Channel Assignment in Multi-channel Wireless Ad hoc Networks

By

Yi Qu
B.Eng. Xidian University

A thesis submitted to the Faulty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada

May 2007

Your file Votre référence
ISBN: 978-0-494-27000-4
Our file Notre référence
ISBN: 978-0-494-27000-4

Abstract

Many research efforts on wireless ad hoc networks are based on a single channel. Multi-channel is a promising approach to increasing network capacity. This thesis proposed two network layer channel assignment solutions – proactive Receiver-based Channel Assignment (RCA) and on-demand Negotiation-based Channel Assignment (NCA). A multi-interface node structure is designed to support RCA and NCA. Each wireless node has a dedicated control interface and dual data interfaces. Optimized Link State Routing protocol is extended to support the proposed solutions, but the routing algorithm is decoupled with the channel control. RCA assigns a channel to each node based on node topology only. NCA considers both topology and traffic pattern information. Based on simulation with ns2 [NS2], Both RCA and NCA have better performance than that of the single channel approach, and NCA performs better and requires less channel resources than RCA.

Acknowledgements

Many thanks first go to my supervisors, Professor Chung-Horng Lung and Dr. Anand Srinivasan, for their valuable guidance, thoughtful advice and continuous support. Secondly, I would like to acknowledge the support of Dr. Kalai Kalaichelvan and Dr. Pramod Dhakal from EION Inc., and also CITO (a part of OCE Inc.), Precarn and EION Inc. for partially funding this research. Finally, I am very grateful to my friends and my family for their direct or indirect contribution to this thesis. Special thanks go to Cinderella H. Wang for her encouragement.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AODV	Ad hoc On-Demand Distance Vector
ARP	Address Resolution Process
CBR	Constraint-Based Routing
CREP	Channel Reply message
CREQ	Channel Request message
CSMA/CA	Carrier Sense Multi-Access and Collision Avoidance
CTS	Clear To Send
CUN	Channel Usage Notification message
HTP	Hidden Terminal Problem
LL	Logical Link
MAC	Media Access Control
MANET	Mobile Ad hoc Networks
MMAC	Multi-channel MAC
NCA	Negotiation-based Channel Assignment
OLSR	Optimized Link State Routing
OSI	Open Systems Interconnection
PDR	Packet Delivery Rate
RCA	Receiver-based Channel Assignment
RTS	Request To Send
SC	Single data Channel
SI	Single Interface
SNR	Signal Noise Ratio
SSCH	Slotted Seeded Channel Hopping
TCP	Transmission Control Protocol
TCL	Tool Command Language
UDP	User Datagram Protocol

Chapter 1 Introduction

This chapter introduces the problem with a single channel and the challenges in designing a multi-channel wireless ad hoc network. After that, there will be a discussion on the motivation behind the proposal of a network layer multi-channel solution with a dedicated control interface. The contributions and organization of the thesis will then be described.

1.1 Multi-channel wireless ad hoc networks

In traditional wireless mobile ad hoc networks (MANETs), all nodes communicate on a common channel. Only one node can transmit or receive at the same time in an interference area. Hence, performance can be significantly reduced when nodes are trying to communicate with each other simultaneously. The multi-channel approach is prominent in increasing the throughput for MANETs. Multiple communications can take place concurrently on different channels without interfering with each other. However, there are two challenging problems in designing a multi-channel solution for wireless ad hoc networks—*channel assignment* and *channel switching*. The challenge is due to the deafness problem [So04b] in multi-channel networks; i.e., nodes on different channels cannot communicate with each other.

Channel assignment determines which channel to use for a transmission. Three main channel assignment schemes have been used in previous work—flow-based [So04b], node-based [Kya05, Lee05, Lee06] and negotiation-based [Wu00, So04a, McG06]. In the flow-based channel assignment scheme, all nodes along a flow transmit and receive on a common channel assigned to the flow. In receiver-based node channel assignment, a node

transmits on the channel assigned to the next hop node. Node-based channel assignment is more widely used because it can be decoupled with the routing protocol and thus is simpler than the flow-based scheme. On the other hand, the node-based channel assignment scheme faces the challenge of the *channel switching* problem. Nodes along the flow are assigned orthogonal channels with neighbor nodes. Traditional wireless nodes are normally equipped with only one half-duplex transceiver. The nodes need to switch between receiving (Rx) and transmitting (Tx) channels to relay packets, while the transmitter and receiver must coordinate to switch to the same channel during communication. The negotiation-based channel assignment scheme solves the channel assignment and channel *switching coordination* in the negotiation process. This approach is currently only used in MAC layer solutions.

Multi-channel wireless ad hoc networks have been researched extensively and many solutions have been proposed. A multi-channel solution can be categorized by the channel assignment scheme it takes, by the Open Systems Interconnection [OSI] layer on which it is implemented, and whether a multi-interface is used on each mobile node. This thesis proposes a new solution with a dedicated control interface and negotiation-based channel assignment at the network layer. This innovative multi-channel proposal can achieve significant performance improvement.

1.2 Motivation

Because of the contention on the common communication channel, wireless ad hoc networks have low network capacity [Gup00], and the multi-channel approach has thus attracted a lot attention for improving the wireless ad hoc networks' capacity.

Some multi-channel solutions [So04a, So04b, Bah04b, Mah06] are designed with

the constraint of a single interface. Some MAC layer solutions, Multi-channel MAC (MMAC) [So04a] and Local Coordination-based Multi-channel MAC (LCM-MAC) [Mah06], negotiate data communication channels in the periodic synchronized time slot on a common channel. Time synchronization in wireless networks is a problem. A Link Layer solution, Slotted Seeded Channel Hopping (SSCH) [Bah04b], uses a channel-hopping mechanism by which nodes communicate on an overlapped hopping schedule. The channel hopping/switching delay penalty is not negligible [Jeo05]. A routing protocol coupled channel assignment Multi-Channel Routing Protocol (MCRP) [So04b] realizes routing and channel control communication on a single interface by repeating broadcast control messages on all channels. A multi-channel solution is difficult to realize on a single interface with low overhead. With the cheaper transceivers available now, a multi-interface is common for a wireless node. This thesis will investigate the multi-channel multi-interface approach.

The multi-interface approach has been researched in depth as well for multi-channel wireless ad hoc networks. One interface is normally dedicated as a control interface working on a common channel. This solution eliminates the *deafness problem* [So04b] with a single interface on multi-channel networks and provides an efficient control mechanism. The dedicated control interface was proposed in MAC layer solutions [Wu00, McG06] and network layer solutions [Lee05, Lee06, Qu06]. IEEE 802.11 [802.11] has become the de facto standard. A new MAC protocol is not easily deployed. This thesis uses a dedicated control interface to realize multi-channel control on the network layer. Part of the work (Received-based Channel Assignment) is similar to OLSR-MC [Lee06], but was studied independently [Qu06].

Lee et al. proposed the network layer multi-channel solutions DSDV-MC [Lee05] and OLSR-MC [Lee06] with a dedicated control interface. They use the Receiver-based Channel Assignment (RCA) scheme (Section 2.2.2), which requires a lot of channels in proactive channel assignment and cannot guarantee interference-free channel assignment. Another network layer proposal, MCRP [So04b], takes the approach of a flow-based channel assignment scheme (section2.2.1), which is complicated because of coupling with the routing protocol. The Negotiation-based Channel Assignment (NCA) scheme (Section 2.2.3) requires much fewer channel resources because only nodes along the traffic flow are assigned channels. Furthermore, NCA can achieve interference-free channel assignment in some scenarios where RCA fails to do so. The NCA scheme is commonly used in MAC layer solutions [So04a, Wu00, McG06] for per-packet channel negotiation. NCA at the network layer can achieve optimum channel assignment and be deployed on the legacy MAC protocol. To the best of our knowledge, this approach has not been studied before.

Channel assignment proposals in the existing literature consider only interference (section 2.1.2) in the transmission range, but the interference range is generally known as being two times the transmission range [Pad05]. The correct interference model should be considered in this thesis.

In order to overcome the problems demonstrated by the previous research mentioned above, this thesis aims to design a multi-channel wireless ad hoc network solution with the following objectives:

- Simple and low overhead channel control
- No impact on IEEE 802.11, the legacy MAC and Logical Link (LL) protocol

- Decoupled and compatible with standard MANETs routing protocols
- Interference-free multi-channel assignment scheme
- Efficient in using channel resources

1.3 Contributions

This thesis proposes two network layer multi-channel solutions – RCA and NCA. Each wireless node is equipped with two data interfaces and a dedicated control interface. The proposed multi-channel solutions improve performance significantly and a five-fold throughput improvement is achieved by NCA in simulation with ns-2 [NS2]. The main contributions of this thesis include:

- Propose a dedicated control interface and dual data interfaces to facilitate simple and efficient multi-channel control in a wireless ad hoc network
- Improve the previous network layer Receiver-based Channel Assignment scheme with a more accurate interference model
- Design a new network layer Negotiation-based Channel Assignment scheme that can achieve almost interference-free channel assignment with a small number of channels
- Extended OLSR routing protocol to support the decoupled multi-channel control without introducing extra overhead
- Improve TCP performance by sending back transport layer acknowledgements (ACKs) on the dedicated control interface.

Part of this thesis appeared in the following publication:

- Y. Qu, C.-H. Lung, and A. Srinivasan, "Multi-channel OLSR with Dedicated

Control Interface", Proc. of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), August 2006

Another paper about the NCA solution was submitted to Globecom'07:

- Y. Qu, C.-H. Lung, and A. Srinivasan, "Network Layer Negotiation-based Channel Assignment in Multi-channel Wireless Networks", submitted to Globecom'07, Mar, 2007

1.4 Thesis organization

This thesis is organized as follows. Chapter 2 presents some multi-channel background and related work; Chapter 3 proposes a network layer multi-channel solution using a dedicated control interface, and discusses related channel assignment schemes; Chapter 4 features the simulation and analysis; and conclusions and future work are discussed in the last chapter.

Chapter 2 Background and Related Work

First, this chapter will present background knowledge about interference in wireless communication. Next, previous related work on multi-channel networks will be reviewed from three perspectives.

- Channel assignment schemes: the flow-based, node-based, and negotiation-based channel assignment scheme
- Proposals implemented on different OSI layers: MAC layer, Logical Link (LL) layer or network layer.
- Number of interfaces: single interface or multi-interface. For multi-interface node structure, routing and channel control packets may be transmitted on all interfaces or on a dedicated interface.

Classification		So04b	Kya05	Qu06	Lee06	Bah04b	So04a	Wu00	McG06
Channel assignment scheme	Flow-based	√							
	Node-based		√	√	√	n/a			
	Negotiation-based						√	√	√
Number of interfaces	Single interface	√				√	√		
	Multi-interface		√	√	√			√	√
	Dedicated control interface			√	√			√	√
OSI layers	Network layer	√	√	√	√				
	LL layer					√			
	MAC layer						√	√	√

Table 2-1. Overview of Multi-channel Related Work

2.1 Wireless interference

This section provides background information about wireless interference. Section 2.1.1 describes the Carrier Sense Multi-Access and Collision Avoidance (CSMA/CA) mechanism of IEEE 802.11 [802.11]. Based on an understanding of the 802.11 media access mechanism, Section 2.1.2 presents an accurate model for interference area and interference range. Section 2.1.3 introduces IEEE 802.11 channels at the physical layer that can be used for simultaneous transmission.

2.1.1 IEEE 802.11 MAC

IEEE 802.11 is the most commonly used wireless Media Access Control (MAC) protocol. It uses a CSMA/CA media access mechanism. The carrier sense is composed of physical and virtual carrier sense functions. The physical carrier sense tries to avoid collision by testing the media state at the transmitter. But it is on the receiver where collisions occur; therefore, the physical carrier sense cannot avoid collisions in cases such as the Hidden Terminal Problem (HTP). HTP [802.11] is solved by the virtual carrier sense, realized by the RTS (request to send)/CTS (clear to send) mechanism, as shown in Figure 2-1.

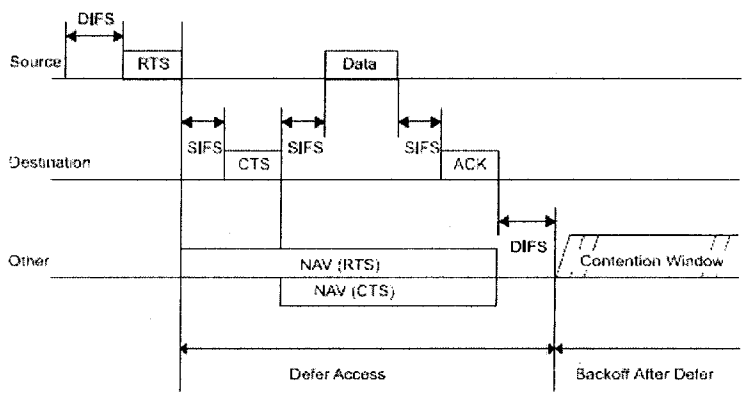


Figure 2-1. IEEE 802.11 RTS/CTS Mechanism [802.11]

The source node accesses the medium when the medium is free for distributed coordination function inter-frame space (DIFS). Short inter-frame space (SIFS) is used between frames belonging to a single dialog. The source node sends a RTS and the destination node replies with a CTS. Other nodes hearing RTS and CTS suspend transmission for a specified time, indicated by the Network Allocation Vector (NAV) in the RTS/CTS frames. The destination node sends back an acknowledgement (ACK) of the MAC layer after successfully receiving the data frame. If the source node cannot receive an ACK because data or ACK frames were dropped due to collision, the data is retransmitted on the MAC layer. After the defer time specified in NAV plus one DIFS, the nodes assume the medium is free and access the medium with the backoff algorithm.

IEEE 802.11 with the RTS/CTS mechanism can achieve collision-free access on the common channel medium. A transmission can defer other transmissions in the contention/interference area. In order for there to be simultaneous transmissions without interference, the transmissions have to be on orthogonal channels.

2.1.2 Interference area

The interference area of a node is the area in which collisions may happen if other nodes transmit or receive on the same channel at the same time. A data-transmitting node is an ACK frame-receiving node where the ACK collision could happen. A data-receiving node is an ACK frame-transmitting node at the same time. This means that each node involving traffic plays the role of both transmitter and receiver on the MAC layer. There are four kinds of frame collisions [Pad05] on the data transmitter and receiver: Data-Data, Data-ACK, ACK-Data and ACK-ACK collision. Therefore the interference area of a transmission link is not only the interference area around the

receiving node, as assumed by some current network layer multi-channel solutions [Lee05, Lee06, Qu06], but also covers the interference area of the transmitter.

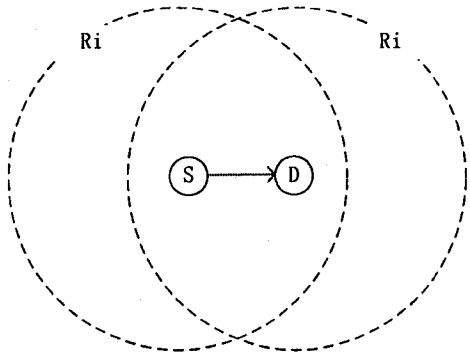


Figure 2-2. Interference Area of a Transmission

As illustrated by Figure 2-2, the interference area of an existing transmission link is the combined disk areas of radius Ri (interference range) centered at the source and destination nodes respectively. Inside the interference area, no other nodes can work (transmit or receive) simultaneously on the same channel with the existing transmission.

IEEE 802.11 exchanges RTS/CTS with one-hop neighbors to avoid interference by nodes in radio transmission range (Rtx). But in practice, Ri is generally estimated to be twice Rtx [Pad05], i.e. Ri = 2Rtx. It is easy to understand that noise needs less power to destroy the reception of a packet than the power required to transmit a packet successfully [Xu02]. The "Ri=2Rtx" interference model is also confirmed by collision implementation in ns-2 in which the wireless packet capture threshold (CPThresh) is set at 10.0 and the path loss factor is set at 4. Ri is calculated to be equal to 1.78 Rtx, according to the equation (1) below [Den04]:

$$CPThresh = SNR = Ps/Pn = (Ri / Rtx)^4 = 10 \quad \text{.....(1)}$$

The Signal Noise Ratio (SNR) at the receiver node is signal power (Ps) divided by noise power (Pn). With the two ray way-point propagation model, the received power Pr

equals $(Pt/range)^4$ [Den04] in which Pt indicates transmission power.

For example, in Figure 2-3, the distance between node A and B, denoted as |AB|, is Rtx, if |BC| <= 1.78*Rtx, then a transmission from node C can corrupt the data reception at B. This is referred to as the *Distant Terminal Problem* [Den04]. Actually, the interference range between B and C is not fixed; it becomes smaller if |AB| decreases.

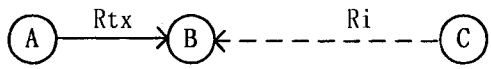


Figure 2-3. Transmission and Interference Range

According to the analysis above, the interference range depends on multiple factors, including transmission power, transmission range, transceiver capture threshold, etc. A generally accepted interference model (Ri = 2Rtx) will be used in this thesis.

2.1.3 IEEE 802.11 channels

IEEE 802.11 specifies multiple channels on the physical layer. IEEE 802.11a (operating on a 5GHz band) offers 12 non-overlapping channels, while the IEEE 802.11b/g (operating on a 2.4GHz band) offers 3 [Fux07]. Nodes can transmit simultaneously on non-overlapping channels without interference. While it is noted in [Fux07] that when interfaces are too close, interference between non-overlapping channels is still obvious. To avoid inter-channel interference, multiple interfaces should work on different frequency bands (2.4GHz and 5GHz) [Draves04], or be separated by enough distance; for instance, 38 cm, according to [Fux07].

When the interface/transceiver switches channel (working frequency), there is a hardware-dependent delay that varies from tens to hundreds of microseconds [Jeo05]. The channel-switching delay is a considerable overhead if the switch is slow and

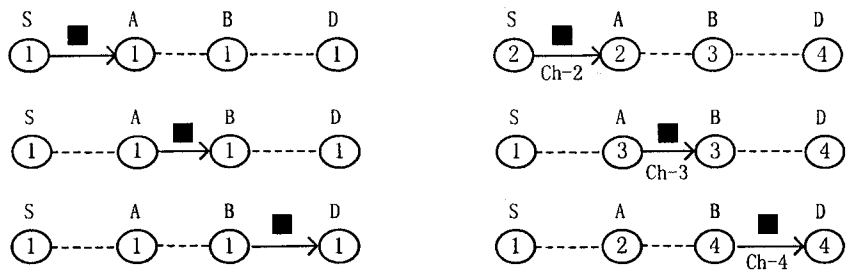
frequent.

2.2 Channel assignment schemes

There are three channel assignment schemes: the flow-based [So04b], node-based [Kya05, Lee05, Lee06, Qu06] and negotiation-based [Wu00, So04a, McG06] channel assignment scheme.

2.2.1 Flow-based channel assignment

Flow-based channel assignment means that the transmission along the flow is on the same channel. The Multi-Channel Routing Protocol (MCRP) [So04b] is one example of this approach. The channel assignment is coupled with route discovery and determined when a route is set up on demand by the traffic flow. All nodes on the flow use the same channel to transmit and receive packets. Because nodes are not switching channels, the *deafness problem* [So04b], in which a receiver listening on a different channel cannot hear the sender, can be avoided. The downside of this approach is that the routing protocol becomes complicated when it is coupled with channel assignment.



(a) Flow-based Channel Assignment *(b) Receiver-based Channel Assignment*

Figure 2-4. Comparison of Two Channel Assignment Approaches

Another problem with the flow-based scheme is that there is no simultaneous transmission along the flow in an interference area. As Figure 2-4 (a) shows, only one of

three transmissions along the flow from S to D can be carried out at the same time.

2.2.2 Node-based channel assignment

Node-based channel assignment includes Receiver-based Channel Assignment (RCA) and Transmitter-based Channel Assignment (TCA). It is decoupled with the routing protocol, and thus simplifies both the routing protocol and channel assignment. RCA assigns a channel to a node as its receiving channel. Similarly, TCA assigns a channel to a node as its transmitting channel. The channel assignment is based on the network topology. Each node is assigned a unique channel to its neighbors in the interference range.

As opposed to the flow-based scheme, as Figure 2-4 (b) shows, a node needs to switch to different channels for transmitting and receiving data. If every node is equipped with only one data interface, the transmitter and receiver need to be coordinated to switch to the same channel for the communication. The channel *switching coordination* is a challenge at the network layer because of the slow message exchange, compared to frame exchange at the MAC layer.

Kyasanur et. al [Kya05] use multiple data interfaces for each node to avoid this *switching coordination* problem. Each node has one data-receiving interface fixed at the assigned receiving channel with the RCA scheme, and the other data transmitting interface(s) is switchable to different channels to transmit data. When a node sends packets, it switches a transmitting interface to the receiving channel of the next hop node. In this way, no coordination is needed between the transmitter and receiver. Even with multiple data interfaces, the coordination cannot be avoided when using the TCA scheme. A receiver needs to tune its receiving interface to the transmitter's transmitting channel

for data reception. However, the receiver has no local knowledge of when and which channel to switch without notification from the transmitter. Therefore, RCA is commonly used as the node-based channel assignment scheme.

2.2.3 Negotiation-based Channel Assignment

Negotiation-based Channel Assignment (NCA) is an on-demand channel assignment scheme. The transmitter negotiates with the receiver for a common free channel to be used. NCA can achieve interference-free channel assignment if we consider only transmission range interference. Unlike with the proactive node-based scheme, fewer channel resources are required because idle nodes will not be assigned a channel. The NCA approach is usually used by MAC layer solutions [Wu00, So04a]. The negotiation is realized via RTS/CTS carrying channel usage information. Per-packet negotiation is feasible at the MAC layer, but not the network layer, because message exchange is slow.

2.3 Solutions at the MAC, LL or network layer

Multi-channel solutions have been proposed on different OSI layers: the MAC layer [Wu00, So04a, Mah06, McG06], Logical Link layer [Bah04b] or network layer [Kya05, Lee05, Lee06, So04b, Qu06].

Wu et al. [Wu00] proposed an on-demand negotiation channel assignment on the MAC layer. Two interfaces are used for each node. RTS/CTS with channel usage information are exchanged on the control interface to negotiate the channel to be used before each data packet transmission. If the packet size is small, the channel negotiation per packet with RTS/CTS is a big overhead that may become a bottleneck even if a

dedicated control interface is used [Wu00]. MMAC [So04a] and LCM-MAC [Mah06] are similar to [Wu00] with regard to the channel negotiation mechanism, except that only one interface is used and the negotiation process is carried out on synchronized time slots on a common channel.

SSCH [Bah04b] is a Logical Link layer solution. Each node switches channels in several (e.g., four) pseudo random hopping schedules. The transmitter follows one or more hopping schedules of the receiver to communicate on the same channel. Hopping schedules of nodes with disjointed communications are mostly non-overlapped; therefore there is no interference between the simultaneous communications. Channel-hopping takes about 150 to 200 μsec for a commercial off-the-shelf 802.11b transceiver [Jeo05]. The hopping time is not negligible overhead.

Most MAC and LL layer multi-channel solutions are designed to work with a single interface wireless node, but de facto IEEE 802.11 is modified and thus impedes its deployment.

Network layer multi-channel solutions [So04b, Lee05, Lee06, Qu06] do not change the layer two protocols. They determine communication channels based on topology information on the network layer. MCRP [So04b] takes the approach of flow-based scheme channel assignment. Reactive routing discovery is attempted on all channels. The channel on which a route is set up successfully will be assigned as the flow channel. DSDV-MC [Lee05] and OLSR-MC [Lee06] use the RCA channel assignment scheme, which is decoupled with the routing establishment. The proactive routing message is extended to make each node aware of its neighbor channel usage. Each node proactively determines its receiving channel by selecting a channel that is not used by its

neighbors. As explained in Section 2.2.1 and 3.2.1, the flow-based and receiver-based channel assignment schemes cannot always generate interference-free channel assignment.

2.4 Single interface or multiple interfaces

Traditional mobile equipment is restricted by the single interface. As multiple-interface mobile equipment has become common, at a low hardware price, multi-interface approach has attracted more study. This section reviews the current multi-channel solutions from the perspective of the number of interfaces used for each node.

2.4.1 Single interface

MMAC [So04a], SSCH [Bah04b] and MCRP [So04b] are multi-channel solutions using single interface nodes. A single interface node in a multi-channel network has the *deafness problem.* To overcome this problem, many multi-channel solutions make modifications on the MAC/LL layer and require time synchronization between wireless nodes. Many routing control broadcast messages have to be retransmitted on various channels because some neighbors, on different channels, cannot hear them. The abovementioned three single interface examples are described below.

MMAC realizes channel control by negotiation in a synchronized time window. Nodes switch to a common default channel during the synchronized time window at the start of every fixed time interval. They negotiate channels for use during the interval.

SSCH is a LL layer solution in which nodes communicate on an overlapped channel-hopping schedule. Broadcast packets transmitted in any one slot are likely to reach only some of the nodes, because the nodes are often on different channels. SSCH

handles this issue through repeated LL layer retransmission of broadcast packets enqueued by higher layers. The retransmission increases bandwidth consumption.

The interface stays on a channel for a period long enough to transmit many packets. The period is a 100ms interval in MMAC and a 10ms time slot in SSCH. This is to reduce the synchronization requirement and channel-switching overhead. On the other hand, it increases the end-to-end delay when a packet traverses through the multi-hop path.

MCRP uses the routing protocol coupled flow-based channel assignment scheme. There is no *deafness problem* along the flow, which is assigned the same channel after the route is established. But during the route discovery process, broadcast routing messages have to be transmitted on all channels.

The above three examples show that the *deafness problem* is a challenge for efficient control communication on a single interface in a multi-channel network.

2.4.2 Data interface

In multi-interface solutions, a node may be equipped with a single data interface [Wu00, McG06, Lee05, Lee06] or multiple data interfaces [Kya05, Qu06]. Many researchers still prefer to keep the number of interfaces as low as possible. Too many interfaces—for example, one interface per channel—are not feasible because of the power consumption and interference between close interfaces [Dra04].

The channel *switching coordination* with single data interface is not big problem for most MAC layer multi-channel solutions [Wu00, McG06] because the coordination is integrated into the channel negotiation process using RTS/CTS; but it is a problem on the network layer because of the slow coordination with message exchange. Lee et al. did not

address this problem in their recent work [Lee05, Lee06]. Multiple data interfaces [Kya05, Qu06] with a dedicated receiving data interface might be the only way to avoid this problem.

Lee et al. [Lee05, Lee06] use RCA for initial node channel assignment. Each node has only one data interface. When a node intends to send packets, it switches to the next hop node channel and will not switch back. Before switching, it broadcasts a channel update message to notify neighbors of its new channel. As Figure 2-5 illustrates, all nodes along the flow will ultimately change to the same channel as the destination node. As a result, it becomes flow-based channel assignment. Although the channel *switching coordination* problem does not exist anymore, interference appears between transmissions along the flow.

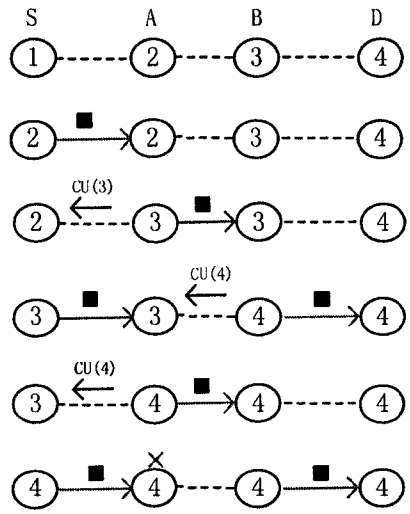


Figure 2-5. OLSR-MC [Lee06] Channel Update

2.4.3 Control interface

In a multi-channel network, nodes may stay in different channels. It is difficult for the channel and routing control messages to communicate between nodes in the absence of a common communication channel. Section 2.4.1 illustrated the challenge in control

communication with a single interface node. Multi-Channel Routing (MCR) [Kya05] is a multi-interface multi-channel solution, but the control interface is still mixed with the data interface. Broadcast routing control messages have to be transmitted on all channels to overcome the *deafness problem.*

A dedicated control interface has been proposed for a long time. The MAC layer proposal DCA [Wu00] negotiates the data channels through RTS/CTS on the dedicated control interface. Dual-radio, proposed by Bahl et al. [Bah04a], uses a low power control interface to realize power control in MANETs. It sheds light on the approach of using a dedicated control interface for more efficient control mechanisms. Lee et al. [Lee05, Lee06] proposed two network layer multi-channel solutions (DSDV-MC and OLSR-MC) with a dedicated control interface. A control interface on the common channel is for routing and channel control traffic; another interface switches on multiple channels for data traffic. A proactive routing message is extended with node channel information to support RCA channel assignment. Qu et al. [Qu06] proposed dedicated control interface and two data interfaces for each node.

Chapter 3 A Network Layer Multi-interface Multi-channel Proposal

This chapter presents a network layer multi-interface multi-channel solution for improving the performance of wireless ad hoc networks. An overview of the solution is described in Section 3.1. The channel assignment scheme is a central part of the solution. Two alternative schemes, RCA and NCA, are described in Sections 3.2 and 3.3. The channel switching time penalty is a related issue in a multi-channel solution. To address this problem, an optimized queue-based channel switching [Kya04] mechanism is presented in Section 3.4. A dedicated control interface can be used for performance improvement other than multi-channel controlling; Section 3.5 discusses how it improves wireless TCP performance.

3.1 Overview of proposed solution

In this thesis, a multi-interface multi-channel network layer solution is proposed. One interface is dedicated to controlling multi-channel assignment and routing, and the other two interfaces are used as data transmitting (Tx) and receiving (Rx) interfaces respectively on different channels and for realizing full duplex data traffic. Two channel assignment schemes, RCA and NCA, are proposed at the network layer.

3.1.1 Dedicated control interface

Data traffic needs multi-channel for simultaneous communication, but efficient channel and routing control requires a common channel. Section 2.4.3 illustrated the

challenges in multi-channel solutions without a dedicated control interface. After separating the control to a dedicated interface on the common channel, data and control interfaces are tuned to carry out different tasks efficiently.

A dedicated control interface facilitates the design of simple and efficient routing and multi-channel control protocols. Furthermore, when control traffic is separated from the data traffic on different interfaces, the critical control packets are transmitted more reliably without contention with data packets. Most MANET routing messages are broadcast packets. Broadcast is not protected by RTS/CTS [802.11]. In data and control mixed interface, they are easily dropped because of collision with data packets in the busy medium. This ultimately degrades data traffic performance.

3.1.2 Dual data interfaces

Besides the control interface, each node has two data interfaces, dedicated to data transmitting and receiving respectively. The data-receiving interface stays on the assigned Rx channel. The data-transmitting interface is switchable to the next hop node channel for transmitting. The dual-data-interface node structure is to eliminate the channel *switching coordination* problem in a network layer multi-channel solution. It also realizes full-duplex transmission for better throughput and short end-to-end delay.

IEEE 802.11b has three non-overlapping channels and its data rate is low, but it is good enough for the control interface. IEEE 802.11a is used for data interfaces because it has more (12) non-overlapping channels and a high data rate. Two data interfaces may have interference even though they are on different channels; enough distance should be put between them.

3.1.3 Network layer solution

This thesis realizes the multi-channel solution at the network layer, thus working with the off-the-shelf IEEE 802.11 layer two (MAC/LL) protocol. Two channel assignment schemes, Receiver-based Channel Assignment (RCA) and Negotiation-based Channel Assignment (NCA), are used in the network layer solutions in this thesis. The proactive routing protocol Optimized Link State Routing Protocol [OLSR] is extended to support multi-channel control.

RCA is a proactive channel assignment scheme that is similar to the previous proposals in DSDV-MC [Lee05] and OLSR-MC [Lee06]. This thesis improves the previous RCA solutions by using a more accurate interference model, described in Section 2.1.2. The channel-switching coordination problem overlooked in [Lee05, Lee06] is solved using dual data interfaces. Another network layer channel assignment scheme, flow-based channel assignment, is not considered in this thesis because it is complicated and not interference free (Section 2.2.1).

NCA is an on-demand channel assignment scheme usually used at the MAC layer. The Network layer NCA solution improves RCA with optimized channel resource utilization by assigning channels only to nodes with traffic. Since the traffic pattern is considered in addition to the topology information, NCA can avoid some channel conflicting scenarios in RCA.

In the network layer RCA or NCA scheme, each node needs to know its neighbor's channel usage information. Nodes exchange channel information via extended Hello messages. RCA needs periodic Hello messages to determine and maintain each node channel; therefore, RCA requires the support of a proactive routing protocol like

OLSR. Some on-demand routing protocols, such as Ad hoc On-Demand Distance Vector [AODV] Routing, exchange periodic Hello messages only between nodes along the traffic flow. AODV cannot be used for RCA topology-based channel assignment, but it is sufficient for NCA, which does not maintain channels for the nodes that are not participating in the communication, while OLSR is still used for NCA in this thesis in order to make the same comparison base line with RCA. Furthermore, the high control overhead of the proactive routing protocol is not a problem when a dedicated control interface is used, and the proactive routing protocol is faster in route establishment than the on-demand routing protocol.

OLSR is only running on the control interface. The data and control interfaces have the same assumed transmission range and share the same topology; therefore, routing information learned on the control interface can be used for data control on the data interfaces. The detailed RCA and NCA solutions are described in Sections 3.2 and 3.3 respectively.

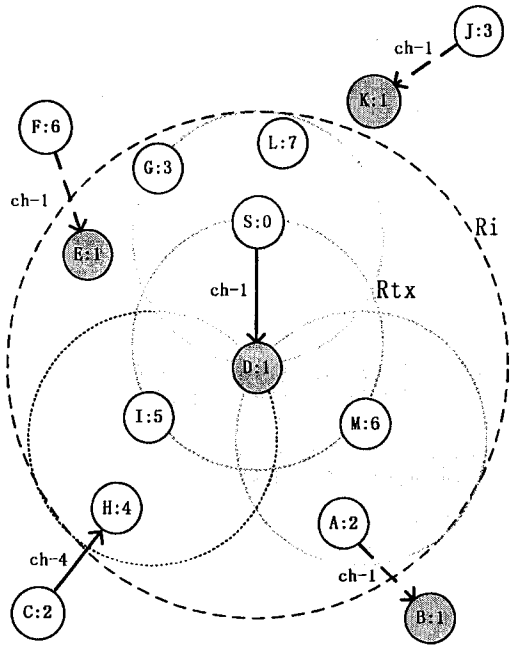
3.2 Receiver-based channel assignment

This section first introduces the RCA high-level design principles, then the multi-channel extended OLSR and RCA scheme implementation is described.

3.2.1 RCA high-level design

The RCA scheme improves the previous work by assigning a unique node (receiving) channel in a two-hop range area. All nodes are initially assigned a random channel as a receiving channel. Each node notifies the neighbors in a two-hop range about its assigned channel. When a neighbor receives this notification, if its own assigned

channel is the same, it will switch its channel to a free channel. For example, as shown in Figure 3-1, D:1 indicates the node D and its assigned receiving channel (ch-1). Node D notifies its neighbors S, I, L, M, G, H and A that the ch-1 is being used. When node H is made aware of this through a Hello message from node I, and if H was also assigned the ch-1, H will randomly select a new free channel, for example ch-4. Thus if there are SD and CH flows, they can be transmitted simultaneously without the interference that might otherwise exist on the same channel. Channel usage information is embedded in OLSR Hello message (please see Section 3.2.2 for details).



RCA

Figure 3-1. RCA with Unique Channels in Two-hop Range

RCA cannot avoid interference in a few scenarios:

- Scenario 1: three-hop neighbor E is assigned the same channel, but it could be inside two-hop interference range (Ri = 2Rtx). Then D and E interfere with each other.
- Scenario 2: Node B is outside Ri of node D. Node B and D can be assigned the

same channel, but transmitter A is inside Ri. Then A and D interfere with each other.

- Scenario 3: Node K is a two-hop neighbor of node S, but not in Ri range of node D, so it can be assigned the ch-1. Then S and K interfere with each other.

As indicated in Scenario 1, MAC layer interference cannot be accurately represented by network layer (topology) information. In fact, interference is dependent on SNR. Channel collision on the network layer is only a simplified model and interference cannot be avoided completely. Sometimes, a node over two hops away can still be in interference range Ri. This is an inborn problem in the network layer multi-channel approach.

In Scenario 2 and 3, the traffic pattern is unknown to the proactive RCA in advance. The RCA scheme depends solely on topology and considers only the node-receiving channel. But when a certain traffic scenario applies, the transmitting nodes may cause channel collision with other transmitting or receiving nodes. It might not be possible to resolve channel collision in all scenarios with proactive topology-based RCA. On-demand scheme NCA can overcome this problem because it considers channel assignment for both the transmitting and receiving nodes.

3.2.2 Extended OLSR and RCA implementation

This RCA multi-channel solution realizes routing protocol decoupled channel control with very low overhead by piggybacking on the extended OLSR Hello message and *neighborTuple*. This thesis implements the extension in ns-2 with an OLSR implementation [OOLSR] from INRIA. Although the RCA implementation is integrated

with the OLSR protocol, the channel control is independent of the routing protocol.

3.2.2.1 Hello message format

The new OLSR Hello message structure (Figure 3-2) has two new fields: *orig_chan* and *neighbor_chan_bitmap*. The field *orig_chan* specifies the node (receiving) channel of the Hello message originator. The *neighbor_chan_bitmap* field specifies the channel usage status of one-hop neighbors.

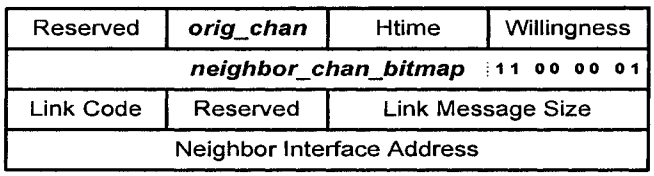


Figure 3-2. OLSR Hello Message Format in RCA

neighbor_chan_bitmap represents the overall channel status of the one-hop neighbors. It is composed of 32 bits to indicate the status of a maximum of 16 channels, with 2 bits for each channel. The channels refer to orthogonal channels that may not directly map to the corresponding 802.11 physical channels. For example, 802.11b/g in the U.S. has 11 channels, but only 3 of them are non-overlapping (orthogonal) channels: channels 1, 6 and 11 [Fux07]. Then, only the lowest 6 bits in *neighbor_chan_bitmap* are concerned. 802.11a has 12 non-overlapping channels. 32 bits is sufficient for current and future underlying physical protocols.

Table 3-1. Definition of Channel Status

2bit channel status	*Definition*
00	Free channel
01	Used by one neighbor node
11	Used by multiple neighbor nodes

A channel status in *neighbor_chan_bitmap* could be 00, 01 or 11, as shown in Table 3-1. For example, "00" means the channel is a free channel in a one-hop range; i.e.,

it is not assigned to any one-hop neighbor as a node-receiving channel.

3.2.2.2 Hello message generation and processing

When a node generates a Hello message, it constructs the *orig_chan* field with its assigned (receiving) channel. Each node has a randomly assigned initial channel after power-on. The channel may change later if it conflicts with another node in a two-hop range. The Hello message originator iterates the *N_neighbor_channel* field of all *neighborTuples* in its *neighborTupleSet* to construct the *neighbor_chan_bitmap* field according to the channel status definition in Table 3-1. For example, if a node has only one (one-hop) neighbor assigned ch_0 as the node (receiving) channel, there will be only one *neighborTuple* with *N_neighbor_channel* field as "0", and then the channel status of ch_0 in *neighbor_chan_bitmap* will be "01". Similarly, if no neighbors are assigned ch_1 and ch_2, then the status of these two channels will be "00". Ch_3 status is "11" if it is assigned to more than one neighbor. With this example, the *neighbor_chan_bitmap* will be "... 11 00 00 01", as shown in Figure 3-2.

The new Hello message is extended this way to construct the new format *neighborTuple,* which is extended with two new fields *N_neighbor_channel* and *N_2hop_channel_bitmap*. The two new fields correspond to the "*orig_chan*" and "*neighbor_chan_bitmap*" of the Hello message from the neighbor. They are updated in the Hello message processing. The multi-channel extension need not be concerned with the channel information update and expiration mechanism. It is taken care of by standard OLSR *neighborTuple* maintenance.

The new *neighborTuple* provides support for multi-channel packet forwarding and channel assignment. When a node is to send packets, it looks up the routing table to

find the next hop address, searches its local OLSR *neighborTupleSet* to find the corresponding *neighborTuple,* and then reads the *N_neighbor_channel* field of the *neighborTuple* to find out the transmission channel.

3.2.2.3 Channel collision detect and resolve

A node initiates with a random channel on start-up. It updates its node channel when it detects channel collision in processing the received Hello messages. For example, node B (at ch_b) receives a Hello message from its neighbor A. One-hop neighbor channel collision occurs if ch_b equals the *orig_chan* of the Hello message. Two-hop neighbor channel collision exists if the ch_b status in *neighbor_chan_bitmap* of this Hello message is "11". This means that besides B, node A has other one-hop neighbor(s) on ch_b as well.

On detecting channel collision in processing Hello messages, a node randomly selects a free channel that is not used by its one- and two-hop neighbors. A node finds all the free channels by iterating the *neighborTupleSet.* A channel is free if it is not used as a *N_neighbor_channel* in any *neighborTuple* and its status is "00" in the *N_2hop_channel_bitmap* field of all *neighborTuples.* If no free channel is available, the node stays at the current channel.

If a node selects a new channel, it will not switch its receiving interface to the new channel until it has sent out a Hello message to notify its neighbors about its new channel. This concern minimizes the lag between node channel switching and the neighbor's *neighborTuple* update; hence, it decreases the possibility of packets being sent on the old channel, while the next hop node has switched to a new receiving channel.

3.3 Negotiation-based channel assignment

RCA has two limitations: first, it cannot resolve channel collision related to a traffic pattern, like Scenario 2 & 3, mentioned in Section 3.2.1. Second, RCA requires a large number of orthogonal channels to keep each node assigned with a different channel to its neighbors in the interference range, even when the nodes are not involved in data transmission.

NCA is a way of solving the above problems. The channel assignment considers the traffic pattern instead of topology alone. And the node's channel is not proactively assigned, but negotiated on-demand by traffic.

The NCA high-level design principle will be discussed in Section 3.3.1 below, and the detailed negotiation protocol is then presented in Section 3.3.2. After that, the channel maintenance process used to support the NCA is described in Section 3.3.3.

3.3.1 NCA high-level design

Negotiation on the network layer by message exchange is not as fast as that on the MAC layer; on the other hand, the network layer NCA does not need per-packet negotiation like the MAC layer solution does. It assigns a node-receiving channel that is dependent on traffic pattern and topology. The negotiation is not frequent because the traffic pattern and topology change slowly (in seconds). Therefore, the network layer NCA is a feasible approach. To the best of our knowledge, this thesis is the first work using this approach.

With NCA, two nodes negotiate a common free channel to be assigned to the receiver node. Then both the transmitter and the receiver will claim that channel to be

used in their respective (two-hop) interference range. Subsequently, no other nodes in the interference area can use the same channel for receiving or transmitting. The traffic pattern-based channel assignment can solve the almost all RCA problems, except Scenario 1, mentioned in Section 3.2.1.

As illustrated in Figure 3-3, the transmission from node S to node D negotiated ch-1 to be used. Then all the other nodes in a two-hop interference range from S and D cannot use ch-1. Node E and F cannot be reached by two hops if node G does not exist. So they may use ch-1 for communication, even node E and F are in interference range of node S. As explained for RCA Scenario 1 in Section 3.2.1, this problem cannot be solved with a network layer channel assignment solution.

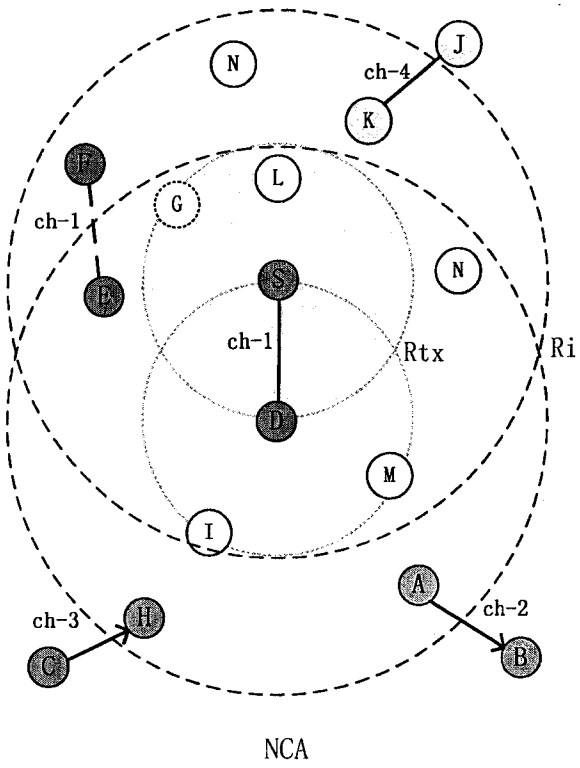


Figure 3-3. Negotiation Channel Assignment

The incorrect behavior in RCA Scenario 2 is solved in NCA. Traffic AB can negotiate an interference-free channel. Although the ch-1 is free for node B, it is not free

for node A, thus they will not use ch-1 to transmit. The problem in RCA Scenario 3 is also solved, because node K is now a two-hop neighbor of transmitter node S. JK will negotiate a channel other than ch-1.

As the example shows, NCA can generate interference-free channel assignment for most scenarios. NCA also overcomes the second limitation of RCA. No channel is assigned to a node without data traffic; therefore the scarce channel resources are used efficiently and many fewer channels are required.

3.3.2 Channel negotiation process

First, this section briefly explains the network layer NCA protocol with the protocol message flow. Next, the NCA implementation is presented through a description of how different types of packet/message are processed. After that, some design details and time-related implementation issues are discussed.

Figure 3-4 shows the steps of NCA process carried out on the control interface. The numbers, in seconds, represent delay to avoid broadcast collisions.

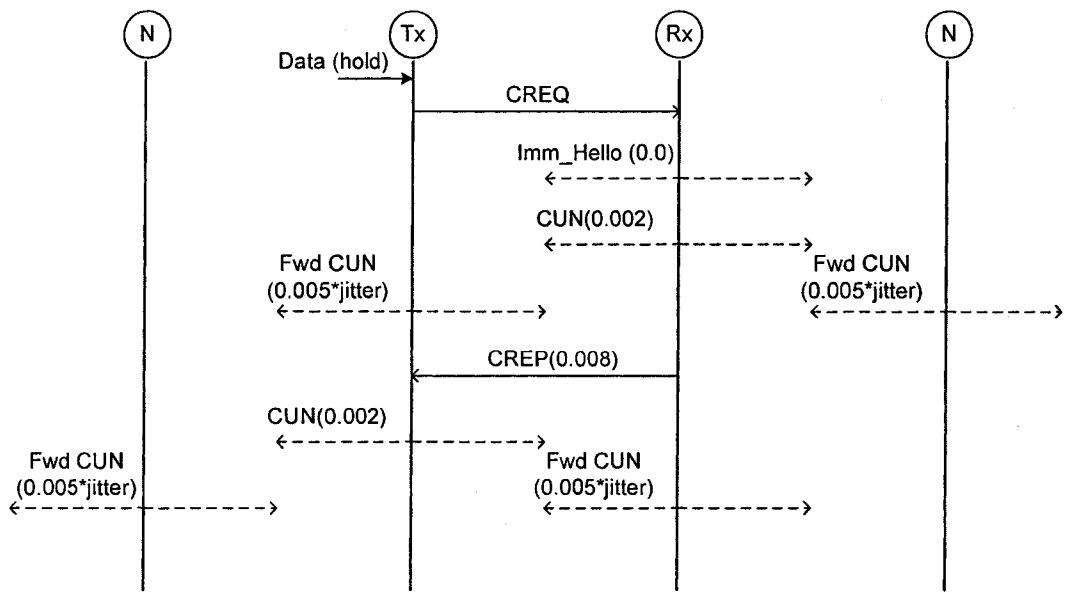


Figure 3-4. NCA Message Flow Chart

1) When a transmitting (Tx) node sends a data packet, it looks for the next hop node receiving channel from the corresponding neighborTuple. If the channel is not the initial default channel (which means the next hop node has been assigned a channel after a negotiation process), the packet will be sent on this channel; otherwise, a unicast Channel Request (CREQ) message is triggered to the next hop node
2) After receiving the CREQ message, receiver (Rx) selects a common free channel between itself and the transmitter. An immediate Hello message (Imm_Hello) is broadcast to update *neighborTuple* in neighbor nodes.
3) A two-hop broadcast Channel Usage Notification (CUN) message is sent from the receiver to claim that the negotiated channel is busy in the two-hop range interference area.
4) A Channel Reply (CREP) message is then sent to the transmitter to trigger a CUN broadcast from the transmitter. Thus the negotiated channel is claimed as busy in the transmitter's interference area.

After negotiation, a channel is assigned to the receiver, and the channel is claimed as busy in the interference area of the current transmission. No other node in this area can use the same channel (except when transmitting to the same receiver), so the interference free channel assignment is realized.

This thesis implements the NCA protocol in ns-2 and extends the OLSR plugin [OOLSR]. Figure 3-5 shows the functional components of the implementation. Channel negotiation is processed in the ns-2 routing agent. The OLSR plugin maintains the channel usage and updates the channel in *NeighborTuple*. The NCA protocol is described in more detail from the perspective of how each type of packet is processed.

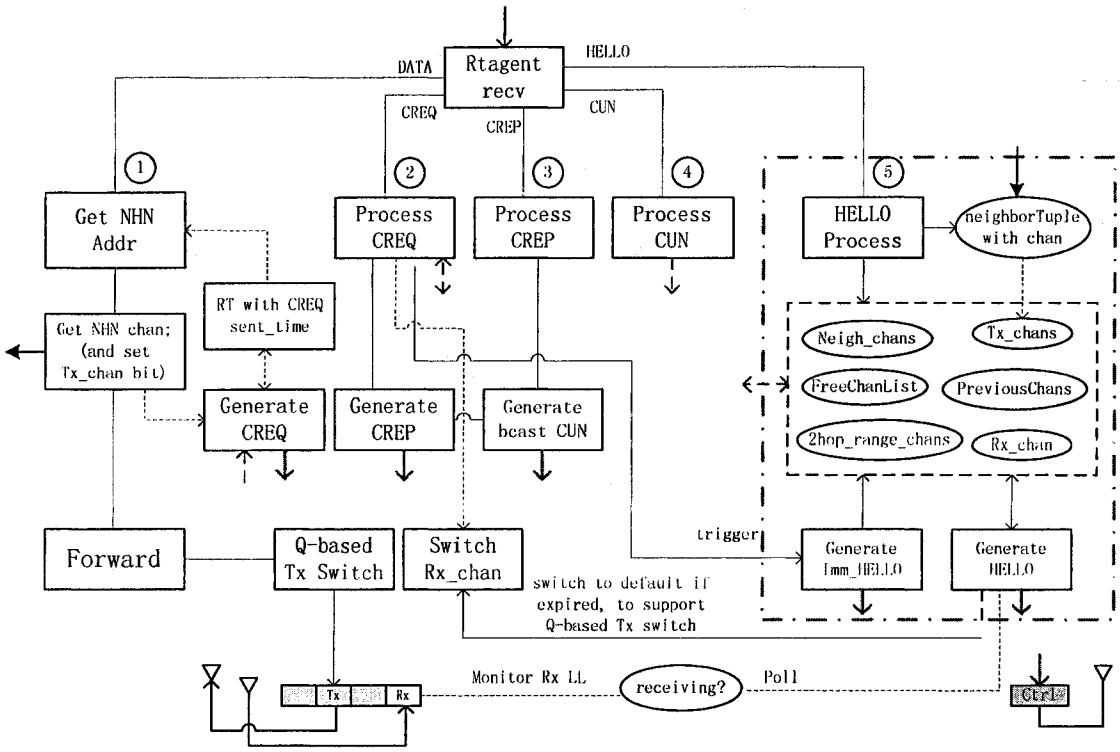


Figure 3-5. NCA Functional Components

(1) RAgent (routing agent) receives a data packet to be forwarded. It looks up the next hop address from the routing table, and checks the receiving channel of the next hop node from the OLSR *neighborTuple*. If the next hop node has been assigned a non-default receiving (Rx) channel, it forwards the packet on this channel and sets the bit in Tx_chan in the OLSR plugin to monitor the transmitting (Tx) channel being used. If the next hop node channel is still on the default channel, it sends a unicast Channel Request (CREQ) message to the next hop node. The "Generate CREQ" function gets the node *freeChanList* from the OLSR plugin, and generates and sends a CREQ message on the control interface.

(2) When the next hop node receives the CREQ message, it checks the sender's *freeChanList*, contained in the CREQ. The receiver checks its own *freeChanList* and selects a common free channel as the negotiated channel to be assigned to itself as

the receiving channel. The successful negotiation will trigger OLSR to generate an immediate Hello message to notify neighbors to update the *N_neighbor_channel* field of the *neighborTuple* corresponding to this node. Then the receiver sends a two-hop broadcast Channel Usage Notification (CUN) message to claim the channel to be busy in the receiver's interference area. After that, a CREP (Channel Reply) message containing the negotiated channel information is sent to the data transmitter. Finally, the receiver switches its data Rx interface to the negotiated channel. On the other hand, if there is no common free channel for the transmitter and the receiver, the receiver does not respond to the CREQ message, and the data Rx interface stays on the default channel.

(3) When the transmitter receives the CREP message, a two-hop broadcast message CUN is triggered to claim the negotiated channel as busy in the transmitter's interference area.

(4) When the neighbors receive the CUN message, they update the channel usage information in the OLSR plugin.

(5) A Hello message in NCA is used to maintain channel usage information in addition to its standard OLSR function. The channel usage information is also affected by CREQ, CUN messages and Tx/Rx data interfaces activity. This will be described in more detail in Section 3.3.3. The purpose of channel usage maintenance is to provide the *freeChanList* required in CREQ generation and processing.

After describing the NCA channel negotiation process, we will address some issues concerning design and implementation:

Not Resend CREQ

For simplicity, CREQ messages are designed as UDP packets and not re-sent even if they are dropped in transmission. CREQ will be retried after a CREQ_INTERVAL if channel negotiation fails for any reason. Data traffic can endure a slow negotiation process. Before successful channel negotiation, data packets can still be transmitted on the default channel, although interference may exist with other traffic.

CREQ Retry

If the next hop node is on the default Rx channel and the last CREQ was sent before CREQ retry interval, a new data packet will trigger a CREQ retry. An unavailable free channel is the main reason for negotiation failure. A new free channel may appear if the traffic pattern or network topology changes. To determine the proper value of CREQ_INTERVAL, we refer to the OLSR TC message, which is used to build a topology information base. CREQ_INTERVAL should be a value comparable to TC_INTERVAL (5s) because they all depend on the same topology dynamic. Taking into consideration the traffic pattern changing, which can further increase the dynamic of channel usage, therefore, we set the default CREQ_INTERVAL to 4 seconds.

Imm_Hello

A periodic Hello message is used for channel usage maintenance. An immediate Hello (Imm_Hello) message is used to notify the transmitter to update the *N_neighbor_channel* field of the *neighborTuple*. Imm_Hello has the same structure as the periodic Hello. It is generated with Rx_chan (Figure 3-6) set as the negotiated channel after a successful negotiation in CREQ processing. The scheduled Hello

message is not used because it can delay data transmission in using the new channel by up to 2s (Hello_Interval) after successful channel negotiation.

Time Issues

NCA message sequence and timing are critical. Two time-related issues are concerned in trying to avoid sporadic behavior resulting in wrong channel assignment.

- **Hold data packet**

Sometimes the first and second hop nodes in a two-hop flow are assigned the same receiving channel after negotiation with NCA. For example, nodes B and C in flow A→B(1)→C(1) are all assigned ch_1. Transmission AB and BC cannot occur simultaneously.

This problem happens when the second hop channel negotiation starts before the first hop negotiation is complete. The channel negotiation is triggered by a data packet. The negotiation process takes a few milliseconds for the transmission of the CREQ, Imm_Hello and CUN messages. If the data packet is forwarded on the default channel to the next hop before the first hop negotiation is complete, two negotiation processes may get the same channel because the three nodes have the same *free_channel_list*.

A solution is to hold data packets for 20ms after the first CREQ is sent; thus the negotiation in the next hop along the flow will not start too early and get the same channel as the negotiation in the previous hop.

- **Reduce broadcast packet collision**

The Imm_Hello and CUN messages are broadcast packets. A broadcast packet is

not protected by the RTS/CTS mechanism and may be dropped for collision. As Figure 3-4 shows, the message sequence is carefully designed and jitter is used to reduce the chance of a collision.

An Imm_Hello message is sent before the other negotiation messages to complete the primary task in negotiation—to notify the transmitter update *N_neighbor_channel* field of the corresponding *neighborTuple*. The broadcast CUN message is then sent by the receiver with 5ms jitter to claim the channel in its two-hop interference range. The CREP is sent 8ms after receiving the CREQ, to trigger a CUN message broadcast by the transmitter. The CUN broadcast from the receiver and the transmitter is separated by few milliseconds and collision is avoided.

3.3.3 Channel usage maintenance

In CREQ generation and processing, a node needs to know its own *freeChanList*, which indicates the free channels in its two-hop range interference area. The OLSR Hello message is extended to support the NCA process. Channel usage information is composed of seven variants in the OLSR plugin—*Tx_chans, Rx_chan, 1hop_neighbor_chans, 2hop_neighbor_chans, twohop_range_chans, previousChanUsage* and *freeChanList*. This section describes the maintenance of these variants and the algorithm for calculating the *freeChanList*.

OLSR Hello is modified to distribute channel usage and make the node aware of the channel usage information in its interference (two-hop range) area. The new OLSR Hello format in NCA is shown in Figure 3-6.

Tx_chans		*Rx_chan*	Reserved
1hop_neighbor_chans		Htime	Willingness
Link Code	Reserved	Link Message Size	
Neighbor Interface Address			

Figure 3-6. OLSR Hello Header Structure in NCA

Tx_chans is a 16-bit variant representing the usage of 16 channels. "1" in bit-x means ch_x is used for transmitting. In Figure 3-5, the function *"Get NHN Chan"* gets the next hop node channel for data packet transmission and then sets the corresponding bit in *Tx_chans*. After a Hello message generation, *Tx_chans* is reset to be refreshed with data Tx interface activity during the Hello interval.

Rx_chan is set to be the negotiated channel in function *"Imm_Hello Generation"* (Figure 3-5) after successful channel negotiation. During the Hello interval, the ns-2 node monitors the LL component of the data Rx interface: if there is a packet receiving on the assigned receiving channel, the "*isReceiving*" indicator of the channel is set. Before the generation of a Hello message, this indicator is polled. If it is set, *Rx_chan* is kept as the assigned channel; otherwise, the assigned channel expires and *Rx_chan* is reset to the default channel. The node's assigned channel can be safely expired if no packet is received in a Hello interval. The traffic flow can be assumed to have stopped because inter-packet arrival time is not likely to be longer than 2s.

1hop_neighbor_chans indicates the channels used by one-hop neighbors as Tx or Rx channels. It is distributed in the Hello message to notify one-hop further nodes of their two-hop neighbor channel usage. After every Hello message generation, this variant is reset, and during the following Hello interval, it is constructed by processing Hello messages from the neighbors.

1hop_neighbor_chans | = *hello→Tx_chans* | *1<< hello→Rx_chan*

Figure 3-7 presents the relation of the seven variants of channel usage information. The dotted text boxes indicate the message input. Node's *freeChannelList* is calculated as

freeChanList = ~ (*twohop_range_chans* | *previousChanUsage)*

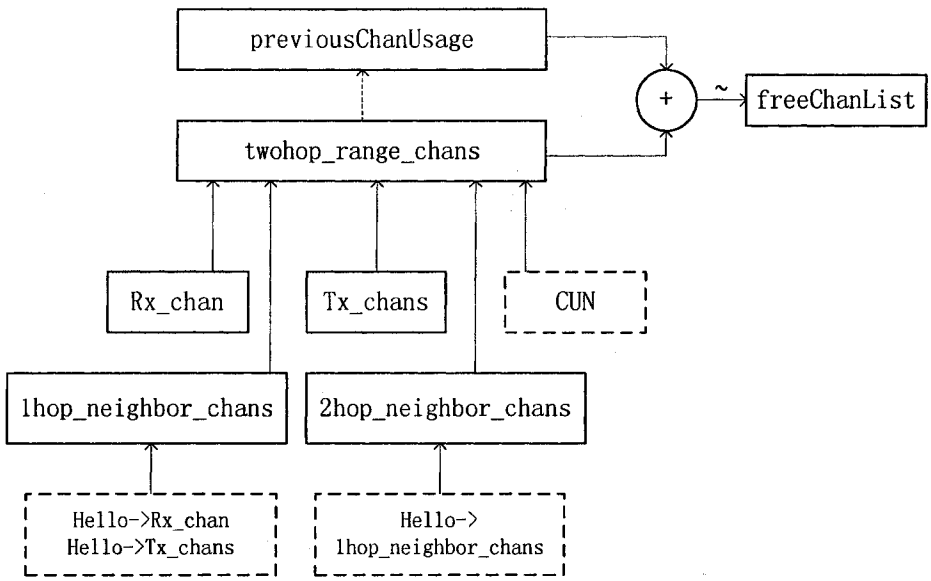


Figure 3-7. Channel Usage Update

freeChanList is defined as the channels not used by any nodes in the two-hop range interference area. The channels used in the two-hop range are indicated by the bits of *twohop_range_chans,* which equals

Rx_chan | Tx_chans | 1hop_neighbor_chans | 2hop_neighbor_chans | CUN

The node's own channels (*Rx_chan, Tx_chans*) and one-hop neighbor's channel (*1hop_neighbor_chans*) were described above with the Hello message. *2hop_neighbor_chans* is constructed by processing *1hop_neighbor_chans* in the received Hello messages. For example, in a chain topology like A – B – C, B's one-hop neighbor C is, relatively, a two-hop neighbor to A. Node A, as B's one-hop neighbor, is treated as

a two-hop neighbor by itself, then A will set its Tx/Rx channels on the corresponding bit of *2hop_neighbor_chans*. But this does not cause any problem in getting a correct *twohop_range_chans* for the final *freeChanList* calculation. A Channel Usage Notification (CUN) message indicates that a new negotiated channel will be used by a node in a two-hop range. CUN message processing should set the channel in *twohop_range_chans*.

At the beginning of the Hello interval, a node saves the current *twohop_range_chans* to *previousChanUsage* and resets *twohop_range_chans* to refresh with the new channel usage information collected during the Hello interval. During the Hello interval, the node may not have a complete view of neighbor channel usage before receiving Hello messages from all neighbors. If a CREQ process is triggered at this time, current *twohop_range_chans* should be bitwise OR the *previousChanUsage* to provide the most recent and complete channel usage information to calculate *freeChanList.*

3.4 Queue-based transmitting channel switching

This section introduces a previously existing queue-based channel-switching proposal [Kya04] in Section 3.4.1, and the optimized mechanism proposed in this thesis is then described. After that, the detailed design and implementation in ns-2 is presented in Section 3.4.2.

3.4.1 Motivation and high-level design

The transmitting interface of a node switches to the receiving channel of its next hop node. Wireless interfaces need a few hundred microseconds for channel switching [Jeo05]. When a node has traffic to multiple next hop nodes on different channels, it is

expensive to switch the transmitting channel per packet. Queue-based switching was proposed in [Kya04]. It uses packet fairness in channel switching—transmitting maximum *BurstLength* queued packets on one channel before switching to another. The channel-switching cost is amortized among the maximum *BurstLength* packets. The time on one channel is bounded by *MaxSwitchTime.*

This thesis proposes an optimized queue-based channel switching using temporal fairness instead of packet fairness. The data transmitting interface works on each queue and channel for an identical period. Temporal fairness can increase overall throughput over packet fairness if flow rates are different for each channel [Awe03]. A node transmits on each channel for a maximum of QmaxTime seconds, to reduce switching overhead when the rate is high. However, the low boundary of transmitting time on a channel is dynamically adjusted with respect to the traffic load level. The transmitting interface does not need to stay on a channel if the queue has emptied before QmaxTime; therefore, channel switching occurs more frequently at low traffic loads or even per packet at the extreme. This adaptive process can keep packet delay low. In the meantime, high channel switching frequency is not detrimental because the switching cost is affordable at a low traffic load.

3.4.2 Queue-based transmitting channel switching implementation

Each Tx channel is associated with a packet queue. A packet to be transmitted on a channel is enqueued in the corresponding queue first, but not sent out until the Tx interface switches to this channel. When a node transmits on a channel for a predefined maximum time (QmaxTime) or the current queue is empty, the node gives other channels/queues a chance to be served. The queue that holds the oldest packet in terms of

arrival time is selected, and then the Tx interface switches to that channel. If no other queues have packets waiting to transmit, the node will reset the Qtimer (transmitting time on a channel/queue) and continue transmitting packets on the current channel. If all queues are empty, the routing agent changes to the IDLE state and it will switch to the corresponding channel when it receives a new outgoing packet.

Figure 3-8 shows the implementation of queue-based Tx channel switching in ns-2.

(1) The OLSR routing agent (RAgent) receives an outgoing packet. It finds the next hop node address from the routing table, and then finds the next hop node receiving channel from the extended OLSR *neighborTuple*. Finally, the RAgent forwards the packet to the LL of the ns-2 interface corresponding to the channel.

(2) The packet is queued and triggered to dequeue when this channel is activated by ActiveTx event (PT_AC). PT_AC resets the Qtimer to the current time and triggers to dequeue the packet. If the packets are transmitted on the control interface or data Rx interface (for example, ACK packet of TCP protocol) , they are directly dequeued and not restricted by the queue-based channel switching mechanism, because there is no channel switching on these interfaces which are working on the single channel.

(3) After the MAC has processed one packet, before the queue resumes to dequeue more packets, preResume function checks if this channel has been used for QmaxTime or if the queue is empty. If yes, the data Tx interface should switch to the channel with the oldest packet waiting in the queue. If no packet is waiting to be transmitted, the Tx interface is set to IDLE.

(4) When the packet is forwarded to the idle Tx interface (note*), or when the

interface switches from one channel to another channel, the switchTxChannel function is called, in which an ActiveTx event (PT_AC) is sent.

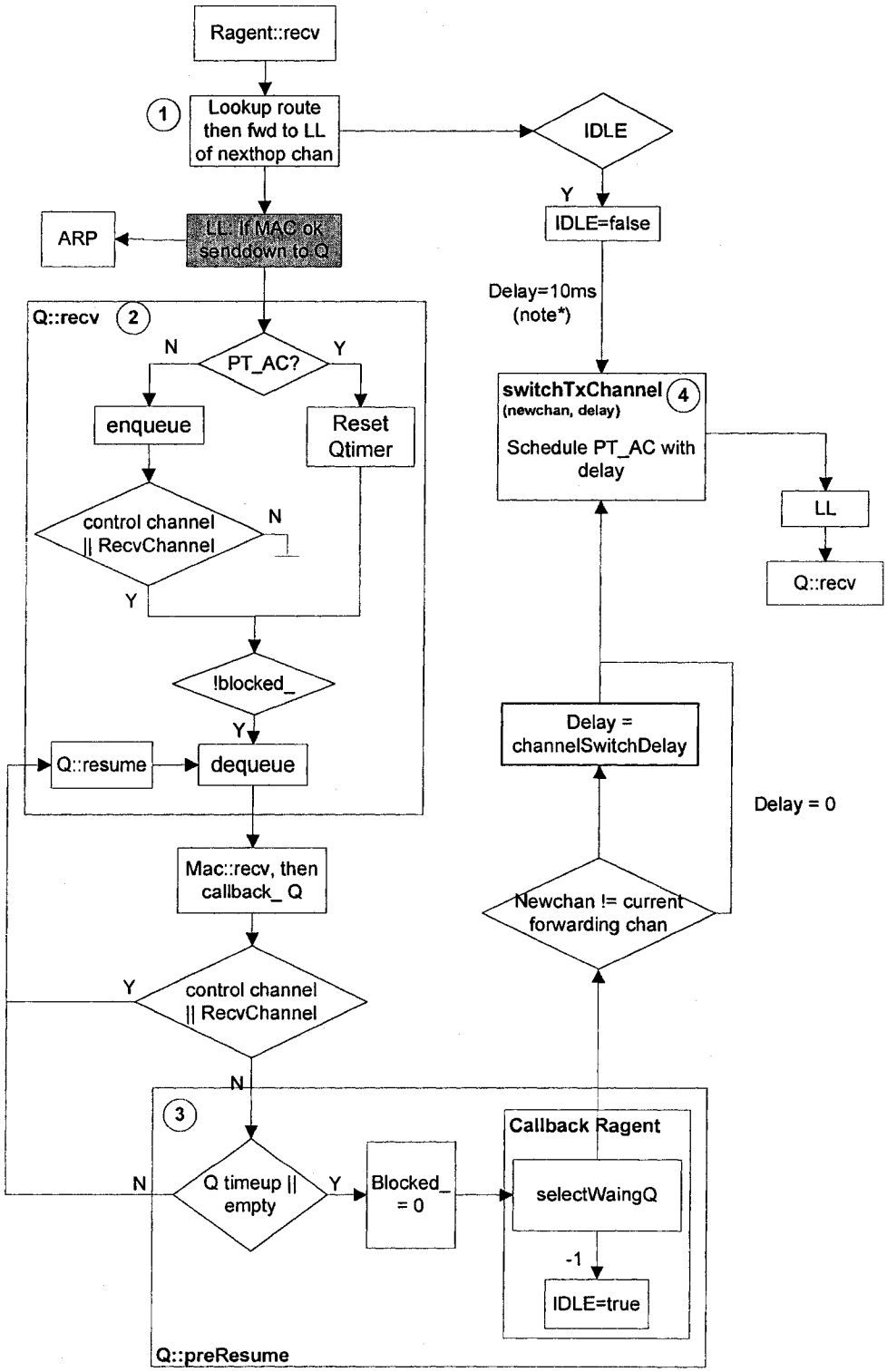


Figure 3-8. Flow Chart of Queue-based Tx Channel Switching

Implementation Note:

When data arrives at the idle Tx interface, the MAC address for the next hop node may need to be resolved before it can be forwarded to the IFq. The Address Resolution Process (ARP) takes about 2ms in the ns-2 simulation. If the PT_AC packet is sent to the queue before the data packets when the queue is still empty, the late-arriving data packet has no chance to be dequeued. Therefore the PT_AC packet is delayed for 10ms to give the data packet enough time to be queued. This delay will increase end-to-end delay, but it happens only when the interface is idle. Most of the time, there is no such delay in the packet dequeueing if the Tx channel switching is called by the queue preResume.

3.5 TCP performance improvement

A dedicated control interface does not only facilitate channel and routing control, but it can also be used to realize other flexible control mechanisms, such as energy consumption control in [Bah04a]. This thesis realizes a mechanism for improving TCP performance in wireless networks using the dedicated control interface.

The Transmission Control Protocol (TCP) [Moc81] was designed for wireline networks. Standard TCP cannot perform well in wireless networks [Ela02] because of the high bit error rate, long and varying delay, etc., in wireless networks. A new TCP for wireless networks has been researched for a long time; as a survey [Ela02] showed, most proposals invent a modified TCP, which cannot be widely deployed due to the inter-working problem. Developing a new TCP is not an objective of this thesis. However, TCP performance can also be improved by sending the ACK messages via the dedicated control interface. How the method works is explained in more detail below, and Section 4.8 presents the simulation results.

TCP traffic flow is composed of forwarding TCP data packets flow and backward ACK packets flow. The two flows intersect on the intermediate nodes. If ACKs are transmitted on the data interface like other regular TCP packets, as shown as ACK1 in Figure 3-9, there are two problems. First, ACKs from D to A and regular forwarding TCP packets from S to A are transmitted on the same channel and may collide on the Rx interface of the intermediate relay node A. Second, Tx data interface of node A needs to switch channels for forward TCP and backward ACK packets. Collision avoidance and Tx channel switching on the intermediate node A introduce more delay and decreased throughput.

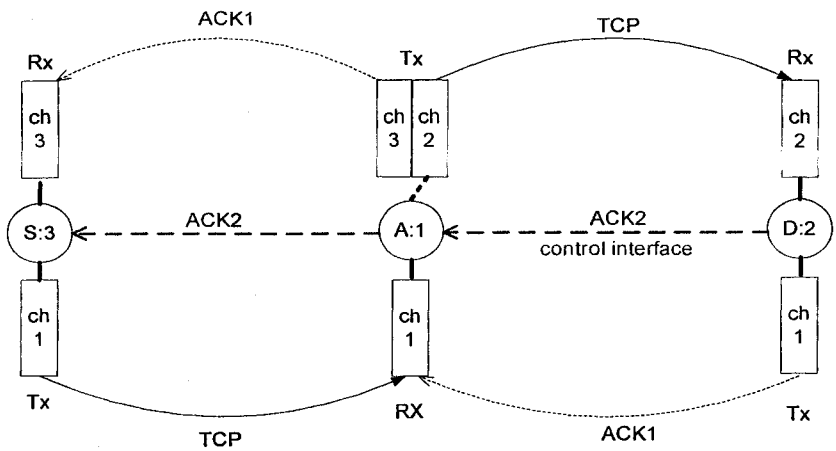


Figure 3-9. Sending TCP ACKs on Control Interface

For efficient TCP packet transmission in a multi-interface environment, the solution is to transfer the ACKs on the dedicated control interface as demonstrated with ACK2 in Figure 3-9. Although ACK packets do not take much bandwidth, compared to TCP data packets, if we mix them with other TCP data packets, collision and Tx channel switching will deteriorate efficient ACK delivery. As part of the TCP flow control mechanism, reliable transmission of ACK packets is critical to high TCP performance. Fast ACK transmission is assured on the lightly loaded common control channel. Thus

we can expect enhanced TCP performance. ACK packets are small (40 bytes) unicast packets, so this mechanism is also scalable, even if many TCP flows are using the common channel for their ACK transmissions.

Chapter 4 Simulations and Analysis

This chapter discusses the simulations of the proposal with the CMU extension to the Network Simulator ns-2.28 [NS2]. Performance is mainly compared between RCA and NCA multi-channel solutions. The single channel model is also simulated as a reference in some applicable scenarios. The single channel model includes the traditional Single Interface (SI) and Single data Channel (SC) with a dedicated control interface. It is fair to compare the NCA and RCA multi-channel solutions with SC because they all have a dedicated control interface.

First, NCA functionality is verified with a simple controlled topology in Section 4.1. Second, some design concerns regarding random-generated simulation topology are described in Section 4.2. Simulations with respect to various parameters then follow. Finally, the queue-based transmitting channel switching and TCP enhancement mechanisms are tested.

4.1 NCA functional verification

This section simulates NCA with some simple scenarios to evaluate the NCA algorithm and verify its implementation. At the same time, performances are compared between NCA, RCA and SC.

Unless otherwise stated, the simulations in this thesis use some common configurations: radio transmission range 250m; 802.11 bandwidth 2Mbps; CBR (Constant Bit Rate) traffic rate 200pkt/s; UDP packet size 512 bytes. This section

simulation is executed on a controlled topology (Figure 4-4): a 42-node grid topology with a 1000m x 1200m rectangle area; node distance 200m.

4.1.1 NCA algorithm basic functions

A four one-hop flows scenario (Figure 4-1) is used to verify the following functions of the NCA algorithm:

- Unique channel assignment in a two-hop range
- Channel reuse outside of interference area
- CREQ (Channel Request) retry

Four flows are started in sequence, one after another with a gap of 20ms. If flows start simultaneously, they may get the same channel by chance. It is similar to the time issue concern described in Section 3.3.2. Because channel collision resolution has not been realized in this research, the simulations try to prevent wrong channel assignment from occurring at the beginning of the channel assignment process.

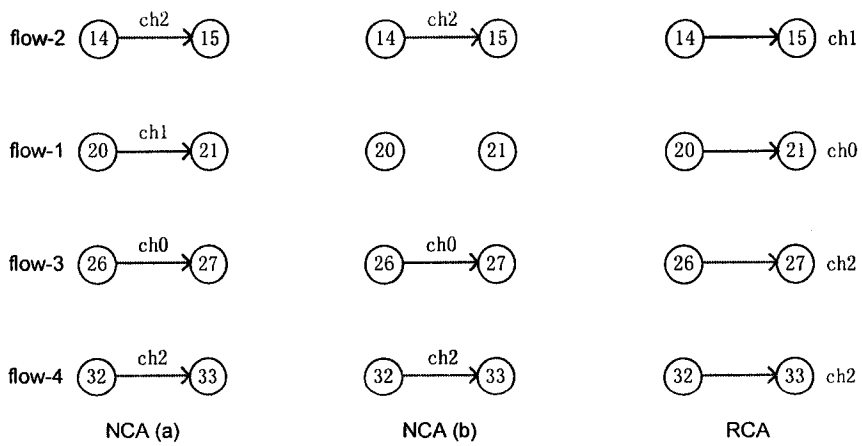


Figure 4-1. Single Hop Flows

There are a total of three channels (0/1/2) available in the scenario shown in Figure 4-1. As illustrated in the figure, NCA (a), flow-1 and 2 get ch_1 and ch_2

respectively; flow-3 stays on the default channel (ch_0) because all the free non-default channels have been used. The flow-4 nodes (32 & 33) are three hops away from the flow-2 nodes. Therefore, flow-4 can reuse ch_2 without causing interference.

All the nodes with traffic try to negotiate a non-default channel. If no other free channel is available, the communication will choose the default channel. The nodes will continue to retry channel negotiation to avoid possible collision on the default channel. If a channel is freed by other traffic, the traffic on the default channel will get it on channel negotiation retry. In this test case, as illustrated by NCA (b) of Figure 4-1, flow-1 is stopped later, and then ch_1 expires and is freed. The new channel usage of node 20 & 21 is notified by a Hello to the neighbors in the two-hop range. flow-3 on the default channel will switch to ch_1 in channel negotiation retry.

RCA assigns each node a channel solely depends on network topology. But when there are not enough channel resources, a node stays on a random channel conflicting with its neighbors. When a traffic pattern is applied on this topology, collision may occur. As depicted by RCA in Figure 4-1, nodes 27 and 33 are assigned the same channel by chance. Interference between flow-3 and flow-4 reduces the PDR (Packet Delivery Rate) to 83%. In a comparison of PDRs, NCA is 100% and SC is 62%.

4.1.2 Multi-hop flows

This section verifies NCA channel assignment in a scenario featuring three two-hop flows and a total of six available channels. In Figure 4-2, NCA intelligently assigns the channels according to traffic demand. The transmitter and receiver claim the negotiated channels in their respective two-hop interference area. Six transmissions of the three two-hop flows work on six different channels. NCA achieves 100% PDR. If the

interference model of RCA had been used, only the receiver claims its two-hop interference area, then the transmission from node 26 to 27 would use the same channel as node 15's transmitting channel. PDR is 78% because of the interference between nodes 15 and 27. This proves the two-hop interference and NCA interference model depicted in Section 3.3.1.

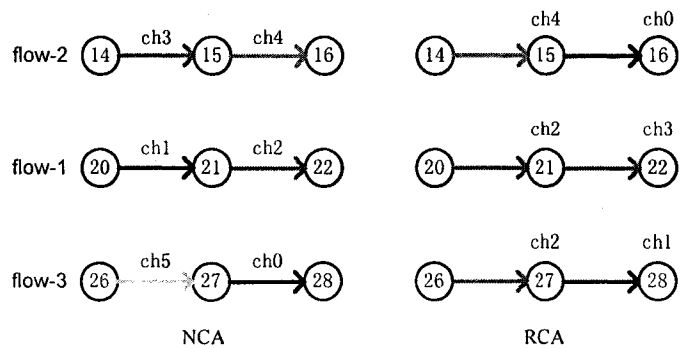


Figure 4-2. Two-hop Flows

RCA assigns channel to all nodes in the network. Not drawn in Figure 4-2, but as shown in Figure 4-4, the test is run on a 42-node grid topology. Six channels are not enough for each node to be assigned a unique channel in two-hop range area. RCA proactively assigns node channels without knowing which nodes will be used in traffic and should be assigned a unique channel with priority. For example, in Figure 4-2, nodes 21 and 27 are assigned the same red channel, while the orange channel is wasted on an idle node (not shown in the figure). The PDR of RCA is 78%. To compare, SC has a PDR of 23.7% because there is more interference.

4.1.3 Intersecting flows

When two flows intersect at a node, transmissions to the common node have to be on the same channel. For example, in Figure 4-3, nodes 20 and 21 negotiated to use blue channel for flow-1 transmission. Node 21 will broadcast an Imm_Hello after channel

negotiation. This message updates the *N_neighbor_channel* field of the corresponding *neighborTuple* in nodes 20, 15, 22 and 27. When flow-2 starts, node 15 checks *neighborTuple* and finds that the next hop node, 21, has a non-default receiving channel; it will use this channel for flow-2 transmission. The two flows collision at node 21. This conflict cannot be avoided. It is part of the future work to extend the routing protocol to find the non-intersecting route.

RCA has the same performance as NCA in this scenario, as long as nodes 22 and 27 are assigned a receiving channel other than blue. PDR comparison: NCA = 68%; RCA = 68%; SC = 34%

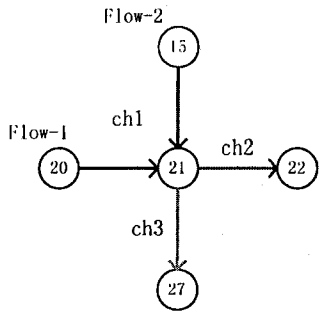


Figure 4-3. Intersecting Flows

4.1.4 A more complicated scenario

NCA is evaluated using a more complicated scenario in the controlled topology. The NCA implementation is verified by checking if channel assignment complies with the algorithm. The NCA algorithm is also evaluated to discover any possible design flaws. Seven flows in the grid topology, as shown in Figure 4-4, are selected with the following considerations in mind:

- Include flows with a varying number of hops, but avoid overly long routes for simplicity.
- No flow intersections where interference is unavoidable for both NCA and RCA.

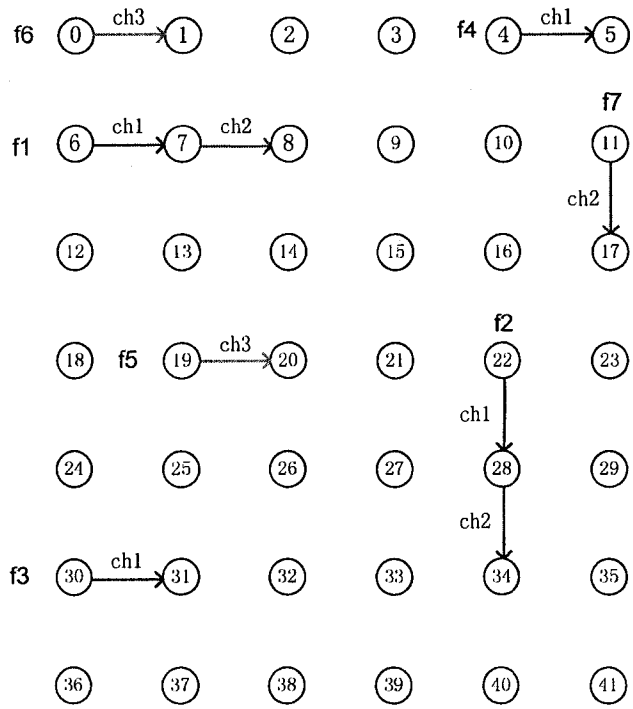


Figure 4-4. Multiple Flows in Controlled Topology

According to the NCA algorithm, this scenario needs a total of three channels for interference-free channel assignment. A simulation with a varying number of channels indicates 100% PDR when the number of channels is above three, while RCA reaches 100% PDR with nine channels. This result verifies the NCA implementation and proves its effectiveness in channel usage.

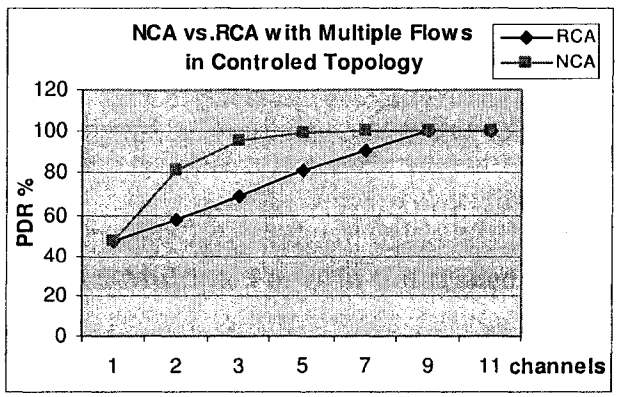


Figure 4-5. RCA vs. NCA in Channel Usage Efficiency

One NCA algorithm flaw was discovered during the simulation. Occasionally,

some nodes get the wrong channel assignment. For example, flow-5 once got ch_2 after negotiation, but it was being used by a transmission from nodes 7 to 8. Therefore the overall PDR decreased to 85% because of this channel conflict. The reason for this was that nodes 19 and 20 did not receive the broadcast CUN messages from nodes 7 and 8. Then they assumed ch_2 was still free. This discovery led to the introduction of the timing and jitter mechanism (Section 3.3.2) to reduce broadcast collision drops

4.2 Simulation topology design

Much of the multi-channel research uses high node density networks, such as 100 nodes in 500x500m² [So04a], 100 nodes in 670x670m² [Lee05], and 50 nodes in 500x500m² [Kya04]. Based on the 250m transmission range, most nodes are in a two-hop interference range (Ri=500m) of each other. Hence, a channel has little chance of being re-used.

Network node density can be defined as the node's average number of neighbors in a two-hop interference area. High node density requires many channel resources if the RCA scheme is used, because all nodes in the two-hop interference area need a unique channel. The current 802.11 technology provides a maximum of 12 available non-overlapping channels [Fux07]. Although NCA channel usage efficiency is more obvious in a high node density network, it is not fair for RCA to use an excessively high node density in performance comparison.

Network scale is another concern related to node density. Increasing network area size is beneficial for channel space reuse. At a fixed node density, the number of total nodes and the length of the route are also increased linearly with area size. Some ad hoc network research uses hundreds of nodes and long routes (up to nine hops in a chain

network used in [Kya04]). As Tschudin et al. pointed out in [Tsc05], "Current technology forms an ad hoc horizon at two to three hops and 10–20 nodes where the benefit from wireless multi-hop ad hoc networking virtually vanishes." Although a multi-channel and multi-interface network has good performance for longer flow, as proved in Section 4.3, the maximum route length of the simulation in this thesis is kept to around four, because a long route increases the possibility of flow intersection, which deteriorates multi-channel performance.

Based on the above two concerns, compared with topologies used in previous work, this thesis uses fewer (36) nodes and a bigger area size (800x800m²) with the default radio transmission range (250m). As shown in Figure 4-6, initially a 5x5 grid topology with a node at each crossing point was drawn for the simulation topology design. Each node has four one-hop neighbors. The shaded disk in Figure 4-6 is an example of a node's transmission area. A few random topologies are generated with a network parameter of 25 nodes in an 800x800m² area. This node density level keeps most nodes routable to each other, but it is not rare to see some unavailable routes between nodes. To form a connected network, 11 more nodes (36 nodes in total) are squeezed into the same-sized area.

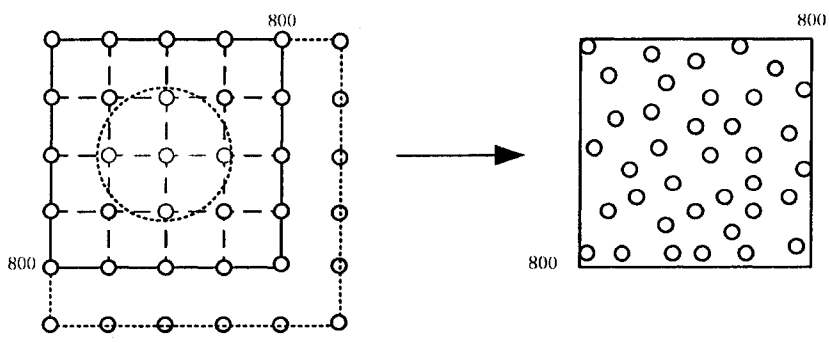


Figure 4-6. Simulation Topology Design

Ten random topologies are generated using ns-2 utility *setdest* with the designed

network parameter 36 nodes in 800x800m² area. The topologies are stationary because the NCA scheme has not realized a channel conflict-resolving mechanism for mobile node networks. The node density, the average number of two-hop range neighbors of a node, can be calculated with the formula

(Num_1hop_routes + Num_2hop_routes)/Total_num_Node

A route length statistic of one sample topology is shown in Table 4-1. In this topology, the node density derived from above formula is (242+312)/36 ≈ 15. Although the number of channels required by RCA for interference-free channel assignment is equal to the maximum number of nodes in a two-hop interference area, the average node density derived from above formula can be seemed as an estimate of channels resources required by RCA.

Table 4-1. Route Length Statistic of a Sample Topology

Route length (hops)	*1*	*2*	*3*	*4*	*5*
Number of routes	*242*	*312*	*370*	*254*	*82*

In summary, the simulation topologies of 36 nodes in 800x800m² used in this research provides a reasonable node density and area size. The number of nodes in the same interference area is about 15 on average, and then RCA's requirement on channel resources could almost be satisfied with existing wireless interface (e.g. 802.11a with 11 channels). A Channel has good chance to be reused because there are many (36-15=11) nodes are outside its interference area. Varying route lengths, ranging from 1 to 5, are available for performance evaluation with respect to flow length.

4.3 Flow length

For CBR UDP traffic, packet collision losses at any hop will contribute to PDR

decrease. If the channel assignment results in interference between transmissions of the hops along the flow, then the longer is the flow, the lower is the PDR.

NCA assigns channels only to nodes along the flow. A chain topology flow needs only a maximum of four channels to achieve a conflict-free channel assignment with NCA. As Figure 4-7 indicates, channels can be reused starting from the fifth hop transmission. For example, the first hop transmission AB is on ch1. Its interference area covers nodes C and D. Nodes E and F can reuse ch1 on the fifth hop.

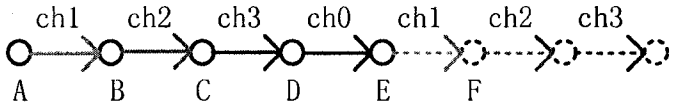


Figure 4-7. Channel Reuse in Chain Topology Flow

RCA need more channels for channel assignment to all nodes. When there are not enough channels, some nodes will stay on the conflicting channel. When these nodes are used for transmission, interference happens. The longer a flow, the more interference possibility between the hops in the same interference area.

In a randomly generated topology described in Section 4.1, 10 random flows are selected for a certain (1~5 hops) flow length and are simulated separately. A point in Figure 4-8 depicts the average simulation result of the 10 flows with the same flow length. Single flow is exercised in each simulation, to evaluate the isolated effect of flow length. Totally four channels are available in the network. MAC 802.11 bandwidth is set to 2Mbps. If using a higher bandwidth, such as 4Mbps, interference is not easily revealed from the performance. For example, the PDR of two-hop flows in SC with 4Mbps bandwidth is 94% which is not much difference to that of the interference-free NCA solution, while with 2Mbps bandwidth, the interference in SC is obviously shown by its low PDR (68%).

As demonstrated in Figure 4-8, a Single Channel (SC) network has interference on all hops along the flow on a common channel. Its PDR drops fastest as the flow length increases. NCA has 100% PDR, which means it generates interference-free channel assignment all the time. RCA's performance is in between NCA and SC.

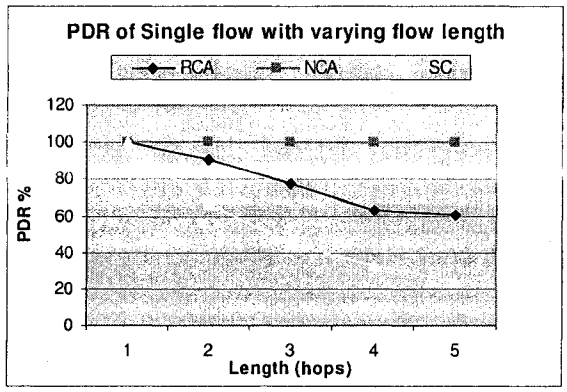


Figure 4-8. PDR Comparison with Varying Flow Lengths

4.4 Network node density

Node density is an important factor for RCA channel assignment. In higher node density networks, more nodes in an interference area can not be assigned free channel and then stay on random conflicting channels. Traffic using these nodes will experience interference. On the other hand, NCA only assigns channels to nodes with traffic, the node density does not have much impact on NCA.

Node density is dependent on the number of nodes and area size when the radio range is fixed. Based on the standard node density designed in Section 4.2, two random topologies with higher and lower node density are generated, as shown in Table 4-2. For each topology, 10 random two-hop flows are selected. Each flow is run separately.

Using RCA in the high-density network, 4 out of 10 flows have a PDR of 67%, which indicates that transmissions on two hops are on the same channel. In comparison, three flows in the medium density network have interference and no channel conflict is

discovered in a low-density network. On the contrary, NCA does not have interference in all three networks. The simulation result proves that NCA's advantage in channel usage efficiency is more obvious in high node density networks.

Table 4-2. Conflict Level with Varying Node Density

Node density		High 36 Nodes–400x400	Medium 36 Nodes–800x800	Low 18 Nodes–800x800
Flows_with_interference	RCA	*4/10*	*3/10*	*0/10*
/Total_flows	NCA	*0/10*	*0/10*	*0/10*

4.5 Number of flows and channels

In the above two sections, a single flow was run to check the effect of route length and node density. In this section, multiple flows in random topologies are used to examine the factors of number of flows and channels. Multiple flows require more channels; more channels reduce channel conflict among the flows. The number of flows and channels are two coupled factors that affect multi-channel performance.

One tuple of the number_of_flows (ranging from 1 to 6) and the number_of_channels (1, 3, 5, 7, 9 or 11) is a simulation point and is used in 10 randomly generated topologies. The difference of simulation results in different topologies mainly comes from the randomness of the traffic pattern, such as flow length and inter-flow relation. The average of the 10 random topologies is used as the result for each simulation point. The simulation is performed using various solutions—RCA, NCA and single channel, and is analyzed from the perspectives of number of channels and number of flows.

4.5.1 Number of flows

In this section, performance with regard to the number of flows will be analyzed. The purpose of increasing the number of flows is to increase the complexity of the traffic pattern. An ideal multi-channel solution is to realize interference-free transmissions in any scenario. The traffic pattern refers to the number of traffic flows, flow length and inter-flow position relationship. A connection pattern, composed of a certain flow defined by source and destination nodes, may have different traffic patterns in randomly generated topologies. Although traffic pattern complexity is not easily measured based on its multiple heterogeneous characteristics, and it is out of control in a randomly generated topology, the total number of (one-hop) transmissions of all flows is a simple index of the complexity.

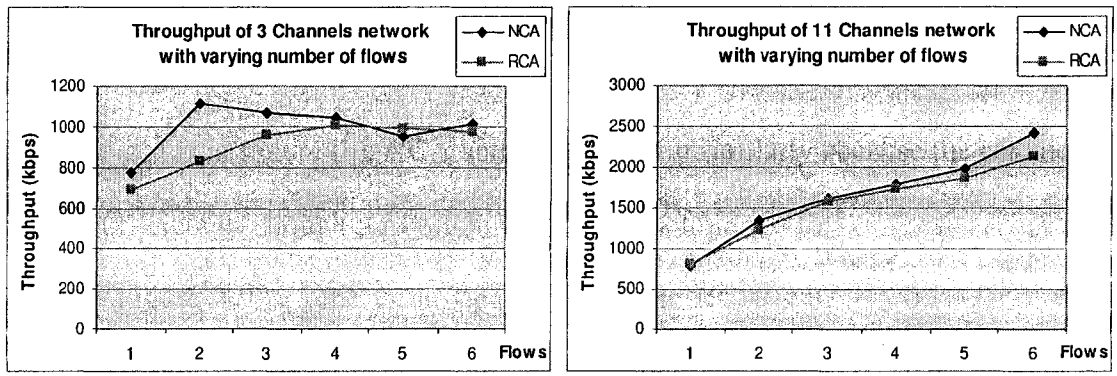
Six flows are first formed with six pairs of nodes randomly selected from all nodes (numbered 0~35) as a pool of flows. It is a known scenario that multi-channel solutions, both NCA and RCA, cannot avoid interference at the intersecting node. The connection pattern cannot be controlled in random topologies but, at least, intersection at the source or destination node can be avoided by not using the same node for any two flows. To make sure that the number of one-hop transmissions increases with the number of flows from 1 to 6, one flow is initially selected from the pool of six flows; each time the number of flow increases by 1, a new connection pattern is formed by taking one more flow from the pool of six flows and adding to the previous connection pattern. For example, the two-flow connection pattern includes flows 9→10 and 18→4; the three-flow connection pattern includes flows 9→10, 18→4 and 16→5.

The number of channels and the number of flows are two coupled factors

affecting multi-channel performance. The performance with regard to the number of flows is different with a different number of total channels. This section analyzes the impact of the number of flows when the number of channels is fixed.

As Figure 4-9 shows, when the number of channels is small (e.g. 3) in Figure (a) or large (e.g. 11) in Figure (b), NCA and RCA perform similarly because the channels are either insufficient or enough for both multi-channel solutions.

In Figure 4-9 (a), the throughput of 1-flow in RCA is 87% (700/800) of that in NCA. This result matches with the result in Section 4.3, where the average PDR of a single flow of 1~3 hops is about 88% in a four-channel network. NCA can make better use of channel resources and achieve much higher throughput than RCA for 2-flow connection pattern. When the number of flows is four or more, both NCA and RCA reach the saturated network capacity of three channels.



(a) (b)

Figure 4-9. Throughput with Respect to Number of Flows with 3 or 11 Channels

In Figure 4-9 (b), network capacity is high with 11 channels; the throughput keeps increasing even at 6 flows. As described in Section 4.2, the simulation topology has a node density of 15 nodes in the interference area. RCA, with 11 channels, can make almost interference-free channel assignment. Therefore its performance is close to NCA's. But with more transmissions, the possibility of channel collision in RCA is

higher. This is why NCA shows a major improvement over RCA at six flows.

Figure 4-10 shows the result for a medium number of channels (e.g. 7). NCA's performance improvement over RCA is obvious for all multi-flow connection patterns. It proves NCA's efficient channel usage when there are not enough channels for RCA.

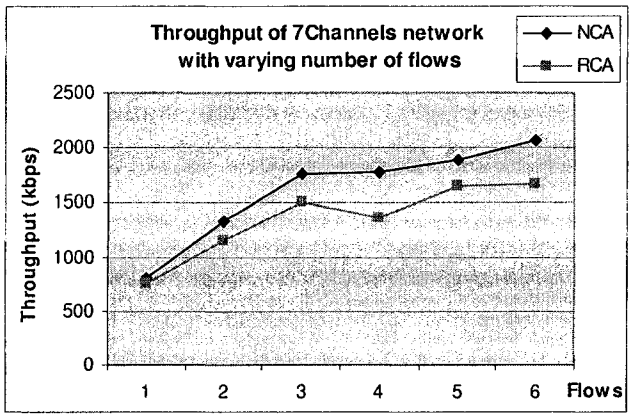


Figure 4-10. Throughput with Respect to Number of Flows with 7 Channels

After examining the impact of the number of flows with a different number of channels separately, the average performance of the different number of channels (3, 5, 7, 9 and 11) is viewed with respect to the varying number of flows. Figure 4-11 depicts the throughput comparison for NCA, RCA, Single Channel (SC) and traditional Single Interface (SI).

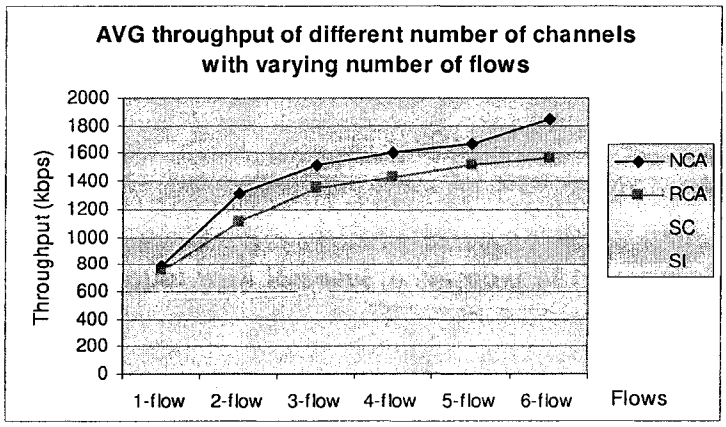


Figure 4-11 . Average Throughput with Respect to Varying Number of Flows

Overall, the NCA and RCA multi-channel solutions show a significant improvement over the SC and SI single channel. As the number of flows increases, SC and SI have no throughput increase because they are already at saturated load level, but NCA and RCA throughput increases continually because of the higher network capacity in a multi-channel network. NCA's advantage over RCA is its efficiency regarding channel usage. The average improvement from RCA to NCA seems constant with respect to the varying number of flows.

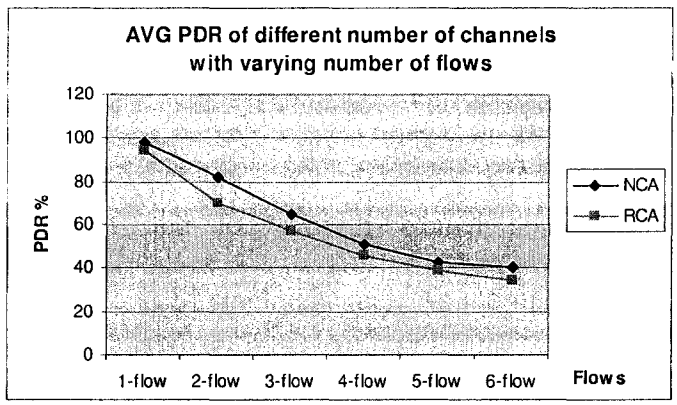


Figure 4-12 . Average PDR with Respect to Varying Number of Flows

A more than three-fold average throughput improvement is monitored in the simulation as demonstrated in Figure 4-11. However the increase is not linear with the number of flows, because the PDR drops at the same time (Figure 4-12). The reason for this is the higher possibility of the formation of bottlenecks, which could be a flow intersection node or an interference area shared by many flows.

The Single Channel (SC) solution uses an additional dedicated control interface compared with the traditional Single Interface (SI) solution. The slight improvement from SI to SC is the routing control traffic offloaded by the dedicated control interface. In the stationary simulation topologies, the dedicated control interface does not show an advantage regarding routing control. Neither SC nor SI drops packets due to unavailable

routes. But SC has the potential to improve routing control performance in faster mobile networks with the help of a dedicated control interface.

4.5.2 Number of channels

In last section, multi-channel solutions are compared with the varying number of flows while the number of channels is fixed for each simulation. This section checks the impact of varying number of channels when the number of flows is fixed.

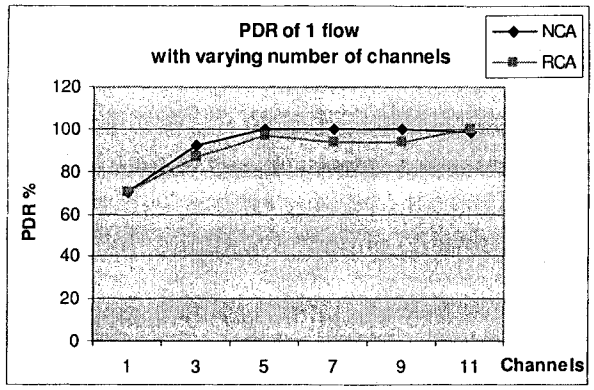
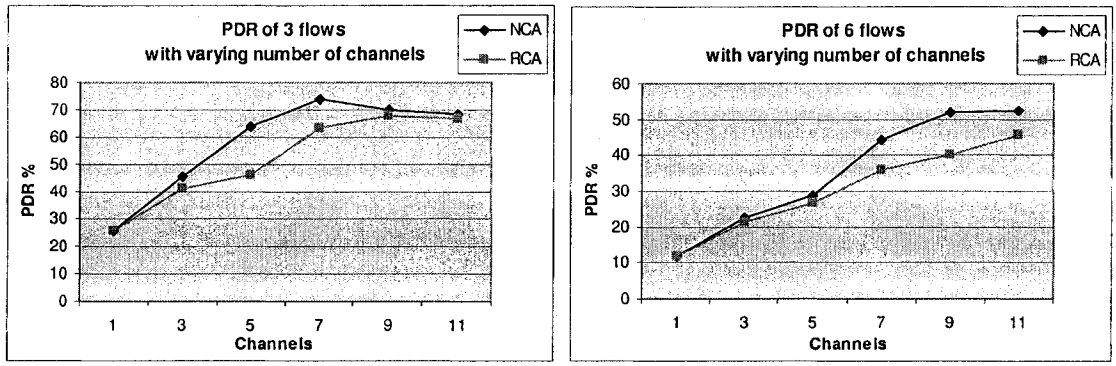


Figure 4-13. Single Flow PDR with Varying Number of Channels

Figure 4-13 shows the PDR of the single flow traffic pattern with respect to the varying number of channels. NCA performs better than RCA but there is not much difference. NCA has assured interference free channel assignment (100% PDR) after the number of channels is greater than 5 (actually 4 as Figure 4-7 indicated). Interference-free RCA need a lot more channels. Channel collision still exists even with 9 channels.

Simulation results with 3 and 6 flows are analyzed. In Figure 4-14, two simulation sets indicate that, NCA shows its advantage over RCA if the number of channels is not too small, which also becomes insufficient for NCA, or too large, that is enough for RCA as well. In simulation with 6 flows traffic (Figure 4-14(b)), NCA with 11 channels can improve network capacity over single channel by almost 5 times, i.e., PDR increases

from 11% at 1-chan to 52% at 11-chan.



(a) (b)

Figure 4-14. PDR of 3 Flows and 6 Flows with Varying Number of Channels

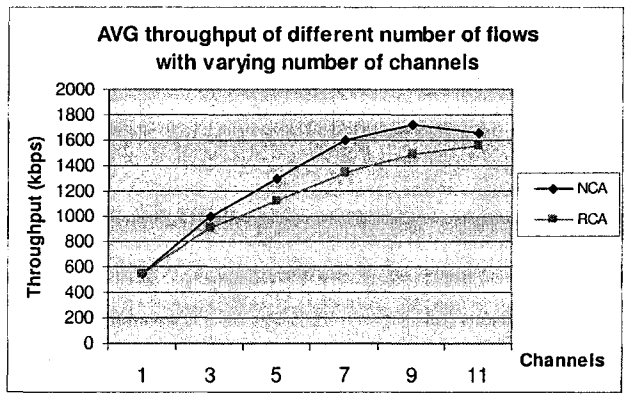


Figure 4-15. Throughput with Varying Number of Channels

Each connection pattern with different number (1-6) of flows is exercised individually with varying number of channels. Take the average throughput of all connection patterns; Figure 4-15 compares NCA and RCA performance with respect to the number of channels. Multi-channel (NCA and RCA) increases network capacity (throughput) with more channels. NCA and RCA achieve the similar maximum throughput when there are enough channels, but NCA use less channels (7 channels) than RCA (11 channels). Overall NCA performs better than RCA especially with medium number of channels. When the number of channels is very small or large, both multi-channel solutions have insufficient or enough channels, then they have similar performance.

4.6 Traffic load rate

In the previous sections, the flow rate was fixed at 200 packets/second. Varying the flow rate impacts each solution differently. The simulation point of three flows with five channels was selected to be run with a varying flow rate. This point shows the obvious performance difference between NCA and RCA in. The simulation was run on 10 random topologies and the average is shown in Figure 4-16.

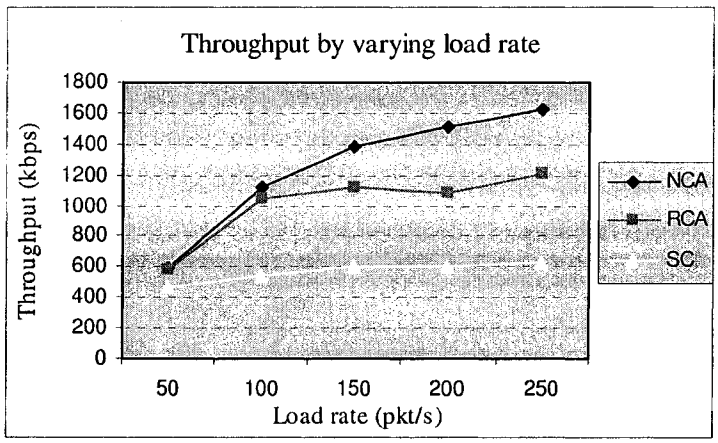


Figure 4-16. Throughput with Respect to Traffic Load Rate

As illustrated in Figure 4-16, with higher flow rates, NCA throughput keeps increasing because of its interference-free channel assignment, while RCA reaches maximum capacity quickly. The solution using effective channel assignment has higher throughput. At low flow rates about 50 pkt/s, NCA, RCA and even SC do not show much of a difference because the packets can still go through even in circumstances where there is interference. The interference is revealed when flow rates are high. NCA, RCA and SC have different saturated load levels.

4.7 Confidence interval

In Section 4.5 and 4.6, one simulation point is the average result of simulation on

ten random topologies. To make sure the results are not biased due to randomness of topology and the simulation tool, this section investigates the confidence interval (CI) of the simulation results. A 90% CI for an average can be derived from equation (2) below [Law00].

$$\textit{90\% Confidence Interval of n sample data} = \textit{average} \pm 1.83 * \sqrt{\frac{\sigma^2}{n}} \quad \ldots\ldots\ldots\ldots\ldots (2)$$

Where σ is the standard deviation and n is the number of observations or experiments. For the thesis, n = 10. Figure 4-17 depicts the 90% CI for the simulation of three channel assignment schemes described in Section 4.6 for 3 flows using 5 channels.

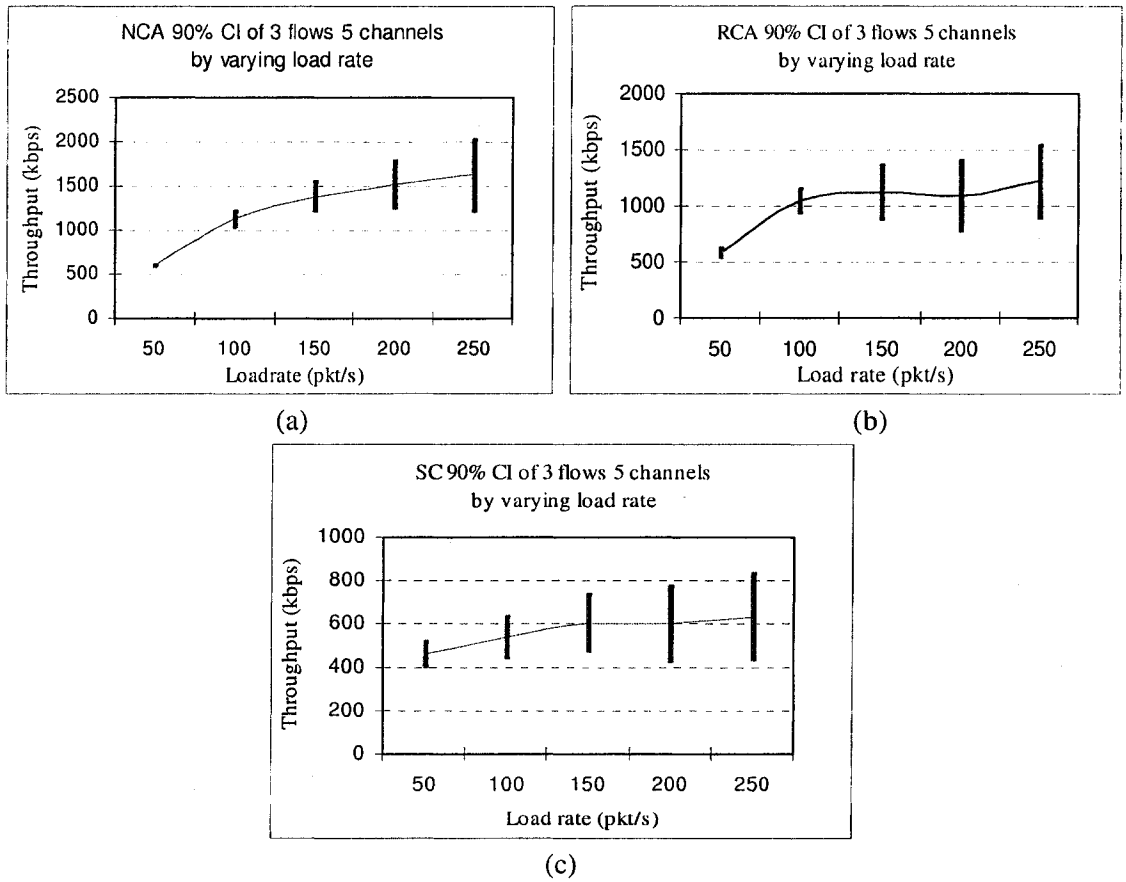


Figure 4-17. Confidence Interval with Varying Load Rate

The variance is mainly due to the route randomness of the traffic pattern when a connection pattern is put in a random topology. There is also a slight variance if a

simulation is run several times on a same topology. This variance comes from the randomness in OLSR routing and the RCA/NCA algorithms. It is much smaller in comparison to the variance due to traffic pattern, and is negligible. From the confidence intervals shown in Figure 4-17, NCA is better than RCA which in turn is better than SC. The confidence intervals for other scenarios (different number of flows and different number of channels) are similar to that depicted in 4-17.

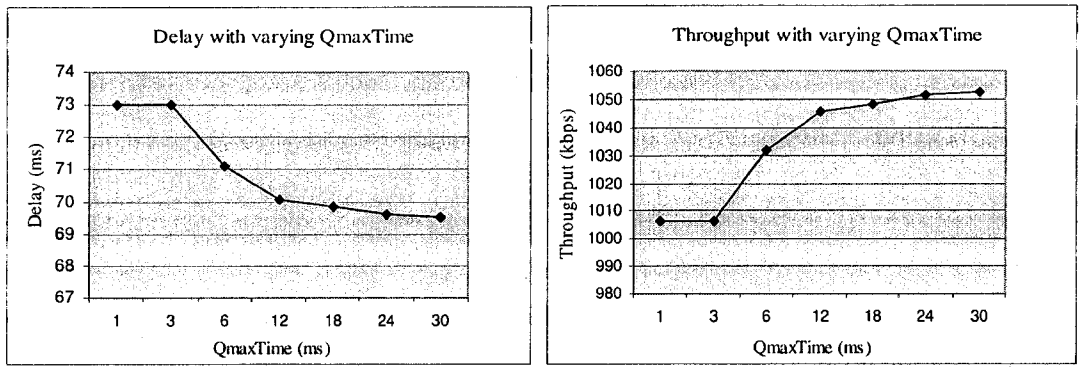
4.8 Queue-based transmitting channel switching

This section investigates the queue-based Tx channel switching performance. If a node transmits on k different channels evenly in time, the average one-hop end-to-end (E-E) packet delay can be calculated with the below equation.

$$E\text{-}E\ Delay = k * Queue\_size * (Transmission\_delay + Switch\_delay/n) \quad .....(3)$$

Where n is the number of packets transmitted on one channel during *QmaxTime*. Traffic flow on each channel is at a saturated load rate so that the queue associated to the channel is always full. *Transmission_delay* is the one hop transmission time including MAC layer overhead. It is measured by running a one-hop transmission at a saturated load rate (300 pkt/s, PDR=88%, throughput=1055kbps). The packet end-to-end delay is 34.6ms as measured, the queue size is 10, and so the *Transmission_delay* is 3.46ms.

To evaluate the queue-based Tx channel switching performance, a simple scenario of switching between two channels is sufficient. One source node S has two one-hop flows to D1 and D2, respectively (i.e., D1← S →D2). The channel switching delay is 200 microseconds. The load rate is 300 pkt/s for each flow and the queue size is 10 for each channel.



(a) (b)

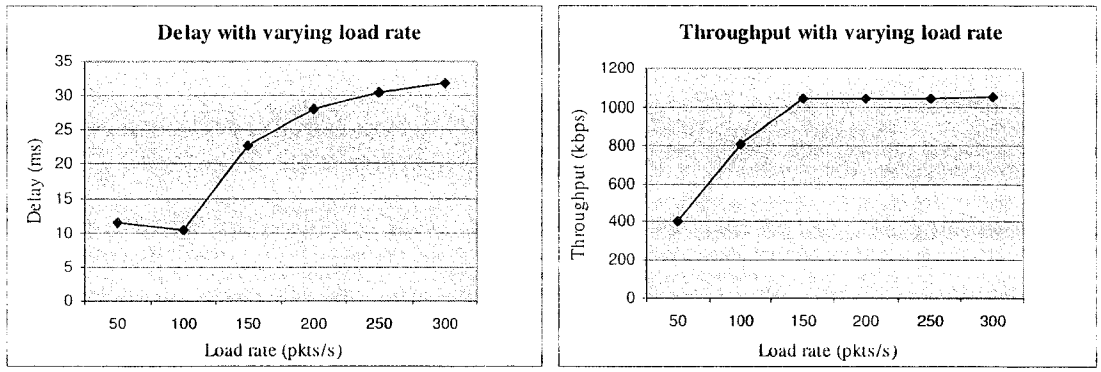
Figure 4-18. Performance with Respect to QmaxTime

As Figure 4-18(a) shows, when QmaxTime is small, the node switches channel per packet (i.e. n =1). The delay is 73ms, which is close to the expected value (73.2ms) calculated from equation (3). The value of "n" increases with the QmaxTime, the channel *Switch_delay* is amortized by more packets sent during every channel switch cycle. Then the end-to-end delay decreases and its low boundary is 69.2ms when n is very large. The simulation result proves that the queue-based Tx channel switch can correctly reduce packet delay and increase throughput (shown in Figure 4-18(b)) at the saturated load level.

Queue size is the more important factor than switching delay in affecting packet end-to-end delay because *Transmission_delay (3.46ms)* is much bigger than *Switch_delay (0.2ms)*. By decreasing the queue size to half (5), the delay is about half the value shown in Figure 4-18 (a). The queue size of 10 is also used in the literature [Kya04]. This is a reasonable value to limit end-to-end delay under 150ms as ITU [G114] recommended. Queue size does not affect throughput significantly.

Different to the proposal in [Kya04], the queue-based channel switch in this thesis is adaptive to load rate. A transmitting interface switches to serve packets in other queues if the current queue is empty, thus the packet delay is controlled as low as possible while

the frequent channel switching does not deteriorate throughput. Figure 4-19 depicts the simulation result with varying load rate. Qmaxtime is 20ms; queue size is 5. When load rate is lower than 100pkt/s, average delay with per packet switch is about 10ms. It is close to the theoretical value of 2*(Transmission_delay + Switch_delay) = 2x(3.46+0.2) = 7ms.



(a) (b)

Figure 4-19. Performance with Respect to Load Rate

4.9 TCP performance improvement

To check the new TCP transmission scheme in which sending ACKs on the control interface, it is compared with two other schemes— sending ACKs as normal data packet on the data interface and using the traditional Single Interface (single channel). One Tahoe TCP [NSDoc] flow is simulated on 10 random topologies (s0~s9) where the flow has varying route lengths. There are some other TCP variants, such as Reno, Vegas, Sack, and etc. [NSDoc], which adopt different congestion and error control techniques. They may perform differently in wireless network, but the mechanism of sending ACK on the control interface can help improve all their performance because of separating ACK and TCP on different interfaces and channels.

The flow is one hop in s1, s2 and s7; two hops in s0, s3, s4, s5 and s9; three hops

in s8; and four hops in s6. Only single flow is run to avoid inter-flow impact. NCA with 11 channels is used.

The PDR is always 100% because TCP is a reliable transmission protocol. In the throughput (Figure 4-20) and E-E delay (Figure 4-21) plots, sending ACK on the control interface (ACK on Ctrl) significantly improves the performance in comparison with Single Interface(SI) and sending ACKs on the data interface (ACK on Data). For example in the scenes of a two-hop flow, "ACK on Ctrl" increases throughput over the other two schemes by a factor of 3 or 1.5, and decreases E-E delay by 2~4 times.

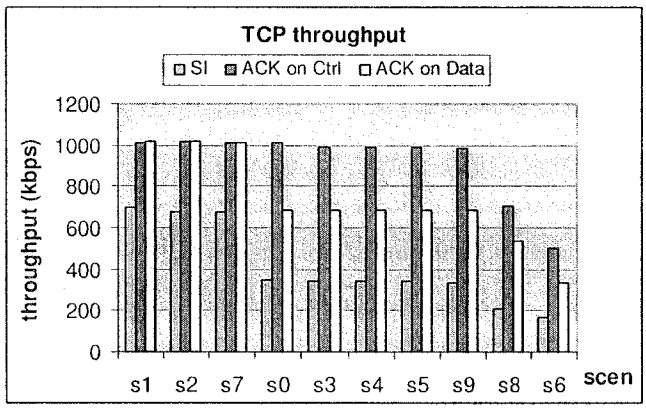


Figure 4-20. TCP Throughput Comparison for Three Schemes

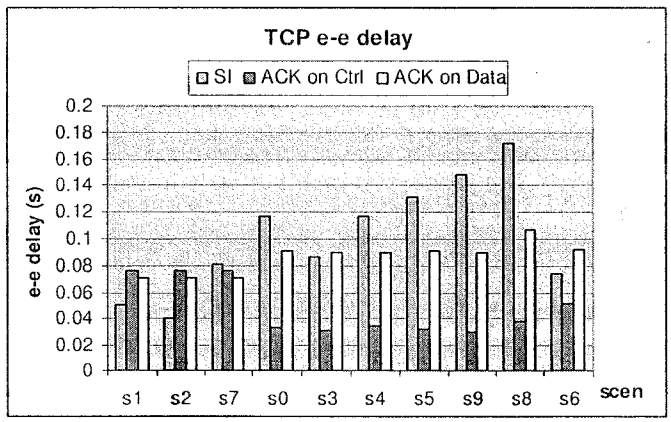


Figure 4-21. TCP E-E Delay Comparison for Three Schemes

In one-hop flow scenes (s1, s2 & s7), "ACK on Ctrl" and "ACK on Data" have the same performance. With "ACK on Data" scheme, backward ACK and forward TCP

packets of one-hop flow are separated on two sets of data interfaces. It is similar to "ACK on Ctrl" that no ACK and TCP packet collision. For multi-hop flows, however, "ACK on Ctrl" performs much better than "ACK on Data" as explained in Section 3.5.

It is worthy to explain why "ACK on Ctrl" has a longer E-E delay in one-hop scenes than in two-hop scenes. At the beginning of simulation, the source node sends several TCP packets, quick ACK reply (less than 2ms in one-hop flow) in one-hop scenes triggers more TCP packets and the queue is fill up. Later on, each TCP packet delivery triggers a backward ACK packet, and the ACK packet arriving triggers one more TCP packet queued up at the source node, therefore the transmitting queue at the source node is always full. The E-E delay is mainly the queue delay (20 packets). Although the ACK packet (40byte) is fast in transmission back to source node, the TCP delivery rate is bounded by slower data packet (512byte) transmission. It is measured as 3.93ms/packet in simulation. Then the E-E delay is almost 80ms (3.93ms/pkt x 20pkts). While in the scenes of multi-hop TCP flow, ACK on the control interface takes longer to be delivered to the source node – 4.03ms in two-hop scenes and 5.63ms in three-hop scene (s8). The hop by hop TCP data packet transmission time (3.93ms) is shorter than ACK arrival period (4.03ms for two-hop flow), therefore the TCP data packet delivery rate is limited by the slower ACK arrival rate; then the throughput decreases with route length increase. But the node transmitting queue is not filled up (8~10 packets in simulation) because of the slower ACK arrival rate, then the two-hop flow E-E delay is even shorter than the one-hop flow.

Simulation plots in this section depict the average result of eight simulation runs. There is no simulation result variance monitored for one- or two-hop flows, but s8 and s6

(three- or four-hop route length) have observed dynamic result, especially with the "ACK on Data" scheme. When an ACK is sent back on the data interface, it may collide with the forwarding TCP packet on the Rx interface of the previous node. The back-off collision avoidance mechanism introduces randomness into the simulation results. When the route is long, the resulting variance is larger.

Chapter 5 Conclusions and Future Work

This thesis proposed an architecture using a dedicated control interface in multi-channel, multi-interface wireless ad hoc networks. This architecture enables simple and efficient multi-channel control. The simulation indicated that a dedicated control interface and the proposed multi-channel protocols can greatly improve performance over single channel wireless networks. The trade-off is higher power consumption by multiple interfaces. The protocol overhead is small enough to be neglected. In RCA, only 5 extra bytes are added in the Hello message generated every 2 seconds. In NCA, some negotiation messages are trigged only on traffic demand. The computational overhead is also small because the channel control related computation occurs infrequently (in seconds) on the network layer.

Two network layer multi-channel solutions were designed. RCA enhances the previous OLSR-MC through a more accurate interference model, while NCA is an innovative network layer approach. NCA is more efficient regarding channel resource usage. The simulation results demonstrate that NCA achieves considerable performance improvement over RCA with fewer channel resources. The impact of different factors on performance was reported. It was discovered that NCA works much better than RCA when the flow length increases or at high flow rates. Compared with SC, NCA achieved up to five times throughput improvement in simulation (Figure 4-14(b)).

5.1 Future work

This thesis has proposed a good performance multi-interface multi-channel solution; on the other hand, there are still many research areas that can be conducted. The following sections discuss three problems. First, a necessary assumption for dedicated control interface is the matching transmission ranges between the control and data interfaces. This might be a problem in the real world. Second, the routing protocol needs to be optimized to avoid flow intersection on which interference is unavoidable. Third, NCA still cannot generate a totally interference-free channel assignment all the time. It is difficult to determine all the interference scenarios at the network layer, as shown in Figure 3-3. We need a complementary mechanism to detect and resolve the incorrect channel assignment or channel collision occurring in mobile networks.

5.1.1 Radio range mismatch

This thesis assumes the same control and data radio (interface) transmission range. This is not likely to be true all the time in the real world. If the control interface has a shorter transmission range, a bigger data radio range will cause interference that is unexpected by the control interface. On the other hand, a shorter data radio transmission range causes failure of packet delivery on the route provided by the control interface. The experiment in [Bah04a] showed a greatly degraded performance when the ranges of the control and data interfaces are mismatched.

Radio range depends on many factors, such as PHY specification, data rate, frequency band, transmission power, the environment, and etc. With more non-overlapping channels offered by 802.11a, it could be suitable for a data interface, On the

other hand, 802.11b could be used for the control interface. 802.11a radio ranges from 25m to 75m, and 802.11b radio has a range of 35m to 100m [802.11wiki]. For the multiple interfaces of a mobile node, radio specification can be initially adjusted in the factory to achieve a similar transmission range.

Besides initial static radio range adjustment, an automatic range adjustment mechanism is proposed to combat the mismatch. As illustrated in Figure 5-1, each node maintains a one-hop neighbor topology on the data interface at a low frequency, for example with a Hello-like message every minute instead of two seconds, as in the OLSR Hello on the control interface. A node compares the neighbor set it discovers on the data and control interfaces. If the data interface discovers fewer neighbors, the node needs to increase its data radio range. For example, with data radio range Rd, node A cannot see node B as a neighbor appearing in the control interface's neighborSet. Node A can then increase Rd. On the other hand, node A with data radio range R_D sees neighbor C, which the control interface cannot see, and then node A needs to reduce R_D.

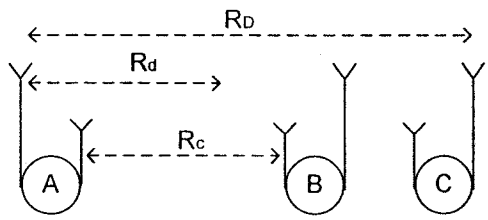


Figure 5-1. Radio Range Automatic Adjustment

The radio transmission range is related to the data rate. An example of range and data rate relation between 802.11a and 802.11b is given in Figure 5-2. A radio has a larger transmission range at a lower data rate.

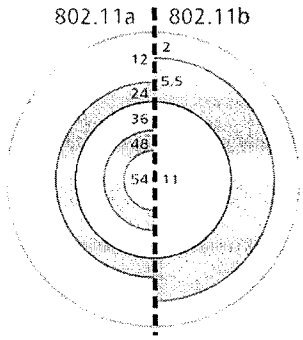


Figure 5-2. Range / Data Rate Differences Between 802.11a & 802.11b [802.11aWP]

All discussion in this thesis is based on a fixed data rate. In a multi-rate wireless network, the data interface range is also changing. A fixed range control interface cannot make a correct routing and topology-based channel assignment. The Data Rate Adaptive Channel Assignment (DR-CA) algorithm [Nir06] was recently reported for multi-channel multi-rate wireless networks. A future work is to investigate if a dedicated control interface can help the multi-rate wireless network.

5.1.2 Disjoint path routing protocol

OLSR counts hop numbers to find the shortest path. Two traffic flows may have intersecting routes and the joint node becomes the bottleneck. Transmissions to the joint node have to be on the same channel and interference between them is unavoidable (Figure 5-3-a). The multi-channel performance is degraded, as shown in Section 4.1.3. Disjoint routes (Figure 5-3-b) can transmit flows simultaneously on different channels.

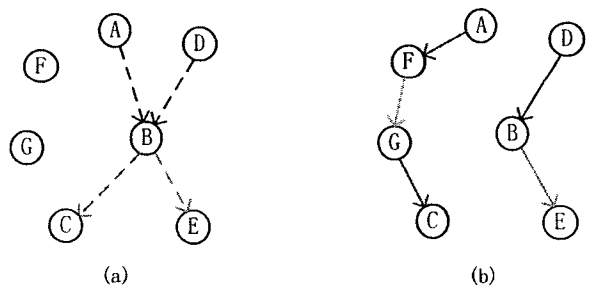


Figure 5-3. Routing Protocol Needs to Avoid Intersecting Nodes

Multi-Channel Routing [Kya05] is a multi-constraint routing protocol to discover an optimized route in multi-channel networks. The metrics of channel diversity and channel switching cost are considered to reduce interference and avoid disjoint routes. The routing protocol and channel assignment are decoupled in this thesis. It is easier to extend the routing protocol to realize disjoint routing. A node can monitor the number of incoming flows. This number can then be translated to the OLSR *willingness* [OLSR] parameter to affect route preference.

5.1.3 Resolving channel collision

Although a more accurate interference model is introduced in the network layer multi-channel solutions, the network layer definition on channel collision cannot capture all the collision scenarios on the MAC layer. Transmissions interfere with each other due to incorrect channel assignment. Node mobility is another reason for channel collision. After the initial channel negotiation is complete, a node may move to a new neighborhood and have channel collision with its new neighbors.

Proactive RCA can partially detect and resolve channel collision caused by node mobility. A node maintains its receiving channel uniqueness through Hello messages. NCA is an on-demand channel assignment algorithm. Collision detection is a proactive maintenance process. It is a challenge to realize them at the same time on the network layer. Cross-layer design [Mar04] is a prominent approach for detecting collision locally at the lower layer close to where it occurs. A possible solution is to monitor the behavior of the interface queue between the LL layer and the MAC layer. Abnormal queue overflow is an indicator of possible channel conflict, and then a new CREQ process can be triggered to resolve it.

References

[802.11] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ANSI/IEEE Std. 802.11, 1999 Edition (R2003).

[802.11aWP] Proxim, "802.11a White Paper," http://www.proxim.com/learn/library/whitepapers/80211a.pdf, Last accessed in Mar. 2007.

[802.11wiki] Wikipedia, "IEEE 802.11," http://en.wikipedia.org/wiki/IEEE_802.11 #Summary, Last accessed in Mar. 2007.

[AODV] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," *IETF RFC 3561*, July 2003.

[Awe04] B. Awerbuch, D. Holmer, and H. Rubens, "High Throughput Route Selection in Multi-Rate Ad Hoc Wireless Networks," *Proc. of the Wireless On-demand Network Systems (WONS)*, Jan. 2004, pp. 251-268.

[Bah04a] P. Bahl, A. Adya, J. Padhye, and A. Wolman, "Reconsidering Wireless Systems with Multiple Radios," *ACM Computing Communication Review*, 34(5), Oct. 2004, pp. 39-46.

[Bah04b] V. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks" *Proc. of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004, pp. 216-230.

[Den04] J. Deng, B. Liang, and P.Varshney, "Tuning the carrier sensing range of IEEE 802.11 MAC," *Proc. of the Global Telecommunications Conference*

(GLOBECOM), 2004, pp. 2987- 2991.

[Dra04] R. Draves. J. Padhye, and B. Zill; "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," *Proc. of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004, pp. 114-128.

[Ela02] H. Elaarag, "Improving TCP performance over mobile networks," *ACM Computing Surveys (CSUR)*, 34(3), Sep. 2002, pp. 357-374.

[Fux07] P. Fuxjager, D. Valerio and F. Ricciato, "The Myth of Non-Overlapping Channels: Interference measurements in 802.11," *Proc. of the 4th Annual Conference on Wireless On demand Network Systems and Services (WONS)*, Jan. 2007, pp. 1-8.

[G114] International Telecommunication Union (ITU) Recommendation G.114, "*One-way Transmission Time,*" http://www.cisco.com/en/US/tech/tk652/tk698/technologies_white_paper09186a00800a8993.shtml#standarfordelaylimits, Last accessed in Mar. 2007.

[Gup00] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, 46(2), Mar. 2000, pp.388-404.

[Jeo05] J. Mo, H. W. So and J. Walrand, "Comparison of Multi-Channel MAC Protocols," *Proc. of the International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Oct. 2005, pp. 209-218.

[Kya04] P. Kyasanur and N.Vaidya, *Routing in Multi-Channel Multi-Interface Ad Hoc Wireless Networks*, Tech. Rep., Dept. of Computer Science, University of Illinois at Urbana-Champaign, Dec. 2004.

[Kya05] P. Kyasanur and N.Vaidya, "Routing and Interface Assignment in Multi-channel

multi-interface wireless networks," *Proc. of the Wireless Communications and Networking Conference*, 2005, pp. 2051-2056.

[Law00] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis, 3rd Edition*, McGraw-Hill, 2000.

[Lee05] U. Lee, S. Midkiff, and J. Park, "A Proactive Routing Protocol for Multi-Channel Wireless Ad-hoc Networks (DSDV-MC)," *Proc. of the International Conference on Information Technology: Coding and Computing*, Apr. 2005, pp. 710-715.

[Lee06] U. Lee, and S.F. Midkiff, "OLSR-MC: A proactive routing protocol for multi-channel wireless ad-hoc networks", *Proc. of the IEEE Wireless Communications and Networking Conference*, Apr. 2006, pp.331-336.

[MAN99] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," *IETF RFC 2501*, Jan. 1999.

[Mah06] R. Maheshwari, H. Gupta and S. Das, "Multichannel MAC Protocols for Wireless Networks," *Proc. of the 3rd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON),* 2006, pp. 393-401.

[Mar04] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross-Layering in Mobile Ad-Hoc Network Design," *Computer Magazine*, 37(2), Feb. 2004, pp. 48–51.

[McG06] M. McGarry, M. Reisslein and V. Syrotiuk, "Access Control in Heterogeneous Multichannel Wireless Networks," *Proc. of the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks*, 2006.

[Moc81] P. Mockapetris, "Transmission Control Protocol," *IETF RFC 793*, Sep. 1981.

[NS2] The CMU Monarch Project's Wireless and Mobility Extensions to ns, Carnegie Mellon University, August, 1999, http://www.monarch.cs.cmu.edu, Last accessed in Mar. 2007.

[NSDoc] The ns Manual, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, http://www.isi.edu/nsnam/ns/doc/index.html, Last accessed in May, 2007.

[Nir06] Niranjan, S. Pandey, and A. Ganz, "Design and Evaluation of Multichannel Multirate Wireless Networks," *Mobile Networks and Applications*, 11(5), Oct. 2006, pp. 518-524.

[OLSR] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *IETF RFC 3626*, Oct. 2003.

[OOLSR] oolsr-0.99.15, INRIA OLSR implementation, http://hipercom.inria.fr/OOLSR, Last accessed in Mar. 2007.

[Pad05] J. Padhye, S. Agarwal, and V. N. Padmanabhan, "Estimation of Link Interference in Static Multi-hop Wireless Networks," *Proc. of the Internet Measurement Conference*, Oct. 2005, pp. 305-310.

[Qu06] Y. Qu, C.-H. Lung, and A. Srinivasan, "Multi-channel OLSR with Dedicated Control Interface", *Proc. of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS),* Aug. 2006, pp. 155-162.

[So04a] J. So and N. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver," *Proc. of the ACM Intl. Symp. On Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004, pp. 222-233.

[So04b] J. So and N. Vaidya, *A Routing Protocol for Utilizing Multiple Channels in Multi-Hop Wireless Networks with a Single Transceiver*, Tech. Rep., Dept. of Computer Science, University of Illinois at Urbana-Champaign, Oct. 2004.

[Tsc05] C. Tschudin, P, Gunningberg, H.Lundgren, and E. Nordström, "Lessons from Experimental MANET Research," *Elsevier Ad Hoc Networks Journal*, 3(2), Mar. 2005, pp. 221-233.

[Wu00] S.L Wu, C.Y Lin Y.C Tseng, and J.P Sheu, "A New Multi-channel MAC Protocol with On-demand Channel Assignment for Multi-hop Mobile Ad Hoc Networks," *Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, Dec. 2000, pp.232-237.

[Xu02] K. Xu, M. Gerla, and S. Bae, "How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks," *Proc. of the Global Telecommunications Conference (GLOBECOM)*, Nov. 2002, pp. 72- 76.

Appendix A Multi-channel Implementation in ns-2

Simulation in this research is carried out with extended Network Simulator ns-2.28 [NS2]. The original ns-2 wireless node has only one interface on a common channel. Figure A-1 depicts a wireless node structure in ns-2. The ns-2 node interface is composed of the LL layer, the ARP module, the interface queue (IFq), MAC layer, and the physical layer (network interface NetIF and channel). To extend the mobile node with multiple interfaces, we need to replicate the whole ns-2 node interface from LL to channel.

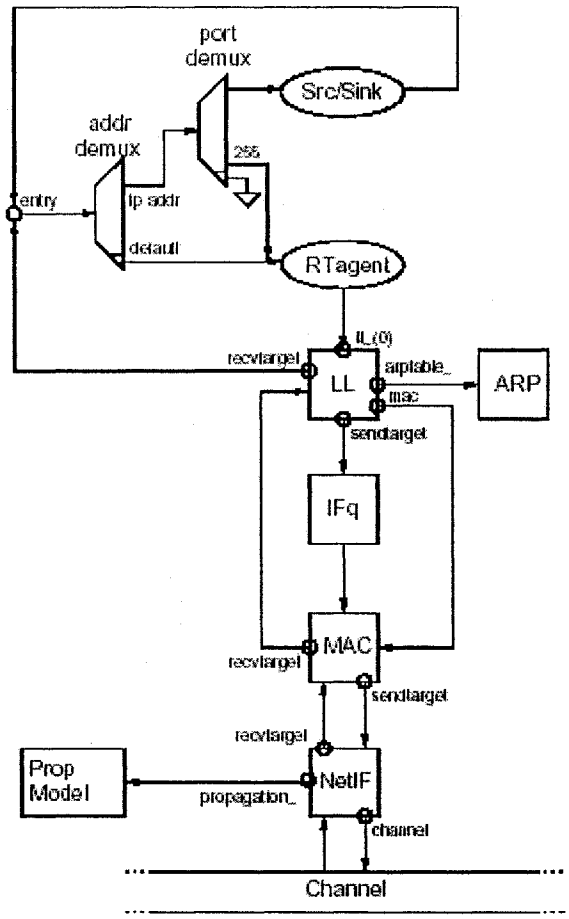


Figure A-1. Schematic of a Mobile Node in [NS2]

An ns-2 wireless node interface is a single channel interface. To realize the multi-channel data transmitting (Tx) and receiving (Rx) interface, an intuitive method is to multiply the connection between channel instances and the node NetIF (Figure A-2). RAgent finds the next hop node address for the outgoing packet; determines which channel is to be used and then puts the channel information in the packet header. For transmitting packets, NetIF checks the channel information in the outgoing packet header, and then forwards it to the corresponding "down target" channel. This packet processing is fast in C++ space. Each channel instance has a list of linked NetIFs listening on this channel. A packet transmitted on a channel will be received by all NetIFs listening on this channel and then continually processed by MAC. The linkage from channel to NetIf is statically configured with TCL script. It is not easy changed by multi-channel protocol during the simulation running. In addition to channel multiplication, if queue-based Tx channel switching is used, IFq and its connection with the LL need also been replicated. This approach of realizing multi-channel interface needs software structural change in ns-2.

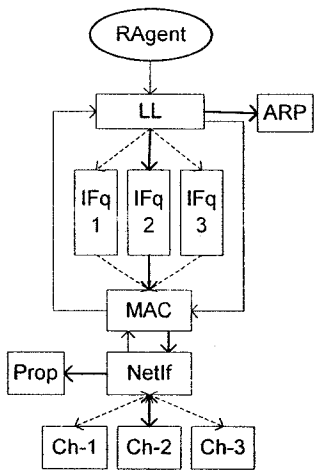


Figure A-2. Multi-channel Interface in ns-2 Mobile Node

Another simpler implementation method does not require modification on the ns-

2 interface structure. The whole ns-2 interface, including all layers from LL to channel, is used to represent one logical channel of a real node multi-channel interface. As Figure A-3 shows, eleven ns-2 interfaces represent the eleven logical data channels (ch0~ch10) which are used by Tx and Rx data interfaces. The third node interface, control interface on the common control channel, is represented by one more ns-2 interface. RAgent connects to the LL components of all ns-2 interfaces. RAgent forwards routing or multi-channel control packets to the control interface (LL_0) and data packets to one of the ns-2 interfaces representing the data Tx channel.

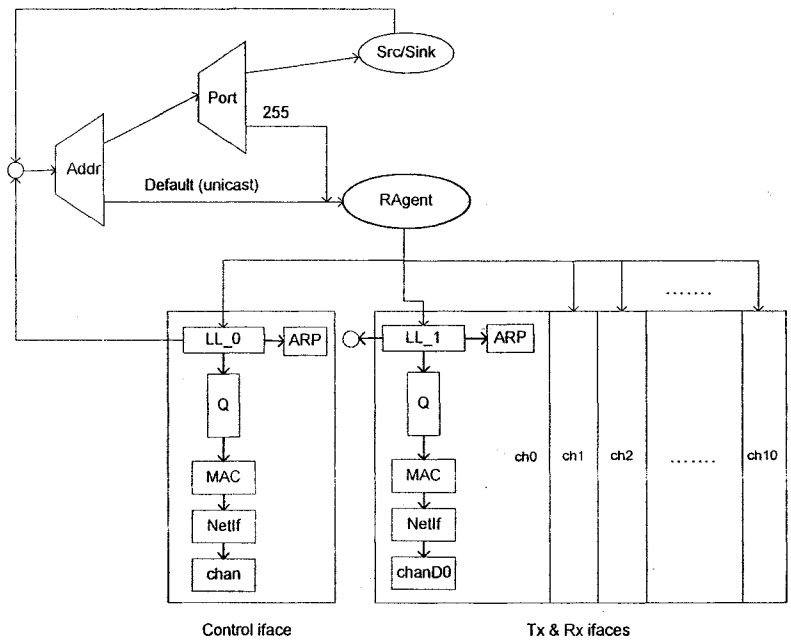


Figure A-3. Structure of Multi-channel Multi-interface Mobile Node in ns-2

A node has data Tx and Rx multi-channel interfaces which only one channel is active on each interface at a time. We do not use one set of ns-2 interfaces (from ch0 to ch10) for each data interface. Usually Tx and Rx interfaces work on different channels, therefore only two of eleven ns-2 interfaces (channels) should be active at a time. In case Tx and Rx node interfaces use the same channel, then only one ns-2 interface is active.

This is a correct representation of the scenario in real life, because interference prevents two node interfaces working simultaneously on the same channel.

The Tx data interface switches to different next hop node channels. Only one logical channel can be activated at a time. It is controlled by a queue-based channel switching mechanism (Section 4.7). If outgoing packets are sent down from the routing agent to an inactive channel, the packets will be held in the IFq until this channel is activated.

For packet receiving, packets on all channels can be received by one of the ns-2 data interfaces. Packet received on ns-2 interface other than the assigned node channel will be dropped at MAC layer because of the incorrect destination MAC address. Therefore only one logical channel is actually active for receiving at a time.

Appendix B TCL Scripts

This sections list three important TCL script examples used in simulation.

- Ns2.tcl

This is ns-2 configuration script. Multi-channel multi-interface ns-2 node is constructed. Simulation control variants will be transfer into this script as arguments, such as the number of channels (*numofchannels*), connection pattern (*cp*) and network topology (*sc*) etc.

- Runsim.tcl

  This is the simulation execution control file. It set the simulation scenarios and call the ns2.tcl to automatically run a set of simulations, analyze simulation trace and generate report.

- Trace analysis script and an example of analysis report are appended at the end

B-1 Ns-2.tcl

```
# ======================================================================
# Default Script Options
# ======================================================================
set opt(mc)       1
set opt(cp)    ""   ;# connect pattern "cbr-rate-pktsize-flows"
set opt(sc)          ""          ;# scen (topology and movement)

set opt(ifqlen)    30
set opt(switchdelay)          200       ;# 200 us
set opt(numofchannels)    5          ;# Change NumofChan with opt

# ======================================================================

proc usage {} {
  global argv0

  puts "\nusage: $argv0 -mc 1|0 -cp -sc \[-ifqlen \] \[-switchdelay \] \n"
  puts "Qmaxtime is only indicator here, need recompile after change\n"
```

```
}

proc getopt {argc argv} {
        global opt

        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue

                set name [string range $arg 1 end]
                set opt($name) [lindex $argv [expr $i+1]]
        }
}

#-----------------------------------------------------------------------
# Initialization Script Options
#-----------------------------------------------------------------------

getopt $argc $argv

if { $argc <= 1 } {
   usage
   exit
}

set TraceName "../trace/NCA_$opt(numofchannels)chans_$opt(cp)_$opt(sc)"

# (possibly) Remove old trace
exec sh -c "rm -f $TraceName"

# remove old tmp folder
exec sh -c "rm -rf tmp && mkdir tmp"

# Default node configuration
set nodeConfig "no-log 1; log-none ; log-route 1"

# Load the OOLSR as plugin
load-plugin /home/quyi/ns-allinone-2.28-oolsr-0.99.15/ns-2.28/smolsr/oolsr-plugin

# ======================================================================
# Define options
# ======================================================================

set val(cp)   "../ConnectionPattern/$opt(cp)"
set val(sc)   "../scen_out_ss/$opt(sc)"

set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(nn)    36
set val(rp) PLUGINPROTOCOL
set val(x) 800
set val(y) 800
set val(stop)   41.0         ;# simulation time

Mac/802_11 set dataRate_ 2Mb        ;# 802.11 dataRate

# ======================================================================
```

```
# Main Program
# ======================================================================

set ns_         [new Simulator]

ns-random 0

set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
set god_ [create-god $val(nn)]

$ns_ use-newtrace
set tracefd     [open $TraceName w]
$ns_ trace-all $tracefd


#
# define how node should be created
#

#Creating Channel

set chan_ [new $val(chan)]
set chan_0_ [new $val(chan)]
set chan_1_ [new $val(chan)]
set chan_2_ [new $val(chan)]
set chan_3_ [new $val(chan)]
set chan_4_ [new $val(chan)]
set chan_5_ [new $val(chan)]
set chan_6_ [new $val(chan)]
set chan_7_ [new $val(chan)]
set chan_8_ [new $val(chan)]
set chan_9_ [new $val(chan)]
set chan_10_ [new $val(chan)]

#global node setting

$ns_ node-config -adhocRouting $val(rp) \
   -llType $val(ll) \
   -macType $val(mac) \
   -ifqType $val(ifq) \
   -ifqLen $opt(ifqlen) \
   -antType $val(ant) \
   -propType $val(prop) \
   -phyType $val(netif) \
   -channel $chan_ \
   -channelD0 $chan_0_ \
   -channelD1 $chan_1_ \
   -channelD2 $chan_2_ \
   -channelD3 $chan_3_ \
   -channelD4 $chan_4_ \
   -channelD5 $chan_5_ \
   -channelD6 $chan_6_ \
   -channelD7 $chan_7_ \
   -channelD8 $chan_8_ \
   -channelD9 $chan_9_ \
   -channelD10 $chan_10_ \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace OFF \
   -macTrace OFF \
   -movementTrace OFF
```

```
puts "node-config set"

#
#  Create the specified number of nodes [$val(nn)]
#

for {set i 0} {$i < $val(nn) } {incr i} {
   set node_($i) [$ns_ node]
   [$node_($i) set ragent_] multichan $opt(mc)
   [$node_($i) set ragent_] numofchannels $opt(numofchannels)      ;# new in NCA
   [$node_($i) set ragent_] switchdelay $opt(switchdelay)

   $node_($i) random-motion 0

   [$node_($i) set ragent_] set-config \
     "$nodeConfig ; log-file-name tmp/oolsr-node-$i.log"

}


#
# Define traffic model
#
puts "Loading connection pattern...$opt(cp)"
source $val(cp)

#
# Define node movement model
#
 puts "Loading scenario file...$opt(sc)"
 source $val(sc)

# Define node initial position in nam
# i don't use nam
 for {set i 0} {$i < $val(nn)} {incr i} {
   # 20 defines the node size in nam, must adjust it according to your scenario
   # The function must be called after mobility model is defined
    $ns_ initial_node_pos $node_($i) 20
 }


#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at $val(stop).0 "$node_($i) reset";
}

$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"

puts "Starting Simulation..."
for {set j 1.0} {$j < $val(stop)} {set j [expr $j * 2.3 ]} {
          $ns_ at $j "puts t=$j"
   }

$ns_ run
```

B-2 Runsim.tcl

Here is an example of the control file to run NCA with variant number of flows and channels.

```
#!/bin/csh
#
unset noclobber
#
set logfile = 'NCA2-chanflow.log'
set NSpath = '/home/quyi/ns-allinone-2.28-oolsr-0.99.15'
set NCApath = '/home/quyi/NCA-ns-allinone-2.28-oolsr-0.99.15'
set RCApath = '/home/quyi/RCA-ns-allinone-2.28-oolsr-0.99.15'
#
echo ========================================================== >$logfile
echo === NCA with varying flows and channels on 10 scens ===>>$logfile
echo ========================================================== >>$logfile
echo >>$logfile

mv $NCApath $NSpath

foreach Chans (3 5 7 9 11)
echo +++++++  START Total Number of Channels = $Chans  +++++  >>$logfile
echo >>$logfile
  foreach Flows (2 3 4 5 6)
  echo ===== Start Total $Flows Flows  =====  >>$logfile
        foreach sc ('s0' 's1' 's2' 's3' 's4' 's5' 's6' 's7' 's8' 's9')
        echo --- NCA $Chans Channels/ $Flows Flows/ $sc >>$logfile
        $NSpath/ns-2.28/ns nca-2M.tcl -cp $Flows'flows' -sc $sc -numofchannels $Chans
        awk -f ../trace/traceanalyse.awk ../trace/'NCA_'$Chans'chans_'$Flows'flows_'$sc >>$logfile
        echo >>$logfile
        end
  echo ===== End Total $Flows Flows  =====  >>$logfile
  echo >>$logfile
  end
echo ++++++  END Total Number of Channels = $Chans  +++++++  >>$logfile
echo >>$logfile
end

mv $NSpath $NCApath
echo end
```

B-3 Trace Analysis Script

```
# To execute
# $awk -f template.awk out.tr > resultfile
#
BEGIN {alldrops=0; NRTEdrops=0; IFQdrops=0; ARPDrops=0;ENDDrops=0; priQDrops=0;
        sends=0; recvs=0; highest_packet_id=500000;
        CBR_size=0; sum=0;traffic_end=0; RETDrops=0;
        pluginCOL=0; arpCOL=0; allCOL=0; cbrCOL=0; Loop = 0 ;#
    } #Init
#
# Event type
#  s send
#  r receive
#  d drop
#  f forward
```

```
{  action = $1;  time = $3;  #event type
#
# Next hop info
   Hs_nodeId = $5;                  # -Hs: id for this node
   Hd_nextHopid = $7;               # -Hd: id for next hop towards the destination.
#
# Node property
# -Nw: reason for the event. The different reasons for dropping a packet are given below:
# "END" DROP_END_OF_SIMULATION
# "COL" DROP_MAC_COLLISION
# "DUP" DROP_MAC_DUPLICATE
# "ERR" DROP_MAC_PACKET_ERROR
# "RET" DROP_MAC_RETRY_COUNT_EXCEEDED
# "STA" DROP_MAC_INVALID_STATE
# "BSY" DROP_MAC_BUSY
# "NRTE" DROP_RTR_NO_ROUTE i.e no route is available.
# "LOOP" DROP_RTR_ROUTE_LOOP i.e there is a routing loop
# "TTL" DROP_RTR_TTL i.e TTL has reached zero.
# "TOUT" DROP_RTR_QTIMEOUT i.e packet has expired.
# "CBK" DROP_RTR_ARP_CALLBACK
# "IFQ" DROP_IFQ_QFULL i.e no buffer space in IFQ.
# "ARP" DROP_IFQ_ARP_FULL i.e dropped by ARP
   Ni_nodeId = $9;                  # -Ni: node id
   Nx_nodeX = $11;                  # -Nx: node[] 0,,3   0...4s x-coordinate
   Ny_nodeY = $13;                  # -Ny: node[] 0,,3   0...4s y-coordinate
   Nz_nodeZ = $15;                  # -Nz: node[] 0,,3   0...4s z-coordinate
   Ne_nodeEnergy = $17;      # -Ne: node energy level
   Nl_trace = $19;          # -Nl: trace level, such as AGT, RTR, MAC
   Nw_dropReason = $21;  # -Nw: reason for the event.
#
# Packet info at MAC level
   Ma_duration = $23;               # -Ma: duration
   Md_dstEthAdd = $25;    # -Md: dst[] 0,,3   0...4s ethernet address
   Ms_srcEthAdd = $27;      # -Ms: src[] 0,,3   0...4s ethernet address
   Mt_ethType = $29;        # -Mt: ethernet type
#
# Packet information at IP level T
   Is_srcAddPrt = $31;              # -Is: source address.source port number
   typeARP = $31;                   # for ARP pkt, this position is -P arp
   Id_dstAddprt = $33;      # -Id: dest address.dest port number
   It_pktTypt = $35;        # -It: packet type
   Il_pktSize = $37;        # -Il: packet size
   If_flowId = $39;         # -If: flow id
   Ii_PktId = $41;          # -Ii: unique id
   Iv_ttl = $43;           # -Iv: ttl value
#
# -P cbr Constant bit rate. Information about the CBR application is represented by the following tags:
   Pn_cbr = $45;            # -Pn
   Pi_seq = $47;            # -Pi: sequence number
   Pf_forTimes = $49;       # -Pf: how many times this pkt was forwarded
   Po_optTimes = $51;           # -Po: optimal number of forwards
#
#============= CALCULATE PDR  ========================================
#
# so only checked recv olsr packets, because some may not recv because of conflication
  if (action == "s" && It_pktTypt=="cbr" && Nl_trace == "AGT")  sends++;
  if (action == "r" && It_pktTypt=="cbr" && Nl_trace == "AGT")  recvs++;
  # all drops, when turn OFF MAC trace, pkt drop at MAC can not be counted, but it does not affect PDR and
throughput caculate
  if (action == "d" && It_pktTypt=="cbr" )      alldrops++          # all traced drops
  if (action == "d" && It_pktTypt=="cbr" && Nw_dropReason == "NRTE") NRTEdrops++;
  # if (Hs_nodeId == Hd_nextHopid  && Nw_dropReason == "NRTE") {NRTEdrops--; Loop++;}
```

```
  if (action == "d" && It_pktTypt=="cbr" && Nw_dropReason == "IFQ") IFQdrops++;
  if (action == "d" && It_pktTypt=="cbr" && Nw_dropReason == "ARP") ARPDrops++;
  if (action == "d" && It_pktTypt=="cbr" && Nw_dropReason == "---") priQDrops++;
  if (action == "d" && It_pktTypt=="cbr" && Nw_dropReason == "END") ENDDrops++; # it is avoid by stop
traffic before End simulation
  if (action == "d" && It_pktTypt=="cbr" && Nw_dropReason == "RET") RETDrops++;
#
#============= CALCULATE THROUGHPUT ========================================
#
  if (CBR_size == 0)  {                                    # just read the size one time
        if (It_pktTypt=="cbr")        {                    #
          CBR_size = Il_pktSize;                          #
        }                                                  #
  }                                                        #
#
#============= CALCULATE DELAY ============================================
  # getting start time is not a problem, provided you're not starting
  # traffic at 0.0.
  # could test for sending node_1_address or flow_id here.
#
  if ( Ii_PktId > highest_packet_id ) highest_packet_id = Ii_PktId;
#
  if ( start_time[Ii_PktId] == 0 ) start_time[Ii_PktId] = time;
#
  # only useful for small unicast where packet_id doesn't wrap.
  # checking receive means avoiding recording drops
  if ( action != "d" ) {
    if ( action == "r" ) {
      end_time[Ii_PktId] = time;
      traffic_end = time;
        }
  } else {
    end_time[Ii_PktId] = -1;
   }

}
END {
   for ( Ii_PktId = 0; Ii_PktId <= highest_packet_id; Ii_PktId++ ) {
     start = start_time[Ii_PktId];
     end = end_time[Ii_PktId];
     packet_duration = end - start;
     if ( start < end ) sum= packet_duration+sum;
  }
 delay=sum/recvs;            #
 throughput = recvs*CBR_size*8/1024/(traffic_end-10);  # traffic start at 10s
 PDR = 100*recvs/(sends-ENDDrops);
 MACDrops = sends-recvs-alldrops;              # not traced MAC, but caculate,  total drops (sends-recvs) - traced
drops = no traced MAC drops
#
printf(" e-e delay =          \t %f \n", delay);
printf(" PDR (100%) =  \t %f (received = %d, sent = %d)\n", PDR, recvs, sends-ENDDrops);
printf(" Throughput =  \t %f kbps (pktsize=%d, duration=%f)\n",throughput,CBR_size,traffic_end-10);

#printf(" Total drop =  \t %d \n", sends-recvs-ENDDrops);          # include all traced drop (except END) and non-
traced (MAC) drop
#printf(" NTRE drop =    \t %d \n", NRTEdrops);
#printf(" IFQ drop =     \t %d \n", IFQdrops);
#printf(" ARP drop =     \t %d \n", ARPDrops);
#printf(" priQ drop =        \t %d \n", priQDrops);
# all non-traced drop are counted as MAC, it is little inaccurate if there are few other (eg.LOOP) drop at RTR or
AGT.
# if I turn MAC trace ON, i can statistic specificly, but it slow down simulation too much
```

```
#printf(" MAC drops =    \t %d  \n", MACDrops);
}
```

B-4 Example of Analyzing Report

```
=== NCA flowrate 100 ===
- s0 -
 e-e delay =              0.141782
 PDR (100%) =    96.625561 (received = 8619, sent = 8920)
 Throughput =    1131.380645 kbps (pktsize=512, duration=30.472503)
- s1 -
 e-e delay =              0.039700
 PDR (100%) =    99.720201 (received = 8910, sent = 8935)
 Throughput =    1186.681581 kbps (pktsize=512, duration=30.033330)
=== SC flowrate 100 ===
- s0 -
 e-e delay =              0.756811
 PDR (100%) =    31.670209 (received = 2831, sent = 8939)
 Throughput =    368.228329 kbps (pktsize=512, duration=30.752658)
- s1 -
 e-e delay =              0.093616
 PDR (100%) =    70.826345 (received = 6334, sent = 8943)
 Throughput =    830.019767 kbps (pktsize=512, duration=30.524574)
```