SHORT: A Static-Hybrid Approach for Routing Real Time Applications Over Multichannel, Multihop Wireless Networks[*]

Vijay Raman and Nitin H. Vaidya

ECE & Coordinated Science Lab
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA.
{vraman3,nhv}@illinois.edu

Abstract. Many of the existing multichannel wireless network implementations rely on channel switching capability of the wireless radios to ensure network connectivity. However, due to both software and hardware restrictions switching channels incur a significant delay, which can be prohibitive for many delay sensitive, real time applications, such as VoIP and interactive gaming. The situation can be worse in the case of a multihop network, as every node along the traffic path may require a channel switch that adds up to the overall end-to-end delay. This motivates the need for efficient routing strategies that can make use of the flexibilities of a multichannel network while favoring delay sensitive applications by routing them on low delay paths. In this paper, we propose SHORT, a *S*tatic-*H*ybrid approach for r*O*uting *R*eal *T*ime applications over multichannel, multihop wireless networks, which ensures low delay paths for delay sensitive applications. Using measurements on a real multichannel testbed, we show that our protocol can provide significantly low delay multihop paths for delay sensitive applications (eg., VoIP) without degrading the throughput performance of non-delay sensitive, best effort traffic, such as TCP that may co-exist in a network.

Keywords: multichannel routing; VoIP over multichannel; channel switch delay; static channel allocation.

1 Introduction

Multichannel wireless networks are gaining popularity due to the variety of flexibilities that they can offer [1, 2]. For instance, when nodes in a network are tuned to different channels, the amount of contention on any single channel is reduced. Moreover, when we use orthogonal channels, the overall interference in the network can also be reduced. Additionally, most of the multichannel deployments propose to use multiple radios on each nodes [3, 4, 5, 6, 7]. By ensuring that the radios within a node are always operated on different, orthogonal channels, a node can effectively transmit and receive simultaneously.

Three popular channel and interface allocation strategies exist in the literature, namely common control channel approach [8] (where nodes decide on communication channel prior to a transmission using control messages on a common channel), static channel approach [9, 10] (in which the channels for all the radios of a node are fixed), and hybrid channel approach [11] (in which the channels for only a subset of radios are fixed apriori and that for the remaining radios are varied dynamically during communication). Among these three approaches, the hybrid multichannel protocol has been shown to be efficient in providing higher system throughput [11]. However, the hybrid channel allocation approach are not optimized for providing low delays for real time applications, such as VoIP. This is because, while a static channel allocation achieves network connectivity by a careful topology preserving channel selection [9], a hybrid channel allocation relies on the radios of a node to switch across channels to maintain network connectivity. Even with a fast hardware, the latencies associated with channel switching (as explained later) is prohibitive for delay sensitive application such as VoIP or interactive gaming, especially in the case of a multihop operation. Because no such channel switch delays exist in a static channel approach, such a scheme may be beneficial for delay sensitive applications. However, a purely static channel-based approach is not suitable for providing higher throughput values for non-delay sensitive applications. Moreover, a pure-static channel approach is not suitable in a network where the link characteristics keep varying that can make the network topology change with time (as in a mobile network). Therefore, we need a newer scheme that can exploit the advantages of both the static and the hybrid channel allocation schemes.

Routing real time applications over multichannel wireless networks has been handled in several different ways in literature. Most of the existing approaches concentrate on provisioning QoS in multichannel wireless networks [12, 13]. The authors in [13], for instance, propose a topology control and QoS routing approach with a goal

[*] This work was supported in part by NSF grant 06-27074

for providing bandwidth aware routing for real time flows. However, the approach requires significant topology information for its execution, and hence not wholly suitable for an unmanaged network. In [12], the authors provide a QoS-aware multichannel scheduling algorithm for providing higher priorities for VoIP packets, by which they are scheduled more often than non-real time packets. A similar approach for scheduling delay sensitive flows more often than non-delay sensitive flows is proposed in [14]. In [15], the authors propose a gateway controlled channel allocation scheme, where the channel allocation to the nodes are determined by the gateway based on the flows in the network. However, the scheme does not differentiate between real time and non-real time flows. In this paper, we propose a mechanism that can provide low delay routes for real time applications and high throughput routes for non-real time applications, which can complement any of the existing QoS mechanisms. The goal is to consider practical difficulties (such as hardware delays) that may exist in a network, which many of the existing QoS mechanisms overlook.

We propose a mechanism called SHORT that exploits the benefits of a static channel approach for providing lower delay paths for real time applications, while at the same time utilizes the flexibilities of a hybrid channel approach for providing higher throughputs for non-delay sensitive applications. According to this approach, we design a protocol that can, depending on the type of traffic being routed, control the channel allocation strategy of the nodes. More specifically, when routing a delay sensitive flow, the routing protocol, after determining the route to be taken for the flow, forces the nodes on the path to behave as in a static channel approach. In other words, the radios in the nodes are controlled in such a way to prevent them from switching across channels for the duration of the real time flow. A hybrid channel allocation scheme is used for routing non delay-sensitive flows. We modify the multichannel AODV routing protocol proposed in [11] for this purpose. Note that, while our protocol enables the nodes on a real time flow's path to behave as in the static channel mechanism, the actual path is not determined by our approach and is taken care by the multichannel routing protocol [11], discussed briefly in the Section 2 that is complemented by the hybrid channel allocation protocol (hence the name static-hybrid approach).

Using actual implementations on a multichannel mesh testbed, called Net-X [5] we show that the end-to-end delays of real time applications is significantly lower in SHORT when compared to a purely hybrid approach. Furthermore, we show that the throughput of non-delay sensitive applications is also not degraded.

2 Background

In this section, we provide a brief overview of the hybrid channel allocation protocol, called HMCP and the multichannel routing protocol [11] that are used in the testbed on which we carry out our experiments. In the discussion that follows, we assume that every node is equipped with two radios or interfaces (the terms interfaces and radios are used interchangeably in this paper and both mean a wireless radio).

2.1 Hybrid Multichannel Protocol Operation

The main challenge in a multichannel network implementation lies in ensuring that nodes operating on different channels can coordinate and communicate with other without much overhead. The hybrid multichannel protocol (HMCP) [11] ensures connectivity between nodes by allowing one of the two wireless interfaces to switch across channels as required. The other interface remains fixed on a channel as long as the channel is perceived to be good. We call the interface that may switch often across channels as the *switchable* interface and the interface that operates on a fixed channel as the *fixed* interface. Only the fixed interface is used for data reception. However, a data transmission can be from any of the two interfaces, fixed or switchable; this depends on the channel of the fixed interface on the neighboring node to which a multi-hop flow is directed. In general, if a neighboring node is operating on the same fixed channel as the current node, then the transmission can be through the fixed interface, otherwise the switchable interface is used for transmission after switching its channel to the fixed channel of the neighboring node. Thus, a node can potentially transmit and receive simultaneously, if the channels on which they transmit and receive are different. The necessary control messages that are exchanged to communicate the channel information between the nodes is discussed later in this section. Once a node switches to a channel, it stays on that channel for a pre-determined amount of time before switching to the next channel. The amount of time spent by the switchable interface may vary depending on the availability of packets to be sent on that channel. Because the channel on which a switchable interface operates depends on the channel allocated to the fixed interface of a neighboring node, it is clear that we need to allocate channels only to the fixed interface of a node. Figure 1 shows an example of our protocol operation for a bidirectional data transmission from node A to node C, with node B as an intermediate node. (Solid lines indicate transmission form A to C and dotted lines indicate the transmission from C to A. The switchable radio in B switches between the two directions.)

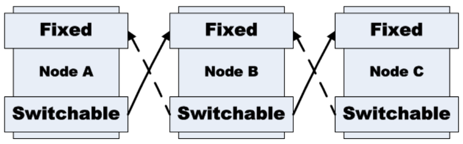

Fig. 1. Example multichannel protocol operation

The HMCP protocol operation requires that every node be aware of the (fixed) channels on which their neighboring nodes are listening. In other words, two neighboring nodes cannot communicate with each other even if one of them is not aware of the channel of the other. The nodes are made aware of the neighbor channels by the exchange of a broadcast `hello` message that contains the channel information. Any broadcast message sent by a node is transmitted on all the channels so that all of a node's neighbor that may be listening on any of the channels may receive the broadcast message. To help in load-balancing among the channels that are used within a neighborhood, the `hello` messages are propagated over two-hops. This allows every node to be aware of the channel information of all the neighbors that are up to two hops away.

The HMCP protocol also defines a channel allocation mechanism for allocating channels to the fixed interface. Briefly, the channel allocation algorithm works by using the two hop channel information exchanged using the broadcast `hello` messages for choosing a channel that is used by the least number of nodes in its (two hop) neighborhood. This helps in fairly balancing the number of nodes that are on each of the channels. Due to space restrictions we skip the details of the channel allocation mechanism. However, interested readers can refer to [5] for more information on the channel allocation algorithm.

2.2 Multichannel Routing Protocol

The routing mechanism used currently in our testbed is an AODV protocol, modified for multichannel operation. The modifications to the original AODV protocol include incorporating a mechanism for finding a channel diverse route, avoiding bottlenecks, and reducing the overall expected transmission time in addition to reducing the number of hops. More specifically, to utilize the benefit of using multiple channels, it is necessary to make sure that a flow experiences minimum intra-flow interference (interference due to transmissions of the same flow on adjacent hops). This requires that the route taken by the flows is such that the adjacent hops are on different channels as much as possible. Furthermore, it is preferable to avoid routing multiple flows through a single node, as this may result in the node requiring to switch its transmission channel frequently for routing the flows, which may possibly be targeted at neighbors on different channels. These requirements are incorporated in the form of a routing metric, called the MCR metric [5], as the traditional routing metric based on hop count is not suitable. The MCR metric, in brief, uses the statistics of channel usage from the interface drivers and uses it to calculate the cost for switching the channels for routing a flow. Additionally, the cost of a link per channel is estimated using the popular ETT metric [16] on every channel, which when coupled with the switching costs and summed up over the entire path results in the MCR routing metric. If $SC(c_i)$ is the channel switching cost of channel c_i used in the i^{th} hop of transmission, and ETT_i is the estimated transmission time in the i^{th} hop, then the MCR metric is given by,

$$\text{MCR} = (1-\beta) * \sum_{i=1}^{h} (ETT_i + SC(c_i)) + \beta * \max_{1 \le j \le c} X_j$$

where, β is a weight between 0 and 1, h is the number of hops on the path, and c is the total number of channels. X_j is the total ETT cost on channel j and is given by,

$$X_j = \sum_{\forall i \text{ such that } c_i = j} ETT_i.$$

The ETT of a link is given by, $ETT = ETX * \frac{S}{B}$, where ETX is the expected number of transmission attempts (including re-transmissions) required to transmit a packet, S is the average packet size and B is the data rate of the link. The expected number of transmissions is estimated based on the loss in the link.

The multichannel protocol also incorporates few other modifications. For instance, when a routing entry is created for a node, it is also necessary now to indicate the channel and the actual interface to use for reaching the next hop. The multichannel routing protocol incorporates the appropriate mechanism for creating the route entries. Furthermore, optimizations such as route caching, available in the original AODV protocol, is not performed as the channel allocations and the corresponding costs may change frequently, which can be estimated accurately only at the destination. Finally, the multichannel routing protocol incorporates a procedure called “Route Refresh”, by which a source node initiates a route discovery periodically (currently every 30 seconds in our testbed) for learning routes with better costs or for updating the costs of the current route.

3 Problem Statement

A pure-hybrid channel allocation approach (such as HMCP [11]) is optimized for providing higher system throughputs for non-delay sensitive applications. However, a main drawback with the hybrid channel allocation approach is the channel switching delays associated with the wireless radio hardware and software. For instance, the channel switching delay currently in our hardware, T_s is 5 ms. This includes several components such as, delays due to stopping interrupt service routines of the driver, tuning to the new frequency, re-starting the interrupt service routines and sensing the medium. To compensate for the higher switching delays, it is advisable to spend at least a minimum amount of time in a channel, before switching to another channel for amortizing the switching costs. Additionally, consider a scenario where there are multiple packets to be sent by a node, each on a different channel. In this case, while sufficient time has to be spent transmitting packets on the current channel before switching to the next channel, there has to be a limit on the time spent on any single channel. In the network used for our experiments, the minimum time spent on a channel, T_{min} is 20 ms, and the maximum time spent, T_{max} on a channel before switching to another channel that has packets waiting to be spent is 60 ms. The relevance of these parameters and the procedure used for choosing these values are discussed in more detail in [17]. Thus, the channel switching delay, T_s along with T_{min} and T_{max} together may add to the overall transmission time of a packet.

To illustrate more on the switching delays, we discuss the following simple experiment.

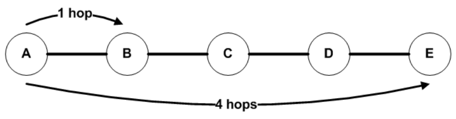

Fig. 2. Topology used for ping experiments

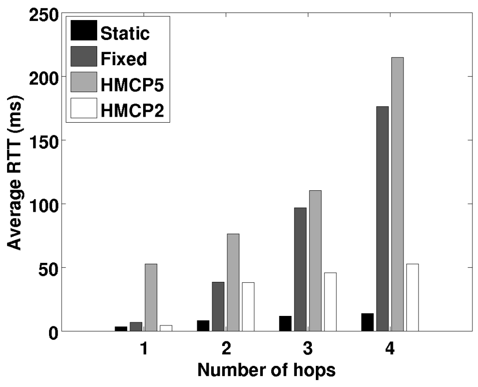

Fig. 3. Results for pinging the nodes in flooding mode

3.1 Ping Experiment

In this experiment, we use up to five wireless nodes that are placed across different offices in our lab and arranged linearly as shown in the Figure 2, each of which are one hop apart from their neighbors, and we initiate one hop, two hop, three hop, and four hop pings in flooding mode with 1500 byte packets. (A node is said to be one hop away from another node if they can have a direct communication between them. If two nodes require k one hop communications between them, through other intermediate nodes, then they are said to be k hops away from each other.) We plot in Figure 3, the resulting average round trip time (RTT) returned by ping when all the nodes use the same fixed channel (labeled as ‘Fixed’ in the plot), when the nodes are assigned channels using a static channel allocation (labeled as ‘Static’), and when the hybrid multichannel protocol with five channels (labeled as ‘HMCP5’) and two channels (labeled as ‘HMCP2’) is used for allocation. Note that in the case of HMCP2, the switchable radio do not switch channels as they always operate on only channel (the other channel is allocated to the fixed radio). In the case of ‘Fixed’, the switchable radios are free to switch across the remaining channels.

We can readily observe from the plot that the average RTT in the case of HMCP5 is significantly higher than the other channel allocations. Furthermore, we observe that the RTTs become worse as the number of hops increase. Finally, we also observe that the RTTs in the case of HMCP2 is much lower than HMCP5 and the same fixed channel allocations, though the actual values are higher than a static allocation. The reason for the increased RTTs in the case of HMCP5 is because of the following factors:

1. A transmission from one node to another that are on different fixed channels requires a channel switching. This can take place at every single hop of the path taken by the flow.

2. Because a periodic broadcast message, such as `hello` or a route refresh has to be sent on every channel, the associated switching delay adds up, at every hop, to the end-to-end delay.

Thus, by assuming a channel switching delay of 5 ms, and by assuming that only a T_{min} amount of time is spent on each of the channel and observing the fact that a message broadcast on the fixed interface do not incur any channel switching delay, a message broadcast on five channels incur a delay of $((5-1)\times 5+(5-1)\times 20 = 100ms)$ and that broadcast on 2 channels incur $((2-1)\times 5+(2-1)\times 20 = 25ms)$. Thus, the broadcast messages alone can cause round trip delays of up to $200ms$ and $50ms$, respectively. This is the reason for HMCP2 to have a lower delay when compared to HMCP5. The reason for higher RTTs in the case of 'Fixed' channel allocation is due to two reasons. The first reason is that the adjacent hops of a flow has to contend for channel access as they are both transmitted on the same channel. The second reason is due to hardware restrictions. Specifically, the wireless driver can schedule a transmission from only one of the two radios at a time. Consequently, a packet queued up on a fixed radio has to share its transmission opportunities with that in the switchable radio, resulting in a higher RTT.

The resulting delays, mainly in the case of HMCP5, are prohibitive for real time, delay sensitive applications such as VoIP or interactive gaming, and therefore alternate mechanisms has to be formulated for routing such applications. However, we should also ensure sufficient throughput for non-delay sensitive applications that may co-exist in the network. This motivates a routing approach that can improve both the delay and throughput performance depending on the type of application. From Figure 3, we see that a static channel allocation may be advantageous for real time applications, as it results in the least RTTs among the four mechanisms compared. Our proposed protocol exploits the advantages of this allocation. In this paper, we assume a dense network scenario that has a predominantly non-delay sensitive traffic with fewer delay sensitive applications. In fact, this mimics a real network scenario, as most of the flows in the present day internet are HTTP or FTP-type best effort traffic.

4 Proposed Approach

Motivated by our initial ping experiments, we develop a new routing strategy, called SHORT for controlling the wireless radios and the underlying channel allocation mechanism. The idea is to make the wireless radios behave as in a static channel allocation mechanism for real time applications and to follow the hybrid channel allocation mechanism for non-real time applications. Accordingly, the nodes in the network operate on one of two modes, namely normal mode and static mode. The normal mode of operation is exactly as explained in Section 2.1 and shown in Figure 1, wherein only the 'fixed' radio is used for receiving data and the 'switchable' radio is used only for transmitting data, after switching to the corresponding channel. This mode of operation is used for non-delay sensitive traffic. For delay sensitive flows, the static mode of operation is used. In this mode, the 'switchable' interface is not allowed to switch channels for the duration of the delay sensitive flow. Rather, after the route for the flow is determined, it remains fixed on the channel of the previous hop's[1] fixed interface. Furthermore, both the fixed and the switchable radios are allowed to receive and transmit. In other words, the switchable interface also behaves like a 'fixed' interface for the duration of the delay sensitive flow. Note that only those nodes that lie in the path of a delay sensitive flow operate in static mode. The remaining nodes in the network continue to behave as in the normal mode. Furthermore, the nodes that are in static mode revert back to normal mode of operation once the delay sensitive flow ends. The associated protocol steps for getting back to normal mode is trivial and not discussed in this paper. While the channel allocations in our protocol are based on HMCP to simplify implementation, any existing dynamic channel allocation can be used, in general.

We wish to explain this concept more clearly using the illustration in Figure 4. The figure shows a traffic flow from node A to C via node B. Let the channels allocated to the fixed radios of the nodes A, B. and C be labeled 1, 2, and 3, respectively. Accordingly, during the static mode of operation, the switchable radio of node C is fixed to channel 2, which is the fixed channel of node B. Similarly, the switchable radio of node B is fixed to the channel

[1] The terms 'previous hop' and 'next hop' imply the appropriate nodes in the path as seen by a node in the 'source to destination' direction of the flow.

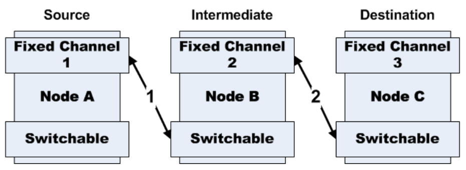

Fig. 4. SHORT protocol operation

1, which is the fixed channel of node A. Thus, traffic from A to B flows on channel 1, and that from B to C flows on channel 2. Moreover, the switchable radios on nodes B and C receive the traffic on these channels transmitted by the fixed radios of nodes A and B, respectively. Observe that any traffic from C to A can be routed using the same configuration, except that the switchable radios will be sending traffic that will now be received by the fixed interface of the nodes B and A. Thus, this setting enables a bi-directional flow without requiring any channel switching. We would like to point out that the switchable interface of node A is not required to be fixed on any channel in this example, and is free to switch across channels as in the normal mode. Because in this example, the nodes B and C behave as in a static channel allocation (both the radios are non-switchable and every node on the path shares a channel with the adjacent hop nodes), we call this as static mode.

In the static mode of operation, a node does not send a broadcast message on all the channels (unlike the normal mode of operation, see Section 2.1). Instead, it simply forwards them on the channels to which the two radios are fixed. Note that this may result in few nodes not being aware of the channel used by their neighboring nodes. Because we require in our protocol that two nodes involved in a direct communication be aware of each other channels (as otherwise the nodes cannot decide on which channel to transmit), this may result in a node losing connectivity with the nodes that are on a channel different from those on which the broadcast messages are sent. When several such nodes lose connectivity with each other, this can eventually result in a network partition. To avoid such a scenario, we propose a channel re-selection mechanism that works in tandem with the routing protocol. More details of the channel re-selection mechanism is explained later in this section.

4.1 SHORT Protocol Operation

We now discuss the details of the SHORT protocol. We assume that the information whether the flow being routed is delay-sensitive or not is available at the routing layer of the source node. Such an information can be passed on from the upper layers by, for instance by setting the ToS (type of service) field in the IP header. The actual mechanism on how this information is passed on from the application to the routing layer is beyond the scope of this paper. We just present the protocol sequence executed for a delay sensitive flow. The sequence of procedures carried out for a non-delay sensitive flow is as done in the multichannel routing protocol, explained in [18] and is not reproduced here. The protocol mechanisms described for delay sensitive flows, however, is a modification of the multichannel routing protocol and to avoid duplication of work, we present only the relevant modifications to the multichannel routing protocol.

Once the source node determines that it is a delay sensitive flow, the following is performed:

1. The source node checks if a route is already available for the destination. If not, it initiates a route request message (RREQ) along with a special flag, `isRealTime` to indicate that the request is for a real time flow and broadcasts it on all channels.

2. Any intermediate node, that is not the destination, simply re-broadcasts the RREQ message on all channels.

3. The destination, upon receiving the RREQ, creates a route response (RREP) message and unicasts the RREP along with the `isRealTime` flag (copied from RREQ) to the node from which the corresponding RREQ was received. Additionally, it takes the following actions only if the channel on which the RREP is unicast (which is the fixed channel of the previous hop node in this path) is different from its own fixed channel:

a. The node sends a broadcast `hello` message as described in Section 2.1 on all the channels. However, in this case, the node includes the flag `isRealTime` along with two channel information. One is the fixed channel that it has been operating on, and the other is the channel over which the RREP is unicast. (Note that the original `hello` message described in Section 2.1 contains only the fixed channel information.) The cost associated with this broadcast is one time and shall be considered as part of the route setup cost, which does not affect the delay experienced by the delay sensitive packets.

b. The node fixes its switchable interface on the channel over which the RREP message is unicast (which is the fixed channel of the previous hop node in this path). The routing entry created for the previous hop node is informed to use the switchable interface in this (reverse) direction.

c. The switchable interface is also informed to start receiving packets on this channel.

4. Any intermediate, upon receiving the RREP along with the `isRealTime` flag, also forwards the RREP message to the node from which it received the corresponding RREQ message. Furthermore, the intermediate node, in addition to performing the set of operations described in Step (3) when the RREP is unicast on a channel different from its own fixed channel, also performs the following:

d. The node creates a routing entry for the next hop node and is informed to use the fixed interface in this direction of flow (forward direction).

5. The source node, upon receiving the RREP starts sending the packets, after creating the routing entry for the next hop node through its fixed interface.

Once the radios of the corresponding nodes in the real time flows path are fixed based on the above steps, any transmission by these nodes (including broadcasts) are restricted to the two fixed channels. Observe that any non-real time flow that has been existing in the chosen real time path prior to the arrival of the real time flow may be affected because of this protocol. In particular, an existing non-real time flow may be dropped during the above process as the radios on the corresponding path will no longer be allowed to switch across channels. We handle this situation by initiating a RERR message, which gets forwarded to the source. The source can then re-initiate a new RREQ message to find a new route. Because of the channel re-selection mechanism (described in the next sub-section), finding a new route will not be difficult and we did not see a significant throughput loss, as a result, during our experimentation.

4.2 Channel Re-selection Mechanism

The channel re-selection mechanism is introduced to maintain network connectivity in spite of nodes in static mode restricting their broadcast to only the two channels that their interfaces are fixed on. The channel re-selection mechanism is only executed by those nodes that lie adjacent to the path chosen for the real time applications and are in the normal mode. For this purpose, the nodes make use of the broadcast `hello` message with a `isRealTime` flag broadcast by a node in the path of a real time flow before switching to the static mode (see Section 4.1 step 3a.). Upon receiving the broadcast message with a `isRealTime` flag, the nodes performing channel re-selection perform the following steps:

1. The node first checks if both of the channels contained in the `hello` message is different from its fixed channel. If its fixed channel is same as one of the channels in the `hello` message, the node discards the message and takes no further steps.

2. If both the channels in the `hello` message are different from the node's fixed channel, then the node selects one of the two channels, chosen uniformly at random, as its new fixed channel.

3. If more than one `hello` message with a `isRealTime` flag is received (which may happen when a node is adjoining two nodes that lie in the path of a real time flow), then the node first tries to choose the channel that is common to a majority of the `hello` messages. Thus, the channel re-selection mechanism is designed to maintain connectivity with a majority of the nodes in the network. If none of the channel is common to the `hello` messages, then the node just selects one of the channels contained in the `hello` messages, uniformly at random, as its fixed channel.

When the majority of flows in the network are real time, the channel re-selection mechanism will tend to make the overall network behave as in a pure-static approach, while when the majority of flows in the network are non-real time, the network behaves as in a pure-hybrid approach, as required.

4.3 Implementation Specific Details

The architecture of our multichannel protocol along with the SHORT implementation is shown in Figure 5. The SHORT protocol consists of two main components, namely the SHORT controller or *C-SHORT* and the SHORT executor or *E-SHORT*. The *C-SHORT* is implemented in the user level and interacts with the multichannel routing protocol for creating routing entries compatible with the static mode of operation whenever a real time flow is to be routed. Furthermore, it is also responsible for setting the *isRealTime* flag when a new route discovery for a real time flow is initiated. Finally, *C-SHORT* indicates to the *E-SHORT* component, through a special IOCTL control message, whether to transition to static mode or revert back to normal mode. (IOCTL messages are used standardly

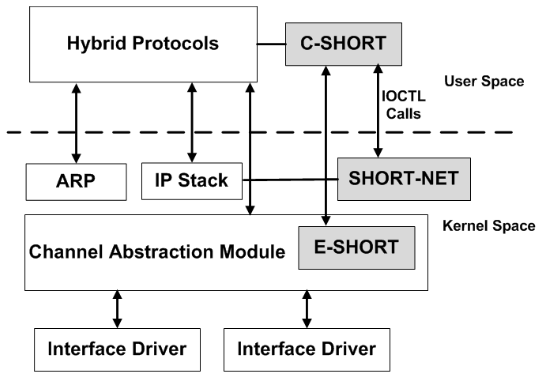

Fig. 5. System architecture with SHORT-specific components in gray

in linux for any interaction between the user space and kernel space code.) If the message is for transitioning to static mode, then the channel to which the switchable radio has to be fixed from now on is also specified.

The *E-SHORT* component, on the other hand, is implemented as a kernel module and resides as part of the linux 'bonding' module[2]. The *E-SHORT* component is responsible for fixing the switchable radio to the channel supplied by the *C-SHORT* component and for restoring the switchable radio back to normal mode, depending on the message from the *C-SHORT* component.

In addition to the two main components, SHORT protocol also consists of a smaller third component, called SHORT-NET, which interacts with the linux netfilter hooks for making the switchable interface behave like a fixed interface for real time flows. In normal mode, the netfilter hook is designed to drop any incoming packets on the switchable radio. The SHORT-NET overrides this and lets the switchable radio to accept the packets while in static mode. The relevant control messages are passed on from the *C-SHORT* as an IOCTL message.

5 Experimental Results

In this section, we present the experimental results to illustrate the performance benefits of the SHORT protocol. Before proceeding further, we first present an overview of our testbed and the associated hardware.

5.1 Testbed Overview

We use a multi-channel, multi-interface, and multi-hop wireless testbed called Net-X, developed by the Wireless Networking Group at the University of Illinois at Urbana-Champaign (UIUC). The testbed consists of 20+ Soekris net4521 boxes distributed across various offices on the fourth floor of the Coordinated Science Lab (CSL) in UIUC. Each of the testbed node has a 133 MHz microprocessor, a compact flash (CF) card slot, two PCMCIA slots, and one mini-PCI slot. We run Linux kernel 2.4.26-based operating system on each of these boards. For our experiments, we equip the test nodes with one mini-PCI and one PCMCIA wireless card. These wireless cards are based on Atheros chipsets and are driven by madwifi drivers. The cards operate in the IEEE 802.11a mode. The mini-PCI cards make use of a pair of external antennas, and the PCMCIA card has its own internal antenna for communication.

5.2 Experimental Methodology

Traffic Details: For evaluating our protocol, we used different traffic sources for generating real time and non-real time traffic. For real time traffic, we used a tool called D-ITG [19] for generating G.711 codec type VoIP packets for 50 seconds. The tool generates about 100 byte VoIP packets every 20 ms. The same D-ITG tools is used generating non-delay sensitive TCP and UDP type packets. The UDP flows are always generated at a rate of 6 Mbps and the packet sizes are fixed at 512 bytes. The size of the TCP packets on the other hand are uniformly distributed between 500 and 1000 bytes, and are generated at the rate of 1000 packets per second. Both UDP and TCP packets are generated for a duration of 50 seconds. Every wireless radio transmits at the maximum power and the physical rate of transmission are fixed at 6 Mbps. For all the experiments we use five orthogonal 802.11a channels, namely 36, 48, 64, 149, and 161 for allocation.

[2] The bonding module has been modified in our system to enable multi-radio operation and the details can be found in [5]

Protocols Compared: We compare the performance of our SHORT protocol with HMCP and two other protocols as described below:

Static channel allocation: For this case, we allocate channels to the radios using a centralized static channel allocation methodology. In other words, knowing the connectivity graph among the nodes, we allocated channels to the two radios in a node such that every node having an edge in the connectivity graph has at least one channel in common. The channel allocation technique is based on the scheme proposed in [9].

Fixed channel for real time traffic: This is a protocol similar, but simpler than SHORT. In this protocol, while generating a route discovery for a real time flow, the source node also includes its current fixed channel in the RREQ message. Every intermediate node re-broadcasts the RREQ message, as usual. However, while forwarding the RREP message the corresponding node changes its fixed channel to that of the source node (which is embedded in the RREP message). Thus, all the nodes in the path of a real time flow use their fixed interface for routing. The advantage of this scheme is that the switchable radios in the nodes need not be fixed and can remain switchable as in normal mode. As a result, unlike SHORT there will be no loss of connectivity. We call this as 'fixed' mode of operation. Figure 6 illustrates this protocol.

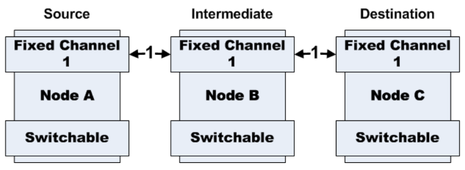

Fig. 6. Fixed mode operation

Performance Metrics: The D-ITG tool is capable of generating per flow statistics on the minimum, maximum, and average delays, average jitter, and throughput achieved. Because throughput is not of concern for real time flows, and delays are not important for non-real time flows, we measure the average and maximum delays, and jitters (which is the variance in time of arrivals of adjacent packets at the destination) for real time traffic and the throughput values for the non-real time traffic. However, due to space restrictions, we only present the average delay values for real time applications. The maximum delay and jitter values can be referred from [20].

Time Synchronization: For measuring delays it is important to have a common notion of clock between the traffic sources and destinations. However, the wireless nodes used have imperfect clocks and proper time synchronization is necessary for measuring time delay values between the sender and receiver. We therefore use `ntpdate` periodically on these nodes for synchronizing their time values. Because the nodes are not connected to the internet, we use a desktop computer as the ntp server and synchronize all the nodes relative to this server. We use the local wired LAN connectivity for time synchronization between the nodes and the desktop ntp server.

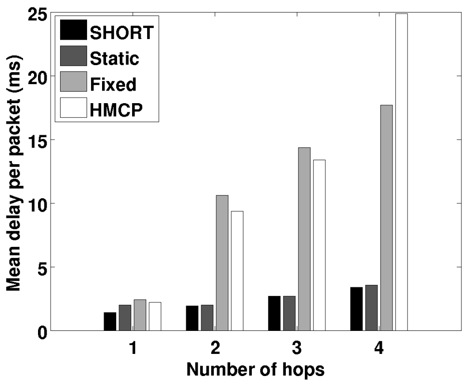

Fig. 7. Average delay for unidirectional flows

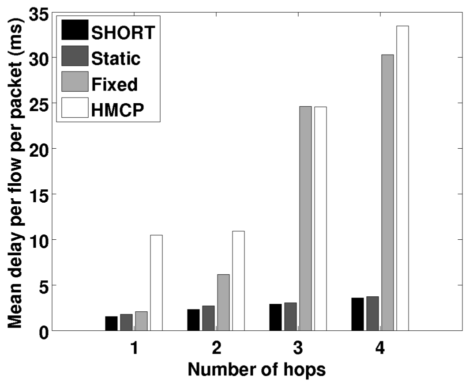

Fig. 8. Average delay for bidirectional flows

5.3 Performance Results

We now discuss in detail the experimental setup and the performance results.

Multihop Experiments: For each of the experiments in this section, we generate flows between a pair of nodes that are separated by one, two, three, and four hops away. Owing to the size of our network, we cannot realize a route that is farther than 4 hops. For each of the scenarios, we chose 10 different source and destination pairs, each of which are picked from different locations in the network, and are separated by different distances.

Unidirectional flows

For this experiment we generate a VoIP flow between a source and a destination that is located one hop away from the source. We then repeat this for destinations that are two hops, three hops, and four hops away from the source node. In each case, we measure the average delay experienced by the packets, and the results averaged over the 10 different source-destinations pairs are plotted in Figure 7. In all the figures presented in this section, the plots corresponding to the static channel allocation are labeled as 'Static' and those obtained for the case where we use the fixed channel for real time flows are labeled as 'Fixed'.

From Figure 7, we first observe that the average delays experienced by the VoIP packets in the case of SHORT and Static are always lower than 5 ms, irrespective of the number of hops. We also observe that the delays in the case of Fixed and HMCP allocations are much higher than SHORT or Static allocation, and the difference increases significantly as the number of hops increase. As mentioned in Section 3, the main reason for higher delays in the case of HMCP is the need to switch the channels at every hop along the multihop path. In the case of Fixed channel allocation, the delays are comparatively lower than HMCP owing to the fact that the fixed radios are used for transmitting the VoIP packets. However, the delays are still high when compared to SHORT or Static. One reason for this is that the fixed radio has to share its transmission opportunity with that of the packets in the switchable radio, as explained in Section 3. Though the average delay of about 38 ms in the case of HMCP for the 4 hop case is acceptable for VoIP packets, the rate at which the delay grows with the number of hops is significant and the delays may become unacceptable in the case of real multichannel deployments, where more than 4 hops may be common. Even in the case of 4 hops, we observed that there were packets that experience more than 200 ms delays (not shown here), which is, certainly unacceptable for VoIP.

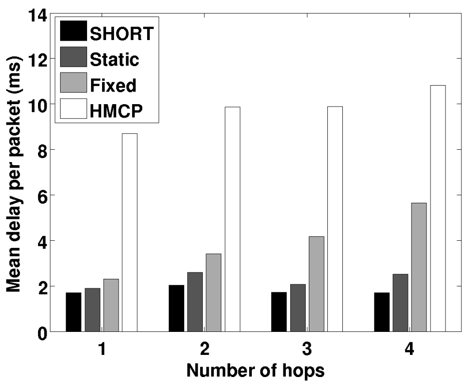

Fig. 9. Average delay of VoIP packets sent with a UDP flow

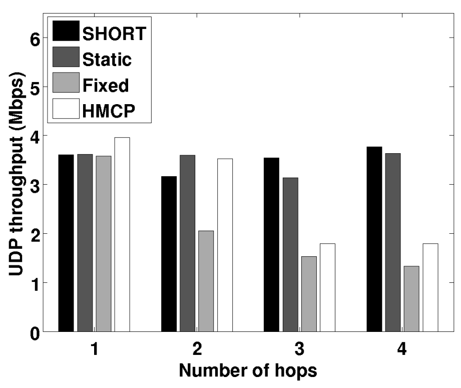

Fig. 10. Throughput of UDP flow went with a VoIP flow

Bidirectional flows

In this case, we generate two VoIP flows, one from a source to the destination and the other from the destination to the source. The delay values averaged over all the flows and over 10 different pairs of nodes, chosen from different location in the network for each scenario, are plotted in Figure 8. We first observe that the delays in the case of SHORT and Static mechanisms are similar to that in the unidirectional case. This is because, in the case of SHORT protocol, once a route is established between two nodes, the same route is used both for the forward and reverse traffic. The same is true in the case of Static mechanism. The delays in the case of HMCP is higher than that in the unidirectional case. This is because a significant time is spent by the switchable radio in switching between the forward and reverse traffic.

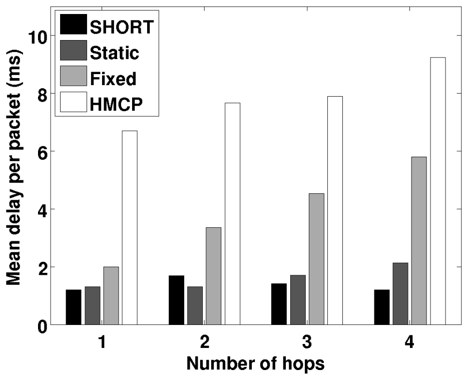

Fig. 11. Average delay of VoIP packets sent with a TCP flow

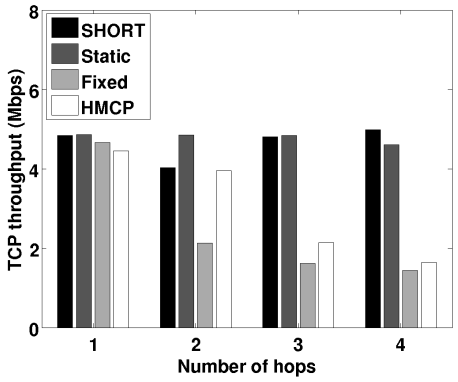

Fig. 12. Throughput of TCP flow went with a VoIP flow

VoIP with UDP and TCP (non-delay sensitive)

For this experiment, we first generate a VoIP flow along with a UDP flow, both from the same source and targeted at the same destination. Figure 9 shows the average delay experienced by the VoIP packets, and Figure 10 when the throughput achieved by the UDP packets, all averaged over 10 different source-destination pairs. Next we generate a VoIP flow along with a TCP flow as before, and the delay and throughput values of the VoIP and TCP packets, respectively are plotted in Figures 11 and 12. We observe from the plots that the throughputs for both UDP and TCP flows remain almost the same, irrespective of the number of hops, in the case of SHORT and Static protocols. However, we observe that the throughputs reduce with the number of hops in the case of Fixed and HMCP. In the case of TCP flows, this is because of the increased RTTs between the source and destination, which in turn affects the packet generation rate at the source. In the case of UDP, this is due to loss of packets during channel switching. Furthermore, in the case of fixed mode, adjacent hops of the same flow contend for transmission as they are on the same channel. This, in turn affects the throughput achieved.

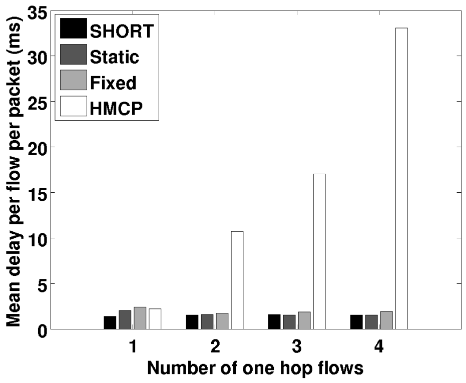

Fig. 13. Average delays for multiple one hop flows from a node

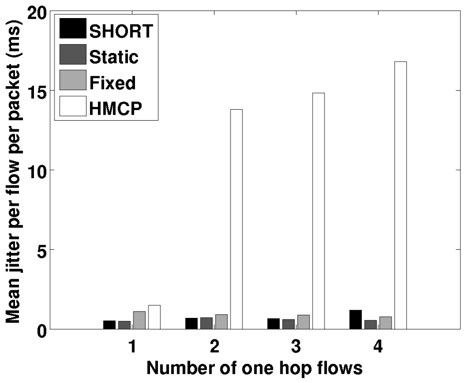

Fig. 14. Average jitters for multiple one hop flows from a node

Single hop experiments: We now evaluate the capability of the protocols in supporting multiple flows from the same source node, as such a scenario may usually involve several channel switches when each flow is sent on a different channel. For this purpose, we choose a source node and four other nodes that are within one hop from each other and generate multiple VoIP flows (varied from one to four) between them. Once gain, we choose 10 different sets of nodes situated at different locations in our network and present the average values. The average delay values per flow are plotted in Figures 13. For this case, we also present the average jitter values in Figure 14. We observe from the delay plots that the average and maximum delay values do not vary much with number of flows in the case of SHORT, Static, and Fixed mechanisms, while it increases significantly for HMCP. (The improper variations in throughputs with the number of hops in the case of SHORT and Static are due to averaging.) This shows that HMCP is not capable of multiple real time flows all from the same source, as it requires significant channel switching. From the jitter values, we observe that HMCP performs poorly while handling multiple flows. Higher jitter values mean that the amount of jitter buffer at the receiving side should also be higher, so as to prevent packet losses. The jitter performance for SHORT and Static are fairly stable irrespective of the number of flows and allows for better codec design.

We also performed some measurements to evaluate if the number of hops taken by a non-delay sensitive application is higher than that taken by HMCP. However, we found that there were no significant throughput losses because of SHORT protocol. We do not provide those results in this paper due to space restrictions. Moreover, our main goal in this paper is to demonstrate the delay performance of SHORT.

6 Conclusion

In this paper, we proposed SHORT, a routing approach that exploits the benefits of both static and hybrid channel allocation strategies. We have implemented the protocol on a real multichannel testbed and using extensive experimental data we have demonstrated the performance benefits of the SHORT protocol over a hybrid channel allocation protocol, called HMCP. All our experimental results illustrate the abilities of SHORT protocol in providing low delay multihop paths for real time traffic, while not affecting the throughputs of non-real time traffic. Our results show that the performance of SHORT protocol is comparable to that of a static channel allocation method. As a future work, we wish to demonstrate the benefits of SHORT using real voice traffic, which has not been possible right now due to hardware restrictions.

References

[1] A. Raniwala and T. c. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multichannel wireless mesh network," in *IEEE Infocom*, 2005.

[2] R. Maheshwari, H. Gupta, and S. Das, "Multichannel mac protocols for wireless networks," in *IEEE SECON*, 2006.

[3] P. Bahl, A. Adya, J. Padhye, and A. Wolman, "Reconsidering wireless systems with multiple radios," in *ACM SIGCOMM*, 2004.

[4] K. Ramachandran, I. Sheriff, E. Belding, and K. Almeroth, "A multi-radio 802.11 mesh network architecture," *MONET journal*, 2008.

[5] P. Kyasanur, C. Chereddi, and N. Vaidya. (2006) Net-x: System extensions for supporting multiple channels, multiple interfaces, and other interface capabilities. [Online]. Available: http://www.crhc.uiuc.edu/wireless/groupPubs.html.

[6] K. Ramachandran, K. Almeroth, and E.M.B.-Royer, "A novel framework for the management of large-scale wireless network testbeds," in *WinMee Workshop*, 2005.

[7] V. Navda, A. Kashyap, and S. Das, "Design and evaluation of imesh: an infrastructure-mode wireless mesh network," in *IEEE WoWMoM Symposium*, 2005.

[8] J. So and N. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *ACM MobiHoc*, 2004.

[9] A. Das, R. Vijayakumar, and S. Roy, "Static channel assignment in multi-radio multi-channel 802.11 wireless mesh networks: Issues, metrics and algorithms," in *IEEE Globecom*, 2006.

[10] M. Marina and S. Das, "A topology control approach to using directional antennas in wireless mesh networks," in *Broadnets*, 2005.

[11] P. Kyasanur and N. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *ACM MC2R*, 2006.

[12] M. Castro, P. Dely, A. Kassler, and N. Vaidya, "Qos-aware channel scheduling for multi-radio/multi-channel wireless mesh networks," in *WinTech*, 2009.

[13] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and qos routing in multi-channel wireless mesh networks," in *ACM MobiHoc*, 2005.

[14] T.-Y. Shen, "Experiments on a multichannel multi-interface wireless network," in *M.S. Thesis, UIUC*, 2008.

[15] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," in *ACM Sigcomm*, 2009.

[16] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless networks," in *ACM MobiCom*, 2004.

[17] C. Chereddi, "System architecture for multichannel multi-interface wireless networks," in *M.S. Thesis, UIUC*, 2006.

[18] P. Kyasanur, "Multichannel wireless networks: Capacity and protocols," in *Ph.D. Dissertation, UIUC*, 2006.

[19] A. Botta, A. Dainotti, and A. Pescape, "Multi-protocol and multi-platform traffic generation and measurement," in *IEEE Infocom*, 2007.

[20] V. Raman and N. Vaidya. (2009) A static-hybrid approach for providing low delay routing for real time applications. [Online]. Available: http://www.crhc.uiuc.edu/wireless/groupPubs.html.