any interference affects both the nodes in similar fashion. So, we also consider a case when the relay nodes are separated by a distance of at least  $1/4$ , but can still hear each other. The probability of occurrence

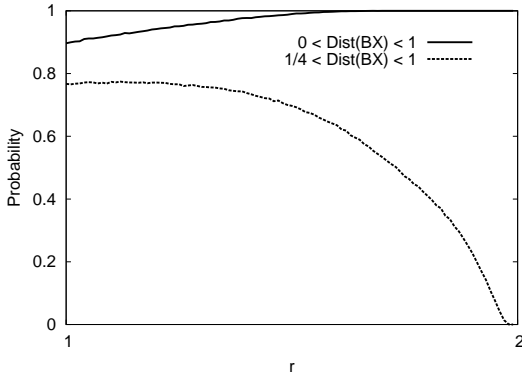


Figure 4: Probability that 2 relay nodes for a pair of 2-hop paths with common end points can hear each other. Probability of the two nodes being atleast 1/4 of the transmission range apart is also plotted.  $\text{Dist}(BX)$  is the distance between the relay nodes  $B$  and  $X$ .

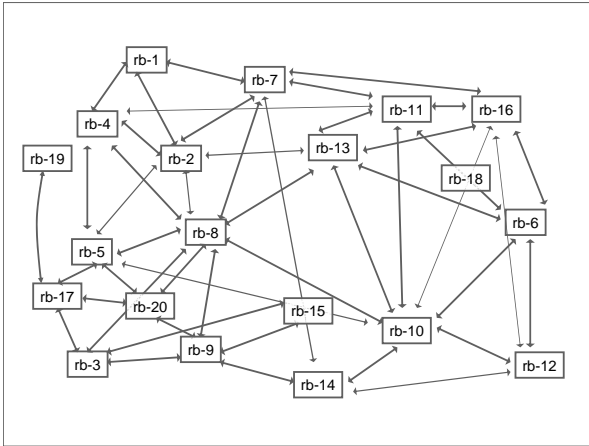


Figure 5: Indoor mesh network testbed. Only links with delivery ratio  $> 50\%$  are shown.

of such case is 0.6, as shown in Figure 4. This analysis shows that whenever there is a pair of two-hop paths with common endpoints, there is a high probability that the relay nodes can act as cooperative relays. Also, the probability that the two nodes do not undergo similar interference effects is sufficiently high enough for Deflect to be useful.

However, what is the guarantee that there is a pair (i.e., an alternate path) to start with? We argue that such redundancy is quite likely. In a recent study [1] on uncoordinated deployment of 802.11 access points in six urban areas in the US, the authors found that the median number of neighbors of an access point is about 4–7 (see Figure 1 in [1]). If we overlay a mesh network on these access points, such high degrees indicate good possibilities for such redundancies.

We obtained some numbers for existence of cooperative relays in two existing testbeds – the mesh testbed at

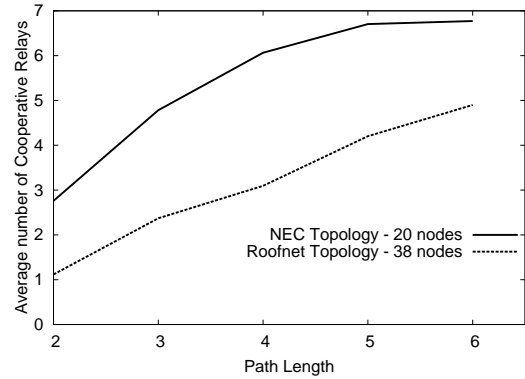


Figure 6: Number of cooperative relays as the path length increases. The topologies of NEC Labs testbed and MIT Roofnet are considered.

NEC Labs, shown in Figure 5, and MIT Roofnet project [16]. The density of mesh testbed at NEC Labs is greater than in Roofnet. We computed the number of cooperative relays for all possible paths of certain lengths in the networks. A path consists of links, where each link has a delivery ratio of atleast 50%. Then the path is broken into consecutive pairs of links, and we determine the number of possible cooperative relay for each such subpath. For every such link pair (A-B-C), we consider  $X$  as a cooperative relay, if it satisfies the following two conditions

- $|d_{AB} \times d_{BC} - d_{AX} \times d_{XC}| < 0.2$ , where  $d_{AB}$  is the delivery ratio of the link  $AB$ . This indicates that the difference between the delivery ratios along the two paths is not significant enough to choose one over another, and
- $1 < ETX_{BX} < 4$ , indicating that a link exists between  $B$  and  $X$ . Deflect does not need  $BX$  to be a perfect link. An  $ETX$  metric of 4 and lower denotes delivery ratios more than 50% in both directions  $BX$  and  $XB$ .

Figure 6 shows the average number of cooperative relays as the path length increases for the two networks considered. All possible paths in the network are considered. The result shows that enough cooperative relays exist in real networks for Deflect to be effective.

### 3 Deflect Design

This section is devoted to the complete design of Deflect including its testbed implementation. We start with the procedure for link quality estimation. The link quality is indicated by the delivery ratio on the link.

### 3.1 Estimating Link Quality via Passive Monitoring

For ease of discussion, we will refer to Figure 2, particularly the sub-topology involving nodes  $A, B, C, X$ . Assume that  $A - B - C$  is a part of the path computed by routing and  $X$  is the cooperative relay node. Node  $X$  snoops on the medium and estimates the delivery ratios of all links  $AB, BC, AX$  and  $XC$  following the procedure we describe here. In 802.11, links must work bi-directionally and thus, the bi-directional delivery ratios (delivery ratio for unicast traffic) must be estimated. Keep this in mind when reading the following procedures. We will discuss this issue further at the end of this subsection.

The estimation is dependent on the use of acknowledgments and sequence numbers in 802.11. In the 802.11 link layer, all unicast frames are acknowledged. If an acknowledgment is not received by the sender, the frame is retransmitted after a suitable backoff. This continues up to a retry limit (4 in our experimental setup) when the frame is dropped. Each frame generated in the link layer carries a 12 bit sequence number, between 0 and 4095. For each new frame, the sequence number is incremented modulo 4096. Retransmitted frames carry the same sequence number as the original frame. A bit in the MAC-layer header distinguishes between the original and retransmitted frames.

In the discussion that follows, the notation  $d_{AB}$  is used to denote the actual delivery ratio for link  $AB$  and  $d_{AB}^X$  is used to denote the delivery ratio for link  $AB$  as estimated by  $X$ . The notation  $t(AB)$  denotes the number of MAC-layer data frames transmitted on the link  $AB$  in a unit time interval.  $t^u(AB)$  denotes only frames with unique sequence numbers. The notation  $t(AB, X)$  denotes the number of frames transmitted on link  $AB$  as heard by node  $X$ . Thus,  $t(AB, X) \leq t(AB)$ .

**Estimating link AB:** The delivery ratio of link  $AB$  can be estimated by simply snooping all frames on link  $AB$  at node  $X$  and computing the fraction that carries a unique sequence number. Note that

$$d_{AB} = \frac{t^u(AB)}{t(AB)},$$

and

$$d_{AB}^X = \frac{t^u(AB, X)}{t(AB, X)}.$$

If losses on links  $AX$  and  $AB$  are completely independent,

$$\frac{t^u(AB, X)}{t^u(AB)} = \frac{t(AB, X)}{t(AB)}.$$

Then,

$$d_{AB}^X = d_{AB}.$$

**Estimating link AX:** The delivery ratio on link  $AX$  can be estimated by counting how many frames transmitted by

$A$  (on link  $AB$ ) are heard by  $X$ . Thus, no actual traffic on the link  $AX$  is needed. However, this is a harder problem as frames transmitted on link  $AB$  could contain one or more retransmitted frames for each originally transmitted frame. All retransmitted frames are identical. Thus, the number of retransmitted frames on  $AB$  cannot be accurately estimated via snooping. Snooped acknowledgment frames from  $B$  cannot be gainfully used either, since  $X$  has no guarantee that the sender  $A$  actually receives the acknowledgment. Because of these limitations, we simply estimate  $d_{AX}$  as

$$d_{AX}^X = \frac{t^u(AB, X)}{t^u(AB)}.$$

Now,  $t^u(AB)$  is unknown at  $X$  and must again be estimated. This is estimated by adding to  $t^u(AB, X)$  the number of ‘‘holes’’ in the sequence numbers among the set of packets used to compute  $t^u(AB, X)$ .

**Estimating link BC:** This procedure is similar to link  $AB$ . Note that here we exploit the ability of  $X$  to overhear  $B$  via link  $BX$ .

**Estimating link XC:** The delivery ratio from  $C$  to  $X$  in one direction is used as an estimator for link  $XC$ . The technique here changes depending on whether  $C$  transmits any frames (for any node). If true, the link  $CX$  can be estimated exactly as in link  $AX$ . Otherwise, the procedure is slightly more complex. Essentially, it involves estimating what fraction of ACK frame transmissions from  $C$  to  $B$  are heard by  $X$ . ACK frames do not carry sequence numbers. They also do not carry source MAC address. Thus, node  $X$  must correlate each overheard data frame to each overheard ACK frame. If  $X$  hears the data frame on link  $BC$ , but misses the corresponding ACK frame and does not hear a following data frame retransmission,  $X$  concludes that it has lost the ACK frame.  $d_{CX}$  is then estimated as follows:

$$d_{CX}^X = \frac{\# \text{ ACK frames heard}}{\# \text{ ACK frames heard} + \# \text{ ACK frames lost}}.$$

Note that in this method  $d_{CX}^X$  underestimates  $d_{CX}$ .

Recall that in 802.11 links are bi-directional, and thus delivery ratios must be estimated in a bi-directional sense. The above estimation procedure estimates the delivery ratios at node  $X$ . According to the procedure,  $d_{AB}$  and  $d_{BC}$  are estimated in the bi-directional sense; but  $d_{AX}$  and  $d_{CX}$  are estimated only in one direction (viz.,  $AX$  and  $CX$ ). In the absence of any further information, it is simply assumed that  $d_{AX}^X$  and  $d_{CX}^X$  are bi-directional estimators. It is possible to improve this estimation further, by observing how the bi-directional ETX metric differs from unidirectional ETX metrics on these links. This will require availability of ETX metric information to the Deflect sublayer. In our experience, however, the simplest mechanism works quite well in practice.

Finally, path  $A - B - C$  should be switched to  $A - X - C$  if  $d_{AB}d_{BC} < d_{AX}d_{XC}$  where all delivery ratios are computed in the bi-directional sense.

### 3.2 Path Adaptation Heuristic

In the previous subsection, we have described the general strategy for estimating link qualities by cooperative relay nodes. In general, more than one cooperative relay nodes are available for a two-hop path, depending on the network topology and link qualities. Any node satisfying the topological conditions discussed in Section 2.3 can offer to be a cooperative relay for a two-hop path. A node can also offer to be a relay for multiple paths. It is possible for any node  $X$  to verify whether it can act as a cooperative relay for a two hop path  $A - B - C$  by simply overhearing transmissions on links  $AB$  and  $BC$ , and thereby ascertaining actual occurrence of packet forwarding on path  $A - B - C$ . Note that this can be determined just via snooping, even without access to any routing layer information. This is important as one of our design goal is to implement the mechanism below, and transparent to, the routing layer.

Once a node (say,  $X$ ) determines that it is a potential cooperative relay, it continues to evaluate the link qualities as detailed in Section 3.1. The delivery ratios are estimated in *time windows* of length  $H$ .  $H$  defines the path switching time granularity and must be chosen carefully. Too small a value will make the statistics, and hence switching decision unreliable. On the other hand, too large a value will tend to average out short term link fluctuations. In general,  $H$  should be much less than ETX metric measurement intervals, and thus  $H$  should not exceed a few seconds. The lower limit on  $H$  depends on the traffic on the path  $A - B - C$ , since the rate at which samples can be collected to estimate the link quality will be proportional to traffic. This means that with slower traffic (in packets/sec)  $H$  must be larger. However, effectiveness of the path adaptation is best observed with faster traffic where a number of consecutive packets can be lost due to burst errors. Thus, our interest here is in supporting streaming/interactive media traffic such as VoIP (20ms inter-packet time in one direction), where  $H$  can be kept reasonably small (a few hundreds of milliseconds).

A second parameter of importance is a *sensitivity* parameter. This is indicated by a threshold  $T$ , by which the estimated delivery ratio on the alternate path ( $d_{AX}d_{XC}$ ) must exceed the estimated delivery ratio on the primary path ( $d_{AB}d_{BC}$ ), before the path is switched. Note that the use of parameters  $H$  and  $T$  are similar to their use in the Divert [18]. As suggested in [18], it is also possible to adapt the values of  $H$  and  $T$ . However, in this paper, we limit ourselves to statically chosen values. Adapting the values with traffic and link behaviors is a topic of further research.

### 3.3 Path Switching

The path switching protocol is straightforward. All potential cooperative relays (i.e., nodes like  $X$  in Fig 2) passively monitor the link qualities continuously on the two-hop paths it is able to bypass. The procedure outlined in section 3.1 is used to determine the estimated delivery ratios. If the estimated delivery ratio on the alternate path exceeds that in the primary path by a threshold  $T$  in a time window  $H$ , the cooperative relay transmits a *signaling message* to the source of the two-hop path (i.e., node  $A$ ) notifying the availability of a better path. The message contains the ID of the cooperative relay ( $X$ ) and the ID of the nodes in the path it can help bypass (i.e.,  $A - B - C$ ). The source node at this point simply switches the next hop to the cooperative relay ( $X$ ). After a path is switched, the role of the original relay node (e.g.,  $B$ ) and the cooperative relay (e.g.,  $X$ ) is reversed. This happens automatically, and not through explicit messaging. Thus, the path can be switched back to the original later when the link qualities change.

Deflect sublayer is implemented beneath the routing or packet forwarding layer. It uses a simple mechanism to keep track of multiple paths that may multiplex at the cooperative relay and have the same prior hop. Again referring to Figure 2, the relay node  $X$  maintains a list of destination IP addresses of packets that it heard along the subpath  $A - B - C$  during the last measurement interval. When  $X$  takes over the relaying and informs  $A$  about the alternate path  $A - X - C$ , it also sends  $A$  this list. Now assume that  $A$  chooses  $X$  as the new relay node. For every outgoing packet whose destination MAC address is  $B$ , the Deflect sublayer at node  $A$  checks whether the packet's destination IP address exists within the list sent by  $X$ . If it does exist, it changes the destination MAC address of that packet from  $B$  to  $X$ . This check ensures that only those packets that were previously routed via path  $A - B - C$  are now re-routed via  $A - X - C$ . The Deflect sublayer of  $X$  also checks on every packet received from  $A$  to verify whether the destination IP address is in the list it just constructed. If so, it transmits the packet to node  $C$  without sending the packet up to the routing or packet forwarding layer. Any other packet (that is not "deflected") received from  $A$  must be targeted to a different destination, and is sent up to the routing or packet forwarding layer.

Note that when a path is switched, it is possible for Deflect to notify routing. However, our design goal in this work is to operate Deflect *underneath and transparent to routing*. This way, long term and end-to-end path changes are generated by the routing protocol and short term and local changes are generated by Deflect. At this time, we do not explore any further tighter coupling between these two protocol layers. Also note that Deflect cannot introduce any routing loops not present in the routing protocol. This is because it simply replaces a two-hop sub-path by

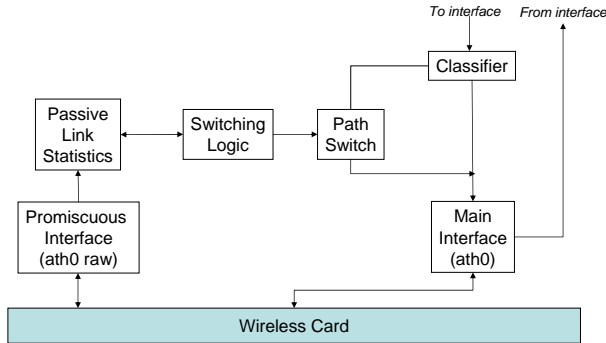


Figure 7: Block diagram describing operation of Deflect beneath the routing/packet forwarding layer.

another with the same end points. Deflect is dependent on the assumption that nodes are cooperating to improve overall performance, much like routing protocols in ad hoc or mesh networks. There are indeed security implications of path switching, similar to any routing protocols. We expect that security issues can be addressed via authentication of signaling messages from  $X$ . However, we do not explore this issue here.

### 3.4 Implementation

We have implemented Deflect on Linux using *Click* modular router [11]. The latest `madwifi` device driver [13] is used in our testbed that works in 802.11/a/b/g interfaces with the Atheros chipset. It allows creation of an additional raw *virtual* device (`ath0raw`) for each wireless interface that allows reception of all 802.11 frames (control, management, data) as if in monitor mode, while the main interface can still operate in the ad hoc or infrastructure mode. This raw virtual interface is used in the passive monitoring procedure. A new Click element is implemented that estimates the delivery ratios on different links passively. See Figure 7. All frames received on the raw virtual interface are processed by this element to update various link quality related statistics. Then the statistics is evaluated in regular intervals ( $H$ ) to check if there is any benefit of switching the current path. The switching logic uses this estimation to send the signaling message to the source node.

Some implementation choices have been made for simplicity. In order not to overload a cooperative relay node of monitoring multiple number of two hop paths, we restrict relay nodes to monitor at most 2 paths at a time. In general, more than one cooperative relay can exist for a single 2-hop path. Thus, multiple such cooperative relays can signal the source node to take over relaying. In our implementation, the source node simply accepts the first one that volunteers and ignores the rest.

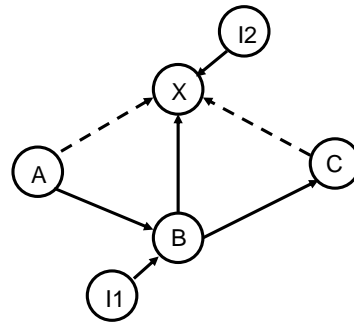


Figure 8: Experimental setup to demonstrate effective estimation of link quality. The interferers  $I1$  and  $I2$  cause packet errors primarily at  $B$  and  $X$  respectively.

## 4 Experimental evaluation

We have conducted several experiments to evaluate the performance benefits of Deflect compared to a conventional routing protocol using the ETX metric. Experiments were conducted on an indoor mesh network testbed consisting of 20 nodes deployed in NEC Labs. Each node is a Routerboard 200 series processor board [24] with 256MB of RAM, 512MB of Compact Flash for secondary storage, and an Atheros-based 802.11a/b/g card. 802.11b mode is used for the experiments reported here. The ETX metric for each link was collected by existing software elements (`ETXMetric` and `LinkStat`) that are part of the Click Toolkit. The evaluation is done in two parts – (i) Evaluating the accuracy of the passive estimation technique; (ii) Characterizing the performance of the path switching mechanism in terms of its reaction time and packet reordering, and resultant benefits on metrics such as delivery ratio, delay and jitter.

### 4.1 Evaluation environment

In order to investigate and characterize the performance of link estimation and path switching in Deflect, we used a controlled environment for generating experimental scenarios. In order to avoid unpredictable effects of external radio sources, all experiments were carried out during night. One node (called the *interferer*) generates interference traffic on a target link. The interference traffic is generated in *bursts* – with alternating on and off periods. The *duty cycle* (the relative duration of on and off periods) is fixed, but the absolute duration of on and off periods is varied to control the dynamic nature of the interference. The interference load (in Kbps) during the on period is also varied. Finally, the location of the estimator node (cooperative relay) and transmit power level of the interferer are also varied between experiments. The interferer

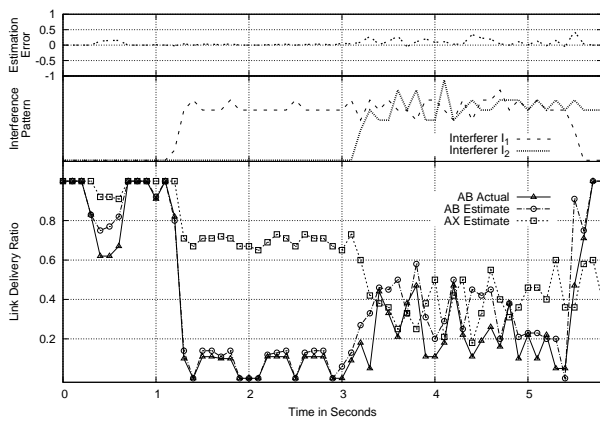


Figure 9: Trace demonstrating the potential benefit of path adaptation and accuracy of link quality estimation.

operates on an overlapping channel because our experience has shown that this makes the interferer more likely to succeed on CCA (clear channel assessment), while causing collision at the receiver.

## 4.2 Accuracy of Link Quality Estimation

Deflect primarily works by passively estimating link qualities of neighboring nodes. We have performed an extensive set of experiments to evaluate the accuracy of the estimation technique in various set ups. For brevity, we present a careful set of evaluations focusing on four nodes in our testbed (in Figure 5), acting as nodes  $A, B, C, X$ . The nodes are  $A = rb-7$ ,  $B = rb-15$ ,  $C = rb-3$ , and  $X = rb-8$ .

In order to provide illustration of the passive estimator in action, Figure 9 shows an example 6 sec trace of packet delivery ratios on link  $AB$  and its estimate at  $X$  (i.e., the quantities  $d_{AB}$  and  $d_{AB}^X$ ) with sampling time window  $H = 100ms$ . In this case, there are two interferers in action. Interferer  $I_1 = rb-12$  generating interference on node  $B$  primarily, and interferer  $I_2 = rb-5$  generating interference on node  $X$  primarily (Figure 8). The traces in Figure 9 also show the estimated delivery ratio on link  $AX$  (i.e.,  $d_{AX}^X$ ). The behavior of the two interferers  $I_1$  and  $I_2$  are shown in terms of number of packets transmitted. Note that there are times when only one interferer is on and also times when both interferers are on. Figure 9 shows that actual and estimated delivery ratios on the  $AB$  link are very close, and the estimated delivery ratios for  $AB$  and  $AX$  links follow the interference patterns, with delivery ratios going down when the respective interferers are on.

In the next set of experiments, we study the estimation of the quality of link quantitatively. As a start, we separate two types of errors: positive ( $d_{AB}^X < d_{AB}$ ) and negative ( $d_{AB}^X > d_{AB}$ ). In Deflect negative error is less desir-

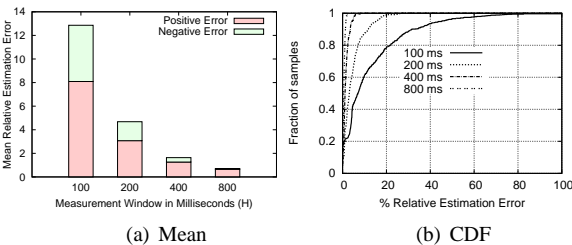


Figure 10: Effect of sampling time window ( $H$ ) on relative estimation error (%).

able as it may result in unnecessary path switching (false positive decision). We compute the relative estimation error (as a percentage) as  $100.0 \times [(d_{AB}^X - d_{AB})/d_{AB}] \%$  in each sampling time window period ( $H$ ), and present statistical data (mean and CDF) for the error. When we present mean relative estimation error, we not only present the mean of the absolute value of the error, but also separate out positive and negative components to show their relative contributions. When we plot CDF, we again plot the CDF of the absolute error.

**Effect of sampling time window ( $H$ ):** We consider only interferer  $I_1$ . The interferer is on for  $w = 100ms$  and off for 300ms (duty cycle = 25%). When it is on, it transmits broadcast packets at 2Mbps. Figure 10(a) shows the mean relative estimation error for different values of  $H$ . Figure 10(b) provides a more detailed information related to the distribution of the error for different  $H$ . We observe that the mean error reduces from 13% to 5% by increasing  $H$  from 100ms to 200ms – a significant drop. It drops further for larger  $H$ . Increasing  $H$  further reduces the error, but also makes the path switching less reactive. A value of 200ms for  $H$  is thus a good tradeoff. The 5% estimation error can be compensated by choosing a larger threshold  $T$  that triggers the path switching.

**Effect of load on link  $AB$ :** High load may result in relatively higher fluctuation of the actual delivery ratio on  $A - B$  for a given interference pattern. A possible reason can be the MAC unfairness at higher load. Higher fluctuation will increase the error in estimation. To understand the effect, we vary the load from 500Kbps to 2Mbps (limited by the effective throughput of a 2 hop path with 802.11b card). Figure 11(a) and Figure 11(b) show the dependency of the relative estimation error on load. Even though the graph shows an increasing trend for the error with load, we note that change of mean error is less than 4% for an increase of load by a factor of 4 (0.5Mbps to 2Mbps).

**Sensitivity to time varying interference:** We generate different interference patterns by changing the on and off periods, but keeping the duty cycle constant at 25%. Smaller on or off periods result on more fluctuation of the interference load and thus of the delivery ratio. This ex-

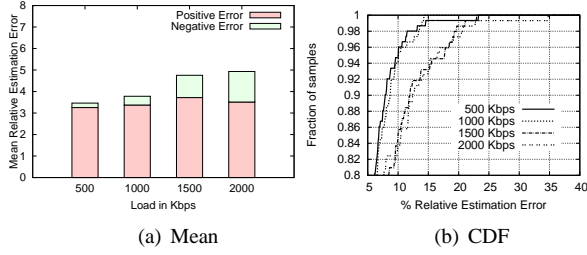


Figure 11: Effect of load on  $AB$  on relative estimation error (%).

periment is designed to understand how fast the estimator can catch up with the actual delivery ratio when the latter fluctuates over time. We use the lowest sampling time window ( $H = 100ms$ ). Figure 12(a) and Figure 12(b) show the dependency of relative estimation error for different on (burst) periods. We note that the mean error ranges from 6% to 12%. From Figure 12(b), we observe that around 80% of the samples have less than 15% error, when the on period is 200ms or more. These results together with the results in Figure 10(a) suggest that the passive estimation can react in the granularity of 200ms or more.

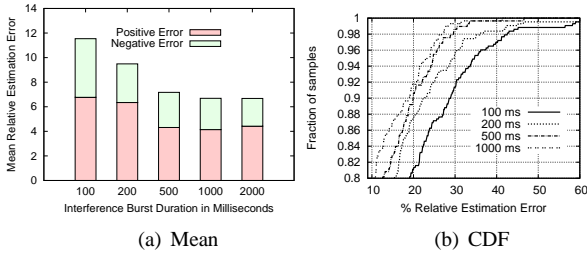


Figure 12: Relative estimation error (%) in time-varying interference pattern.

**Sensitivity to interference level:** Here, we vary the interference level by setting different transmit power levels of the interferer  $I_1$ . The goal of this experiment is to understand the dependency of the error on the interference power. From the Figure 13(a) and Figure 13(b), we note that increasing interference power increases the error. This is likely because the increasing power also results in interference (*albeit* weaker) at the estimator node  $X$ . Interference at  $X$  results in reduction in the number of samples and/or distorting the samples collected leading to increase in error.

**Sensitivity to the location of the estimator node:** In this experiment, we try to find if the estimation error depends upon the location of the estimator. For the two hop path  $A - B - C$  considered in the mesh testbed, we have three nodes that can act as cooperative relay  $X$ . Figure 14(a) and 14(b) show the estimation error for these

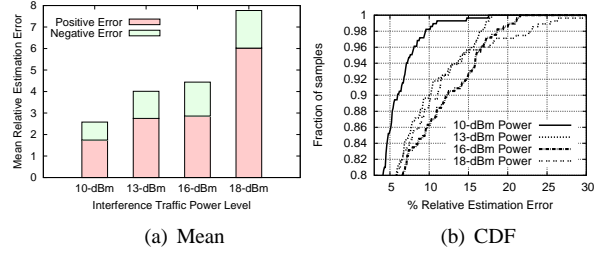


Figure 13: Relative estimation error (%) for different interference levels.

three nodes. We note that error is almost the same for all the three nodes. The difference is less than 1%. Figure 14(b) shows that the error distribution in the samples is similar for all the three locations as well.

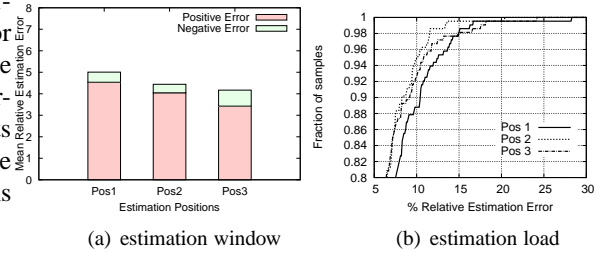


Figure 14: Relative estimation error (%) for different choices of  $X$ .

**Summary of results:** From the above pathological cases studied in the experiment, we can make the following important conclusions: a) with the right choice of parameters the link quality estimation is quite accurate even for a wide variety of interference conditions; b) the negative errors are usually smaller (often much smaller) than positive errors, preventing the protocol from over-reacting; c) estimation error is not dependent on the choice of the estimating node (cooperative relay), providing a great diversity of relay choices; d) a sampling time window as small as 200ms is reasonable to keep errors low (less than 5% overall, with negative error less than 2%).

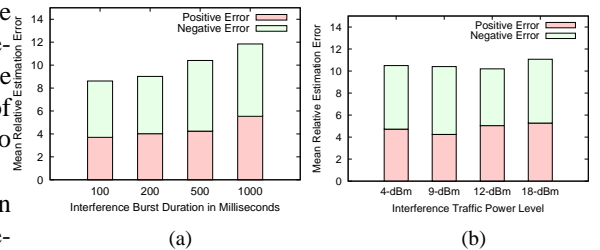


Figure 15: Relative estimation error for link  $AX$ .

**Estimating Link  $AX$ :** So far, we concentrated on link

$AB$  only. Estimating link  $BC$  is similar, as explained in Section 3.1. We also present some results related to link  $AX$ . Figure 15(a) shows the relative estimation error for different on (burst) periods of the interferer, where the sampling time period  $H = 100\text{ms}$ . We observe now that the error increases with increase in burst period. This trend is different from what we have seen before for link  $AB$ . This is because with more stable interference, packet collision probabilities are stable for longer durations of time. This presents less samples to the estimator in certain periods. Changing the power level of interferer in this case seems to have little impact on the estimation error (Figure 15(b)). In general, we note that estimation errors are generally small, often under 10%.

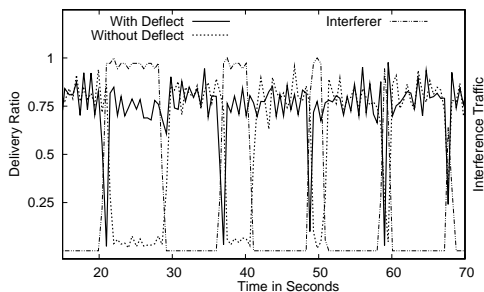


Figure 16: Trace demonstrating the benefit of Deflect in presence of time varying interference.

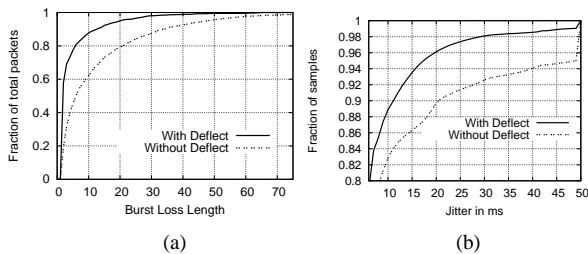


Figure 17: CDF of (a) burst loss length (in packets), (b) delay jitter (in ms) with and without Deflect.

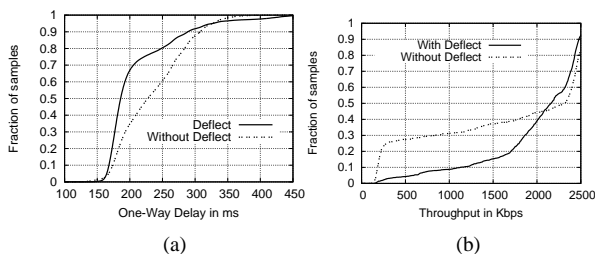


Figure 18: CDF of (a) one-way delay (in ms), (b) instantaneous throughput (in Kbps) with and without Deflect.

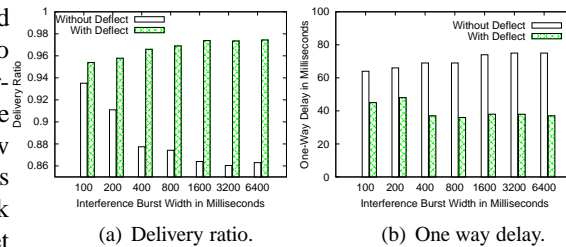


Figure 19: Performance of Deflect with different interference burst sizes.

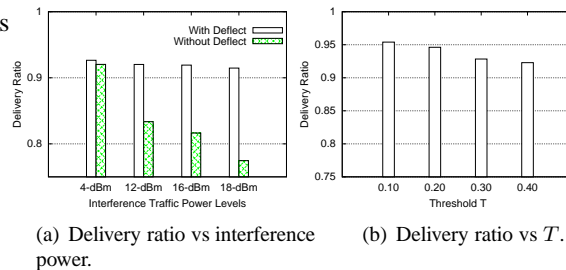


Figure 20: Performance of Deflect with different interference power and  $T$  values.

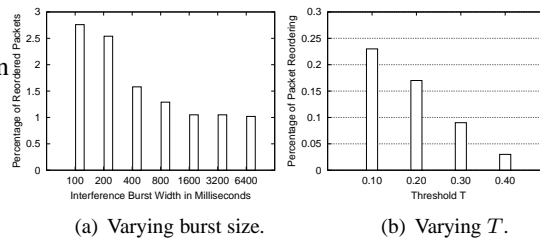


Figure 21: Amount of Packet reordering.

### 4.3 Evaluation of Path Switching

In order to better understand the benefits, we use an implementation of the OLSR routing protocol [3] using ETX as the link quality metric. Then, we run Deflect underneath routing. An illustration of the benefit of using Deflect is in the trace given in Figure 16. When the interferer is on, it transmits back-to-back broadcast packets. The trace uses interference bursts that are usually of a shorter duration than the ETX evaluation period (10sec). We notice that with Deflect, delivery ratio is quite stable due to path switching.

For a more quantitative characterization of benefits, we focus on four performance measures important for streaming/interactive media: *burst loss length*, *jitter*, *one-way delay* and *throughput*. Figure 17(a) shows that at the 80-percentile point Deflect reduces the burst loss length to about one third. The median value is also less than half. Figure 17(b) shows less jitter as well with Deflect, though

the improvement here is less dramatic. In Figure 18(a), we observe similar benefits in terms of one-way delay. In particular, 70% of the samples has delay less than 200ms using Deflect compared to only 34% of the samples without using Deflect. Figure 18(b) shows the throughput for routing with and without Deflect, where again we observe that almost 30% of the time, throughput without Deflect is below 500Kbps. In general, we get many more low throughput samples when Deflect is not used.

**Reaction speed of Deflect:** To study how fast Deflect can react to changing interference, we conduct a series of experiments by using on-off interference as in the previous subsection, and vary the on (burst) period of the interferer. The interferer, when on, transmits back-to-back broadcast packets. We keep the duty cycle the same as before at 25% so that the total amount of interference is kept constant. Figure 19(a) shows the delivery ratio with and without Deflect. Lower burst duration implies a faster changing interference pattern. From the figure, we observe that even for burst size of 100ms, Deflect provides some benefit. With 200ms and more, the gain is significant. A similar trend is observed from one-way delay as shown in figure 19(b). As one would expect that Deflect’s ability to react fast also results in some packet reordering. We present the percentage of reordered packets in Figure 21(a). We observe that the percentage of reordered packets stays quite low, between 1% to 2.8%. We observe only very minor reordering without Deflect.

**Effect of interference power and threshold  $T$ :** The delivery ratio also depends upon the interference level. Figure 20(a) shows this. It also shows that Deflect is robust against power. Delivery ratios for Deflect with varying threshold  $T$  is shown in Figure 20(b). Note decreasing performance with increasing threshold. Reordering also changes with  $T$  (Figure 21(b)), with more reordering will lesser  $T$  as path switches are more frequent.

**Effect of Link Outage:** In our mesh testbed deployed at NEC labs, we found frequent occurrence of mesh nodes going down. We attribute this unstable behavior to exceptions occurring in the embedded hardware and bugs in the software. Deflect can detect such outages and quickly reroute the traffic via an alternative path. To study the reaction time of Deflect, we simulated node outages for short intervals of time by shutting down the wireless card on the node. Figures 22(a) and 22(b) show the performance of Deflect for different outage periods.

**Ambient interference:** Here, instead of using a controlled interference pattern, we study the impact of ambient wireless interference on routing with and without Deflect. To study this, we ran CBR UDP traffic on several isolated 2-hop paths in our mesh testbed during working hours, when a large number of WLAN APs and clients operate in the vicinity of our mesh testbed. We ran the experiment for 8 hours and collected statistics based on

delivery ratio samples collected every 2 seconds. Figure 23(a) demonstrates the gain with Deflect in the presence of such ambient interference. Figure 23(a) shows a 3 second trace showing instantaneous throughput with samples collected at more frequent intervals.

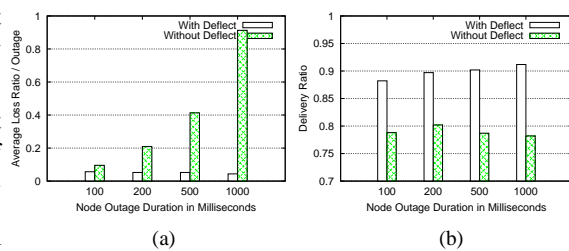


Figure 22: Performance of Deflect under link outages. (a) Loss ratio per outage, (b) Delivery ratio.

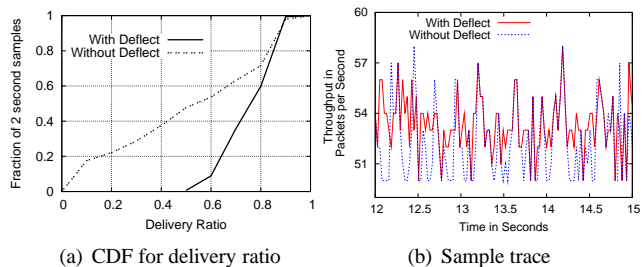


Figure 23: Performance of Deflect under ambient interference conditions.

#### 4.4 Performance of Real Time Applications

We have also studied the benefit of Deflect for supporting VoIP in multihop mesh networks. For the G.729 voice codec used in the experiment, the VoIP quality is specified by the  $R$ -score [4].  $R$ -score is determined based on loss (from network and jitter buffer) and delay using the formula given in [20]. A higher  $R$ -score indicates a better VoIP quality. VoIP traffic for G.729 is generated using a stream of 60 Byte UDP packet sent every 20ms. We generated 4 VoIP call in the experiment and created a interference pattern as shown in Figure 24. Figure 24 shows that Deflect switches paths as the interference turns on within a few seconds (with occasional noisy changes). If Deflect is not used, routing sticks to the same path (path 1). The  $R$ -score with Deflect stays close to 80, while without Deflect,  $R$ -score fluctuates with the interference, often going down to 0.

For our second application, we use mpeg4 streaming of the commonly used Foreman video. To measure the video quality we use the Peak Signal to Noise Ratio (PSNR) as the video quality metric. For every frame in the video, the

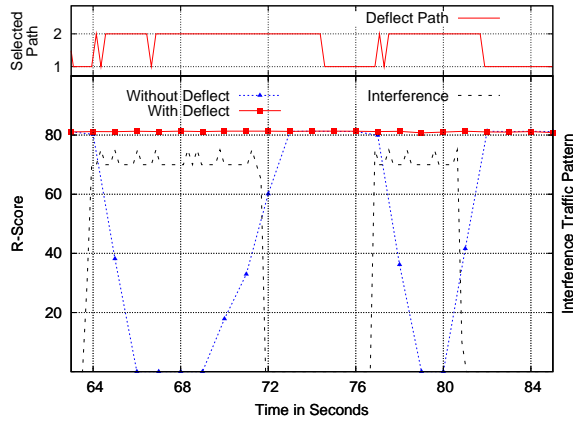


Figure 24: Sample trace showing the performance of VoIP calls with and without Deflect.

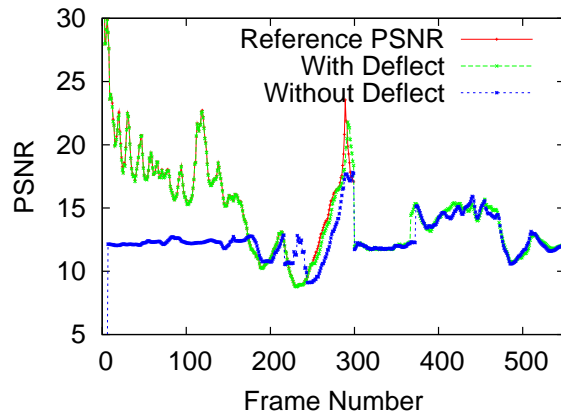


Figure 25: Sample trace showing the PSNR values for a video stream with and without Deflect.

PSNR value was computed and compared with the Reference PSNR (the PSNR of the frame in the original video). Results on PSNR was collected based on a segment of the video where several random interferers are turned on in the neighborhood only for the first half of the video segment. The Figure 25 shows the PSNR for each frame with and without Deflect, while comparing it with the reference PSNR. We note that for the first 200 frames, there is a significant difference between the PSNR without Deflect and with Deflect. Over the whole span, the reference PSNR is almost closely followed by using Deflect.

#### 4.5 Snooping Overhead

Deflect is based on snooping in promiscuous mode. This causes an additional CPU overhead on the cooperative relay nodes. We conducted a set of experiments on our Routerboard nodes to find out how much impact this overhead has on the forwarding capacity of a relay node. To

Load in link $AB$ (Mbps)	Max UDP throughput (Mbps) at $X$	
	w/o snooping	with snooping
0.00	10.0	10.0
10.0	6.40	6.11
20.0	5.67	5.51
28.0	4.76	4.24

Table 1: Impact of snooping on achievable throughput

do this, we increased the load on the link  $AB$  in order to increase snooping overhead on part of  $X$ . In 802.11b we observed no change in the forwarding capacity at  $X$  even when link  $AB$  carries saturation traffic. It turns out that 802.11b link rate is not high enough to cause any CPU bottleneck. To get higher link rate, we switched the interface to 802.11g. For this case, the table 1 shows the maximum forwarding capacity at  $X$  with and without snooping. There is indeed a small reduction in capacity when snooping load is high. But we do not consider this to be a significant issue as the CPU power in embedded router systems is steadily increasing.

## 5 Related Work

There has been a significant interest in exploiting link and path diversity to improve performance of wireless networks. Two recently proposed approaches – *Divert* [18] and *ExOR* [2] – are most related to our work. In *Divert* [18], fine-grain path selection is used in a wireless LAN environment. If the link between a client and the access point (AP) degrades according to a short-term frame delivery statistics, an alternative AP is used for communication without causing an actual handoff. Deflect unlike *Divert*, works in a multihop network, and does not rely on a wired distribution system for coordination between nodes. The uniqueness of Deflect is in its sophisticated passive link estimation technique, the absence of which makes path adaptation difficult in mesh network.

*ExOR* [2] uses an opportunistic forwarding framework in a multihop network, where packets at each hop are broadcast instead of unicast. A coordination protocol is used to determine which node – among those that received the packet correctly – will actually forward the packet further towards the final destination. Such coordination is needed to ensure that more than one node do not end up forwarding the same packet. This coordination could be complex and expensive in a dense network, where there could be many potential forwarders. *ExOR* implements this coordination via a gossiping protocol that works by breaking down the packet transmissions in batches, and including a state of successful packet transmissions in packet headers. Because of the nature of the gossiping protocol, *ExOR* is useful only for bulk transfer applications, and not for real-time applications that Deflect tar-

gets. Also, the scalability of the gossiping protocol is unclear, as it requires per-flow state maintenance.

Multipath routing protocols [15, 19, 21] require the explicit computation of multiple paths during route computation, and can rank them in quality based on a chosen routing metric. Such protocols essentially provide the packet forwarding layer with multiple next hop alternatives. The packet forwarding layer can now switch paths when packet losses occur in a next hop link. Here, the forwarding node must switch paths to an alternative next hop *without any knowledge of the instantaneous condition of the potential alternative next hops*. This may boil down to a blind search for the best alternative. In contrast, Deflect does not explicitly search for alternate paths. Cooperative relays in the neighborhood of the original path simply estimate whether they can offer better alternatives, and they – instead of the forwarding node – take the initiative of path switching. No blind search is needed.

Some routing protocols for mobile ad hoc networks attempt to improve on path lengths by snooping on the radio medium. Examples include DSR protocol [14, 10] and SHORT protocol [7]. These protocols are primarily directed for quickly “shortening” paths in a mobile network, where some nodes may have stale routing information, but can overhear more up-to-date information providing better routes. Our path adaptation approach is also based on snooping, but it is used to estimate link quality metrics directly.

A number of MAC-layer protocols have been developed to determine the forwarding hop in a multihop network based on instantaneous channel condition. Selection diversity forwarding [12], and MAC-layer anycasting [8, 25] are based on this idea. The GeRaF protocol [29] is also similar, but is only applicable when geographic routing is used. A number of similar protocols work in the context of wireless LANs. Prominent examples include the MAD (medium access diversity) protocol [9] and opportunistic packet scheduling protocol [28]. They exploit multiuser diversity at the AP in the AP-to-client communication by probing channel conditions at multiple clients and giving certain preference to clients with good channel conditions. All these approaches require a new MAC layer. In contrast, Deflect works on existing 802.11 MAC and commodity radios.

Packet combining techniques have been used in literature to obtain diversity. In the MRD (multi-radio diversity) technique [17] multiple radios are simultaneously used to receive the same transmission. This technique is targeted for wireless LAN scenarios. For uplink diversity, this can be done using more than one AP that has overlapping coverage. For downlink diversity, more than one radio interface can be used in the client. When multiple copies of the same frame are received with errors by the different interfaces, they can be combined to recover from

the error.

Finally, communications and information theory community have been looking at the concept of cooperative diversity for some time. The idea is somewhat similar to packet combining as above, but uses relaying of the received signal in the physical layer. See, for example, [27, 26]. Much of these ideas have so far been in the realm of theory, because of restrictive assumptions on channel models, functionalities at the physical layer, synchronization, and traffic behavior.

## 6 Conclusions

In this work, we designed and implemented Deflect, a fine grained local path adaptation mechanism that works underneath the routing layer. Deflect provides a zero overhead mechanism for estimating link qualities, allowing link quality probing to be done at a fine granularity of 200ms. Using the link quality estimation, Deflect enables fast local adaptation of end-to-end routes in response to sudden drop in link delivery ratio or node failures. Our experiments shows that the error in estimation is less than 5% with 200ms probe interval. Deflect can provide a higher delivery ratio in presence of time varying interference. Deflect is particularly useful for streaming applications where short term fluctuations in link qualities can severely degrade performance. Additionally, our results also verify that a) the packet snooping overhead does not lead to sacrificing the forwarding capacity of the relay node and b) the packet reordering from path switching in Deflect is less than 4%.

## References

- [1] A. Akella, G. Judd, P. Steenkiste, and S. Seshan. Self Management in Chaotic Wireless Deployments. In *Proc. of ACM Mobicom*, 2005.
- [2] S. Biswas and R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. In *Proc. of SIGCOMM*, 2005.
- [3] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, October 2003.
- [4] R. Cle and J. Rosenbluth. Voice over IP performance monitoring. In *ACM Computer Communication Review*, 2001.
- [5] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom 03)*, September 2003.

- [6] R. Draves, J. Padhye, and B. Zill. Comparison of Routing Metrics for Static Multi-hop Wireless Networks. *SIGCOMM Comput. Commun. Rev.*, 34(4):133–144, 2004.
- [7] C. Gui and P. Mohapatra. SHORT: Self-healing and Optimizing Routing Techniques for Mobile Ad hoc Networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 279–290, 2003.
- [8] S. Jain and S. R. Das. Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks. In *Proc. of IEEE WoWMoM Symposium*, June 2005.
- [9] Z. Ji, Y. Yang, J. Zhou, M. Takai, and R. Bagrodia. Exploiting Medium Access Diversity in Rate Adaptive Wireless LANs. In *Proc. ACM MobiCom Conference*, 2004.
- [10] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In C. E. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001.
- [11] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.
- [12] P. Larsson. Selection Diversity Forwarding in a Multihop Packet Radio Network with Fading Channel and Capture. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5:79–282, October 2001.
- [13] MADwifi. Multiband ATHEROS Driver for WiFi. <http://www.madwifi.org>.
- [14] D. Maltz, J. Broch, J. Jetcheva, and D. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communication*, 17(8), August 1999.
- [15] M. Marina and S. R. Das. On Demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, Dec. 2001. To appear.
- [16] MIT Roofnet. <http://www.pdos.lcs.mit.edu/roofnet>.
- [17] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *11th ACM MOBICOM Conference*, Cologne, Germany, September 2005.
- [18] A. K. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: Fine-grained Path Selection for Wireless LANs. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, MA, June 2004.
- [19] A. Nasipuri, R. Castaneda, and S. R. Das. Performance of Multipath Routing for On-Demand Protocols in Ad Hoc Networks. *ACM/Kluwer Mobile Networks (MONET) Journal*, 6(4):339–349, 2001.
- [20] D. Niculescu, S. Ganguly, K. Kim, and R. Izmailov. Performance of VoIP in a 802.11b wireless mesh network. In *Proc. of the IEEE INFOCOM'06 Conf.*, 2006.
- [21] M. Pearlman, Z. Haas, P. Scholander, and S. Tabrizi. On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc Networks. In *Proc. of ACM MobiHoc 2000 Workshop*, August 2000.
- [22] C. Perkins, E. Royer, and S. R. Das. Ad hoc on demand distance vector (AODV) routing. RFC 3561, July 2003.
- [23] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proc. of the ACM SIGCOMM '94 Conference*, pages 234–244, August 1994.
- [24] Routerboard. <http://www.routerboard.com>.
- [25] R. RoyChoudhury and N. Vaidya. MAC-layer Any-casting in Ad hoc Networks. *Computer Communication Review*, 34(1), 2004.
- [26] A. Scaglione and Y. W. Hong. Opportunistic Large Arrays: Cooperative Transmission in Wireless Multihop Ad hoc networks to reach far distances. *IEEE Transactions on Signal Processing*, 51(8), 2003.
- [27] B. Sirkeci-Mergen and A. Scaglione. Continuum Approach to Dense Wireless Networks with Cooperation. In *IEEE Infocom*, 2005.
- [28] J. Wang, H. Zhai, and Y. Fang. Opportunistic Packet Scheduling and Media Access Control for Wireless LANs and Multi-hop Ad Hoc Networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'04)*, 2004.
- [29] M. Zorzi and R. R. Rao. Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance. *IEEE Trans. Mob. Comput.*, 2, 2003.