Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 52 (2008) 831–849
Computer Networks
www.elsevier.com/locate/comnet

Time-efficient distributed layer-2 auto-configuration for cognitive radio networks ☆

Srinivasan Krishnamurthy, Mansi Thoppian, Srikant Kuppa, R. Chandrasekaran, Neeraj Mittal *, S. Venkatesan, Ravi Prakash

Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, Richardson, TX 75083, USA

Available online 23 November 2007

Abstract

Cognitive radios (CR) have the ability to dynamically adapt to local spectrum availability. In a network comprised of CR-enabled devices, layer-2 auto-configuration involves determining a common set of channels to facilitate communication among participating nodes. This is a unique challenge as nodes in the CR network may be unaware of (a) their neighbors and (b) the channels on which they can communicate with a neighbor. In this paper, we propose a time-efficient distributed algorithm for layer-2 auto-configuration for a CR network. Our algorithm finds the globally common channel set in $2MN + O(DN)$ timeslots, where each node is assigned a unique identifier from the range $[1, \ldots, N]$, M is the maximum number of channels available for communication, and D is the diameter of the network. All nodes know M and N. We present both diameter-aware and diameter-unaware versions of the algorithm. We then show that the proposed algorithms are efficient by proving a matching lower bound. Finally, we investigate a special case when nodes have more knowledge available at their disposal and discuss how the time-complexity of our algorithm can be improved under this case.

Keywords: Cognitive radio; Multiple channels; Neighbor discovery; Distributed algorithm

1. Introduction

With the unprecedented proliferation of wireless communication devices in recent times, it is being claimed that there will be an acute shortage of bandwidth in the near future. This claim is seemingly supported by the observation that most of the available frequency bands have already been allocated to various applications (*e.g.*, television transmission, microwave communication, cellular communication, etc.) [5]. However, it has been noted that a significant portion of the allocated spectrum is underutilized [6,3]. For example, in several urban areas many of the television channels in the VHF and UHF bands are unassigned.

Cognitive radio (CR) technology [19] allows wireless devices to dynamically adapt to spectrum

☆ A preliminary version of this paper appeared in IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2005) [13].

* Corresponding author.
E-mail addresses: ksrini@student.utdallas.edu (S. Krishnamurthy), mansi@utdallas.edu (M. Thoppian), ksrikant@utdallas.edu (S. Kuppa), chandra@utdallas.edu (R. Chandrasekaran), neerajm@utdallas.edu (N. Mittal), venky@utdallas.edu (S. Venkatesan), ravip@utdallas.edu (R. Prakash).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.11.013

availability in their geographical region. The owner of a (licensed) channel is referred to as *primary user* and all other users of the channel as *secondary users* [3]. CR technology enables secondary users to scan and identify unused channels in a frequency spectrum. A channel is said to be *available* if the secondary user can send and receive messages on the channel without interfering with the primary user(s). Such a communication infrastructure based on CR technology has applications in defense and relief and rescue operations. Since the usage of spectrum varies widely from one region to another [6], communication among users (soldiers in a platoon or relief personnel in a disaster-prone area) must rely on a dynamic channel assignment scheme. Also, in military applications and disaster recovery scenarios such as the shuttle recovery effort in east Texas or hurricane affected areas, significant parts of the spectrum are likely to be available for use by secondary users. When a secondary user (hereafter, referred to as a node in the CR network) independently scans the spectrum usage and maintains the set of (locally) available channels, the following layer-2 auto-configuration issues arise:

1. How do nodes detect their neighbors and collectively form a communication infrastructure in the absence of a central authority?
2. How do nodes decide on the set of channel(s) that can be used for communication?

Note that the CR nodes may be turned on at different times. Wireless communication among neighbors is possible if the source and destination nodes tune to a common channel at the same time. In the layer-2 auto-configuration problem in a CR network, a *common set of channels* (referred to as the *globally common channel set*, $\mathscr{G}_0$, in this paper) needs to be determined. The motivations behind finding $\mathscr{G}_0$ are:

a. There may be multiple groups of nodes deployed in a geographical area, say in a military operation (there may be many platoons, with each platoon being a group) or at the site of a natural disaster (firemen, paramedics, police being three groups). It is important that each group chooses a unique channel for communication among the group members with few nodes acting as gateways between groups.
b. Node mobility leads to frequent changes in network topology. For such systems, communication over a *globally common* channel provides a simple and effective solution ([7,17]).
c. Since a globally common channel is available at all the nodes, and the nodes may be distributed over a wide geographical region, using such a channel leads to a fairly stable communication infrastructure.

Finding a common set of available channels (that is, globally common channel set) for communication is non-trivial because of the divergence in the sets of available channels at individual nodes and the absence of a central authority. This is because communication infrastructures in military and relief operations are usually ad hoc in nature. The complexity of the problem is further increased due to the following reasons:

i. Nodes do not have prior knowledge about the number and identities of other nodes in their neighborhood.
ii. Although two nodes may be physically close, their channel availability sets may be different. Thus, nodes are unaware of the existence of a common channel in their one-hop neighborhood.
iii. Changes in neighborhood due to node mobility can play a significant role in computing $\mathscr{G}_0$. So, it is very important that the distributed computation terminates quickly.

1.1. Our Contributions

Let each node be assigned a unique identifier from the range $[1, \ldots, N]$, and $\mathscr{A}_{\text{univ}}$ be the set of M possible channels all the nodes are capable of operating on. In this paper, we propose a 2-phase auto-configuration algorithm that enables the nodes to dynamically compute the globally common channel set, $\mathscr{G}_0$ in a distributed manner. All nodes know $\mathscr{A}_{\text{univ}}$[1] and the value of N. We present a fully distributed algorithm to determine $\mathscr{G}_0$ in $2MN + O(DN)$ timeslots, where D is the diameter of the network. The algorithm consists of two phases. Phase 1 (*neighbor discovery and configuration*) of the algorithm consists of $2MN$ timeslots split equally between *neighbor discovery* and *conveying locally common broadcast channel*. Phase 2 consists of $O(DN)$ timeslots and is used for the computation of $\mathscr{G}_0$. If D is known, then the

[1] When the radios are built, they have a range of frequencies they can operate on.

number of bits exchanged per message is $O(M)$. Otherwise, the number of bits per message is $O(M + \log N)$. The running time of the algorithms is independent of $|\mathscr{G}_0|$. We present a *matching* lower bound for neighbor discovery and show that our algorithm is optimal. Finally, we investigate a special case when channel availability sets of neighboring nodes do not differ drastically. We show that, with this additional assumption, the time-complexity for neighbor discovery can be improved significantly.

1.2. Related work

Neighbor discovery and configuration for multi-hop, *multi-channel* radio networks are problems of interest but most of the current research is on networks with a single broadcast channel [28]. Researchers in the Bluetooth [8] community have studied neighbor discovery in multi-channel systems with frequency-hopping. Symmetric neighbor discovery protocols for Bluetooth that do not depend upon preassigned roles for the nodes have been proposed in [16,27,28,30]. These protocols assume identical frequency distribution at each node.

Alonso et al. [2] have proposed randomized algorithms with expected running times for neighbor discovery that result in nodes agreeing upon a common communication channel in a multi-node, multi-channel environment. Their algorithms are independent of the network configuration. The expected running time is computed based on a probabilistic model that assigns probabilities for a node randomly choosing a channel i and then being engaged in transmission or reception on that channel. They assume that the nodes have identical probability distributions of frequency allocation. The implicit assumption made in the paper is that all frequencies are available at each node.

Many MAC-layer solutions have been proposed for multi-channel networks [10,11,21,22,25,26,31,32,35]. These solutions usually assume either the existence of a common control channel [31,32,35] and/or that every node is equipped with a separate radio interface for each channel [10,21,22]. Solutions for multi-channel wireless mesh networks [11,25,26] assume that all channels are available throughout the network. Krishnamurthy et al. [14] presented a solution to the problem of neighbor discovery and configuration in CR networks under the assumption that a common control channel exists. Thoppian et al. [34] have recently proposed a distributed heuristic for MAC-layer scheduling in CR networks, which again assumes the existence of a common control channel.

2. System model

Throughout this paper, we assume that the multi-hop, multi-channel wireless network formed by a group of CR-enabled nodes remains static for the duration of auto-configuration. The assumption remains valid as long as the nodes in the network do not move rapidly and the auto-configuration algorithm takes a small amount of time. It is expected that if nodes are mobile, the auto-configuration algorithm will be run periodically. Nodes have access to synchronous clocks (such as GPS) and time is slotted. Thus, every node knows when a timeslot starts and when a timeslot ends.

2.1. Medium characteristics

Let $\mathscr{A}_{\text{univ}} = \{c_1, c_2, \ldots, c_M\}$ represent the universal set of M channels that the CR can operate on. Each node is aware of $\mathscr{A}_{\text{univ}}$. We assume that every channel has a unique identity and the mapping between the channel identifier and its frequency is known to all nodes. The communication medium is assumed to be loss-free.

2.2. Node characteristics

Every node, say i, is assigned a unique identifier, say $\mathscr{UID}_i$ from the range $[1, \ldots, N]$. Observe that N denotes the maximum number of nodes a network can contain. In military and relief operations, the maximum number of soldiers in a platoon or firemen assigned for relief efforts is known a priori. Hence, we assume that the maximum number of nodes (given by N) is known to every node. For ease of exposition, $\mathscr{UID}_i$ is represented by i throughout this paper. Every node has a single wireless transceiver and the transceiver is built so that it can transmit or receive in any of the M channels of $\mathscr{A}_{\text{univ}}$. In addition, each node i knows $\mathscr{A}_i$, the set of channels (locally) available to itself. Nodes i and j are said to be (one-hop) *neighbors* of each other if they are within communication range of each other and there is at least one channel common to $\mathscr{A}_i$ and $\mathscr{A}_j$, that is, $\mathscr{A}_i \cap \mathscr{A}_j \neq \emptyset$.

2.3. Network operation

Nodes in the CR network perform one of the following two operations at any time: (i) layer-2 auto-configuration, or (ii) normal operation as shown in Fig. 1. These two operations repeat periodically every T time units. The times at which layer-2 auto-configuration operation start is known *a priori* to all the nodes by letting every node know the value of T. During layer-2 auto-configuration, nodes learn about other participating nodes in the network and also determine the globally common channel set, $\mathscr{G}_0$. In the normal mode of operation, the nodes may behave similarly to the nodes in any other multi-hop wireless network, such as a MANET [4] or a mesh network [1]. (Note that synchronized clocks are needed only for layer-2 auto-configuration; during normal operation, channel access methods without using synchronous clocks, such as contention based approaches, can be used.) It is necessary to periodically invoke the layer-2 auto-configuration algorithm to account for a varying globally common channel set, $\mathscr{G}_0$, due to the changes in network topology and/or channel availability sets of individual nodes. To summarize, each node i has the following parameters configured at boot time: N, M, $\mathscr{A}_{\text{univ}}$, and T. A node i determines its availability set $\mathscr{A}_i$ by sensing the local spectrum.

3. Layer-2 auto-configuration algorithm

During the layer-2 auto-configuration process, the time division multiple access (TDMA) scheme is used for communication among nodes. Time is divided into $O(D)$ rounds. A round is defined as the time taken for every node to communicate with each one of its *neighbors* using a (local) broadcast mechanism. Each round consists of equal-sized intervals referred to as *frames*. The number of frames in a round may vary as shown in Fig. 1. A round in phase 1 consists of M frames while a round in phase 2 consists of only one frame. Each frame is further divided into N *timeslots*, each of equal length. Node i transmits during the ith timeslot in each frame (see Fig. 1) and all other nodes are in receive mode during the ith timeslot. We next present more details of the algorithm.

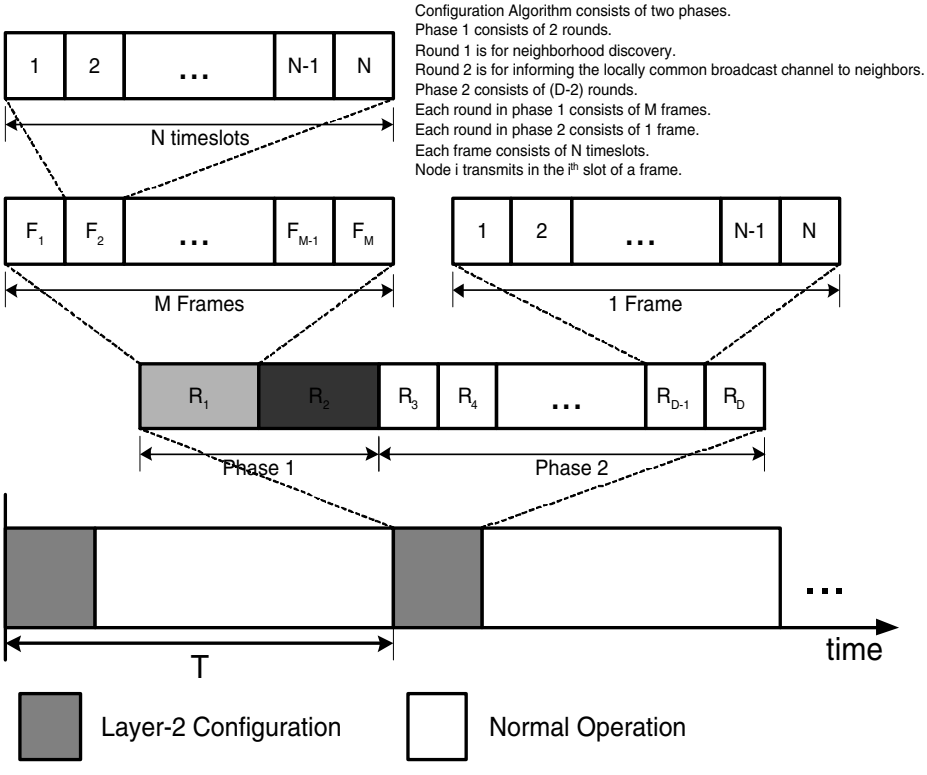


Fig. 1. Operation cycle of a diameter-aware node.

3.1. Data structures

The following data structures are maintained at every node i:

$\mathcal{UID}_i$	identity of the node i
$\mathcal{A}_i$	set of channels available at node i ($\mathcal{A}_i \subseteq \mathcal{A}_{\text{univ}}$)
$\mathcal{NBR}_i$	set of one-hop neighbors of node i
PC_i	preferred channel for node i
$\mathcal{G}_i$	set of channels common among all nodes within r hops of node i
r	current round number

The availability set, $\mathcal{A}_i$ is independently obtained by node i by scanning the medium and estimating the available channels. This step is external to our algorithm. The channels in $\mathcal{A}_i$ are sorted by their frequencies. A *preferred channel* for node i is a channel in which transmissions by i can be received by all of its neighbors. Thus, PC_i has to be a channel that is available at i and all the neighbors of i, that is, $PC_i \in \mathcal{A}_i$ and $PC_i \in \mathcal{A}_j$ for each $j \in \mathcal{NBR}_i$. If there are multiple such common channels, any one of them can be selected as a preferred channel. The concept of a *round* (denoted by r) is defined to monitor the progress of the configuration algorithm, as in other synchronous distributed algorithm [23]. Initially, $\mathcal{UID}_i = i$, $\mathcal{NBR}_i = \emptyset$, $PC_i = \perp$, $\mathcal{G}_i = \mathcal{A}_i$, and $r = 0$.

3.2. Diameter-aware auto-configuration

Assume that all the nodes are aware of the diameter of the network, D. The following algorithm determines the globally common channel set. The algorithm consists of two phases (see Fig. 1).

3.2.1. First phase or neighbor discovery and configuration

The first phase consists of two rounds. The first round is for *neighbor discovery*, while the second round is used by a node i for conveying the common channels in its neighborhood to each of its neighbors. The two rounds differ only in the information that is transmitted. Let $\mathcal{L}_i^x$ denote the set of channels that are common among all nodes in the x-hop neighborhood of node i. Formally,

$$\mathcal{L}_i^x \triangleq \bigcap_{\text{distance between nodes } i \text{ and } j \text{ is at most } x} \mathcal{A}_j$$

Note that, when $x \geqslant D$, $\mathcal{L}_i^x = \mathcal{G}_0$ for each node i in the network. During round 1, each node i communicates its channel availability set $\mathcal{A}_i$ and receives $\mathcal{A}_j$ from each neighbor j. Thus, each node i becomes aware of its set of neighbors $\mathcal{NBR}_i$ and their respective channel availability sets by the end of round 1. Node i then computes the set of channels that are common to itself and its neighbors by performing an intersection operation over the channel availability sets of i and all $j \in \mathcal{NBR}_i$. From node i's perspective, the local information available at i at the end of round 1 is sufficient to establish communication with its neighbors. However, its neighbors are not yet aware of $\mathcal{L}_i^1$. In round 2, i communicates $\mathcal{L}_i^1$ to inform the neighbors of the channels i could potentially use for local broadcast. Each of the two rounds consists of M frames $(F_1, F_2, \ldots, F_M)$ and each frame consists of N timeslots as shown in Fig. 1. Thus, phase 1 consists of $2MN$ timeslots. During frame $F_x (1 \leqslant x \leqslant M)$, every node i with $c_x \in \mathcal{A}_i$ tunes its transceiver to channel c_x. If $c_x \notin \mathcal{A}_i$, i turns off its transceiver during frame F_x.

3.2.1.1. First round or neighbor discovery. Consider the xth frame F_x. Each node i transmits the contents of $\mathcal{G}_i$ on channel c_x during the ith timeslot of frame F_x if $c_x \in \mathcal{A}_i$; i remains silent otherwise during the timeslot. All the other nodes are in receiving mode on channel c_x in the ith timeslot of frame F_x (if c_x is in their availability set). This corresponds to timeslot number $((x-1) \times N + i)$ from the beginning of the first round. During the remaining timeslots of F_x, node i is in the receive mode. If node i receives node k's transmission during (the kth timeslot of) frame F_x, node i learns that $k \in \mathcal{NBR}_i$ and $c_x \in \mathcal{A}_k$. In this case, node i updates its data structures. Note that a node i could send a *hello* message in the timeslots corresponding to channels in $\mathcal{G}_i$ to convey the above information. Transmitting the full contents of $\mathcal{G}_i$ adds robustness to the system.

Consider the sample network shown in Fig. 2. Here, node 2 will transmit the set $\mathcal{G}_2 = \mathcal{A}_2 = \{c_1, c_2, c_3, c_5\}$ during the second timeslot of frames F_1, F_2, F_3 and F_5. Overall they correspond to the 2nd, 9th, 16th and 30th timeslots from the beginning of the first round. During timeslot numbers 20 and 32, node 2 remains silent since c_4 and c_6 do not belong to $\mathcal{A}_2$. Fig. 3 shows the transmissions of all five nodes during the first round. Note that provision has been made for transmission by nodes 3 and 7 during each frame even though the nodes

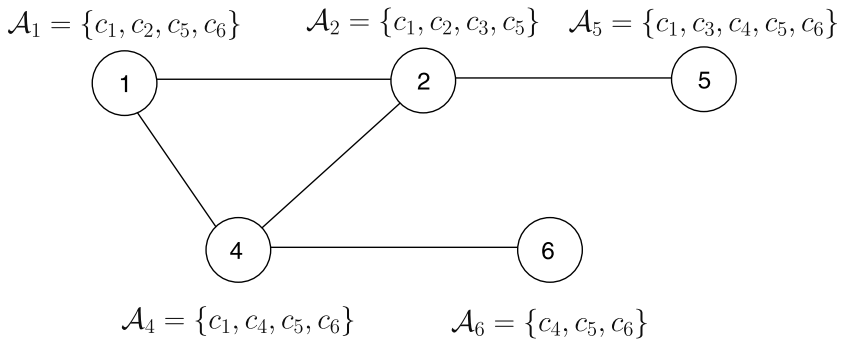


Fig. 2. A sample CR network with $N = 7$ and $M = 6$.

		Timeslots						
		1	2	3	4	5	6	7
Frames	1	$\{c_1, c_2, c_5, c_6\}$	$\{c_1, c_2, c_3, c_5\}$		$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_5, c_6\}$		
	2	$\{c_1, c_2, c_5, c_6\}$	$\{c_1, c_2, c_3, c_5\}$					
	3		$\{c_1, c_2, c_3, c_5\}$			$\{c_1, c_3, c_4, c_5, c_6\}$		
	4				$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_5, c_6\}$	$\{c_4, c_5, c_6\}$	
	5	$\{c_1, c_2, c_5, c_6\}$	$\{c_1, c_2, c_3, c_5\}$		$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_5, c_6\}$	$\{c_4, c_5, c_6\}$	
	6	$\{c_1, c_2, c_5, c_6\}$			$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_5, c_6\}$	$\{c_4, c_5, c_6\}$	

Fig. 3. Transmissions during round 1 of phase 1.

are not present in the network. After the first round, each node i knows the identities of all its one-hop neighbors (maintained locally in set $\mathcal{NBR}_i$) and their respective availability sets. Node i updates the set $\mathcal{G}_i$ as follows: $\mathcal{G}_i = \mathcal{G}_i \cap (\cap_{j \in \mathcal{NBR}_i} \mathcal{G}_{i,j})$, where $\mathcal{G}_{i,j}$ denotes the channel set received by node i from the neighboring node j. It also increments r by one.

Node i can select one of the channels in the updated set $\mathcal{G}_i$ as its *preferred channel* (PC_i) for transmission. When node i transmits on its preferred channel during ith timeslot, its transmission can be received by all of its one-hop neighbors if all nodes $j \in \mathcal{NBR}_i$ tune to PC_i during ith timeslot. For the sample network in Fig. 2, at the end of round 1, $\mathcal{NBR}_1 = \{2, 4\}$ and the updated set $\mathcal{G}_1$ for node 1 is $\{c_1, c_5\}$. So, if node 1 transmits on channel c_1 during the first timeslot of a frame, its transmission can be heard by all of its one-hop neighbors if they tune to c_1. After the first round, only node i knows its preferred channel. Before node i begins to transmit *only* on PC_i, it has to inform its neighbors about its selection. The second round achieves this objective.

3.2.1.2. Second round or locally common channel propagation. Nodes exchange their updated sets in the second round. Every node i transmits $\mathcal{G}_i$ on channel $c_x \in \mathcal{A}_i$ during ith timeslot of frame F_x. Once again, consider the sample network shown in Fig. 2. After the first round, the updated sets are $\mathcal{G}_1 = \{c_1, c_5\}$, $\mathcal{G}_2 = \{c_1, c_5\}$, $\mathcal{G}_4 = \{c_5\}$, $\mathcal{G}_5 = \{c_1, c_3, c_5\}$ and $\mathcal{G}_6 = \{c_4, c_5, c_6\}$. Transmissions by the nodes during the second round are shown in Fig. 4. Nodes use a deterministic mechanism to select a preferred channel from among the channels in $\mathcal{G}_0$ at the end of the first round (for example, the smallest channel). A node i implicitly conveys this information to all its one-hop neighbors during the second round by transmitting its updated $\mathcal{G}_i$. At the end of the second round, each node i updates the set $\mathcal{G}_i$ as described before and increments r by one. In our example, the updated sets are $\mathcal{G}_1 =$

		Timeslots						
		1	2	3	4	5	6	7
Frames	1	$\{c_1, c_5\}$	$\{c_1, c_5\}$		$\{c_5\}$	$\{c_1, c_3, c_5\}$		
	2	$\{c_1, c_5\}$	$\{c_1, c_5\}$					
	3		$\{c_1, c_5\}$			$\{c_1, c_3, c_5\}$		
	4				$\{c_5\}$	$\{c_1, c_3, c_5\}$	$\{c_4, c_5, c_6\}$	
	5	$\{c_1, c_5\}$	$\{c_1, c_5\}$		$\{c_5\}$	$\{c_1, c_3, c_5\}$	$\{c_4, c_5, c_6\}$	
	6	$\{c_1, c_5\}$			$\{c_5\}$	$\{c_1, c_3, c_5\}$	$\{c_4, c_5, c_6\}$	

Fig. 4. Transmissions during round 2 of phase 1.

$\{c_5\}$, $\mathscr{G}_2 = \{c_5\}$, $\mathscr{G}_4 = \{c_5\}$, $\mathscr{G}_5 = \{c_1, c_5\}$ and $\mathscr{G}_6 = \{c_5\}$. Note that this updated set now gives the set of channels that are common to a node and all other nodes in its 2-hop neighborhood ($\mathscr{L}_i^2$).

3.2.2. Second phase or globally common channel set computation

The second phase of the algorithm consists of $D-2$ frames with each frame divided into N timeslots (see Fig. 1). (Therefore, a round corresponds to a single frame in this phase.) Each node i now transmits only on its preferred channel, PC_i, (that was determined during the first phase) during its preassigned timeslot. This reduces the number of timeslots, and in turn, reduces the time complexity of the algorithm. As in the first phase, each node i continues to transmit its updated set $\mathscr{G}_i$ as illustrated in Fig. 5. At the end of the kth round of the second phase, each node i knows the set of channels that are common to i and all the nodes that are within $(k+2)$ hops from node i. Thus, for $k = D-2$, every node i is aware of the set of channels that are common to i and all the nodes that are within D hops. This corresponds to the globally common channel set for the network and the algorithm terminates. The diameter of the sample network illustrated in Fig. 2 is 3. Thus, phase 2 has only one round (3-2) and the updated sets are $\mathscr{G}_1 = \{c_5\}$, $\mathscr{G}_2 = \{c_5\}$, $\mathscr{G}_4 = \{c_5\}$, $\mathscr{G}_5 = \{c_5\}$ and $\mathscr{G}_6 = \{c_5\}$. Thus $\mathscr{G}_i$ at each node converges to the globally common channel set $\{c_5\}$.

Upon termination, the proposed configuration algorithm yields the following:

- All the nodes are able to identify their one-hop neighbors.
- Every node i learns the set of channels that is common to itself and all nodes within k-hop distance from node i, for each $1 \leqslant k \leqslant D$. This information is very useful when $\mathscr{G}_0 = \emptyset$ to support some communication infrastructure (see Section 6.1).
- Every node i identifies the globally common channel set $\mathscr{G}_0$, and hence, enables the normal operation (that follows the configuration process) to take place on one of the channels in the set $\mathscr{G}_0$ (if non-empty).

If the nodes are allowed to transmit more information, the global network topology and the identities of the nodes present in the network can also be determined.

		Timeslots						
		1	2	3	4	5	6	7
Frames	1	$\{c_5\}$	$\{c_5\}$		$\{c_5\}$	$\{c_1, c_5\}$	$\{c_5\}$	

Node i transmits on its preferred channel PC_i.

Fig. 5. Transmissions during phase 2.

3.3. Diameter-unaware configuration

Consider the case where nodes are not aware of the diameter of the network, D. Note that, even though all the nodes know the set $\mathscr{G}_0$ in D rounds, the nodes lack sufficient local knowledge to know when D rounds have been completed and hence, the nodes do not know when to terminate the configuration algorithm. An obvious way, though inefficient, is to assume that D is $N-1$. However, a more efficient can be devised as discussed next.

3.3.1. Peleg's leader election algorithm when diameter is unknown

Peleg [23] proposed a distributed time-optimal leader election algorithm for arbitrary topologies that runs in $O(D)$ rounds even when the nodes lack the knowledge of D. Recall that the term *round* signifies the duration it takes for every node in the network to communicate with all of its neighbors. In Peleg's algorithm, the node with the largest identifier in the network is elected as the leader. Each node maintains two variables: (1) a variable u that contains the largest identifier it has seen so far, and (2) a variable d that contains the longest distance of any node in the network from node u that it is aware of. Nodes broadcast u and d in every round. A node competes to become the leader in the network as long as it is not aware of a node with larger identifier than its own, that is, its variable u contains its own identifier. Once it learns about a node with larger identifier than its own, it stops competing (to become the leader). If a node has the largest identifier among all nodes in the network, then its variable u will always be set to its own identifier and, further, it will receive *increasing* values of d from successively farther nodes every *second* round. Therefore if a node detects that its variable u is set to its own identifier and its variable d has the same value for three consecutive rounds, then the node can deduce that (1) all nodes in the network have heard from it (directly or indirectly), and vice versa, and (2) it has the largest identifier among all nodes in the network. The node then elects itself as the leader and instructs all other nodes to terminate the leader election algorithm. If $d_{\max}$ is the longest distance of any node in the network from the node with the largest identifier, say $\mathscr{U}\mathscr{I}\mathscr{D}_{\max}$, Peleg's algorithm terminates in at most $(3d_{\max}+2)$ rounds from the time node $\mathscr{U}\mathscr{I}\mathscr{D}_{\max}$ wakes up. (In our case, all nodes wake up at the same time.) Further, $d_{\max} \leqslant D \leqslant 2d_{\max}$. Therefore, Peleg's algorithm runs for at most $(3D+2)$ and at least $\left(\frac{3D}{2}+2\right)$ rounds respectively. For more details on this algorithm, readers are referred to the research note in [23].

As an illustration, consider the sample network in Fig. 2. Node 6 has the largest identifier among all nodes in the network. Node 4 learns about node 6 in round 1. Nodes 1 and 2 learn about node 6 in round 2. Node 5 learns about node 6 in round 3. Node 6 receives new estimates for longest distance in rounds 2, 4 and 6 and elects itself as the leader at the end of round 8.

3.3.2. Using Peleg's algorithm to compute the globally common channel set

We propose to run Peleg's algorithm in parallel with the described layer-2 configuration algorithm to enable all nodes to terminate when they do not know the diameter of the network. To that end, every node maintains the two variables u and d as described before, which are initialized and updated according to Peleg's algorithm. Specifically, these variables are included in every message a node transmits in a round and are updated at the end of every round based on the messages the node has received during that round. Since Peleg's algorithm runs for at least $\left(\frac{3D}{2}+2\right)$ rounds, the modified layer-2 configuration algorithm runs for at least D rounds. We know that $\mathscr{G}_i = \mathscr{G}_0$ at the end of D rounds for every node i in the network. Since i only performs a set intersection operation after every round (as described in Section 3.2), executing the configuration algorithm for more than D rounds will not change $\mathscr{G}_i$ as $\mathscr{G}_j$ for every node j in network would have also converged to $\mathscr{G}_0$ by the end of D rounds. Thus, when a node terminates with respect to Peleg's leader election algorithm, it can also terminate with respect to the configuration algorithm. This implies that the configuration algorithm will terminate in at most $(3D+2)$ rounds at every node in the network with the required set $\mathscr{G}_0$.

3.4. Complexity analysis

The diameter-aware configuration algorithm requires exactly D rounds for completion. Phase 1 of the algorithm consist of two rounds of MN timeslots each. The $(D-2)$ rounds of phase 2 require N timeslots for each round. Thus, the total time-complexity of the configuration algorithm is $2MN+(D-2)N$ timeslots. As each node transmits information related only to its channel availability

Table 1
Running time of our configuration algorithms for various topologies when $N = 40$, $M = 80$, and timeslot duration is 1 ms

Topology	Time-complexity		
	Diameter-aware (s)	Diameter-unaware	
		Minimum (s)	Maximum (s)
Ring	7.12	7.68	8.88
Grid (8×5)	6.76	7.14	7.80
Star	6.40	6.60	6.72
Binary tree	6.68	7.02	7.56

set, the number of bits carried per payload is $O(M)$. For a network in which nodes are arranged in a linear topology ($D = N - 1$) with 40 nodes,[2] 80 channels,[3] and timeslot duration of 1 ms [33], the diameter-aware algorithm terminates within 8 s. (Linear topology has the largest diameter and therefore worst-case running time for a given value of N.) Note that the timeslot duration of 1 ms includes the time required for changing channel frequency, preamble required to establish message bit synchronization, and guard bands for synchronization error and propagation time.[4] Peleg's time-optimal leader election algorithm terminates in at most $3d + 2$ rounds, where $d \leqslant D \leqslant 2d$. Thus, our diameter-unaware configuration algorithm requires $2MN$ timeslots for the first phase and at most $3DN$ additional timeslots for the second phase. Thus, the time-complexity of the diameter-unaware algorithm is $2MN + 3DN$ timeslots. This algorithm requires every node to transmit a channel set, current estimate of the highest identifier and current estimate of the longest distance from the node with the highest identifier to any other node in the network. Thus, it requires $O(M + \log N)$ bits per message payload. Once again, for a linear topology with $N = 40$, $M = 80$, and timeslot duration of 1 ms, the diameter-unaware algorithm terminates within 12 s. Table 1 shows running of the two configuration algorithms for various topologies. For the diameter-unaware version, the exact running time depends on where the node with the largest identifier is located in the network.

[2] In the army, a unit of 30–40 soldiers forms a platoon.

[3] In IEEE 802.11b devices operating in the 2.4 GHz band [9] there are three non-overlapping channels (channels 1, 6 and 11) that are 25 MHz apart. Assuming the same distribution, a 2 GHz band may be divided into 80 non-overlapping channels.

[4] Details on computation of timeslot duration were obtained from personal communication with Jeff Barton, Rockwell Collins Inc.

4. Lower bound on neighbor discovery

We have seen that our algorithms have running times in terms of M and N even though the number of nodes actually present in the network, say n, may often be much less than N. It is natural to suspect that the *oblivious*[5] nature of the algorithms is responsible for running times in M and N. In this section, we investigate the problem of designing algorithms that run in time proportional to n even if we do not restrict ourselves to oblivious algorithms.

We show that any collision-free deterministic algorithm for neighbor discovery has to use a large number of timeslots even when the network contains a small number of nodes. We use a slightly weaker definition for collision, wherein we say that a collision occurs (and is indistinguishable from no transmissions) if two nodes that are within two hops of each other, say i and j, transmit simultaneously on a channel c and there is at least one mutual neighbor k listening on c. If none of the mutual neighbors of i and j listen on c or, if none exists, there is no collision. Under these assumptions, we show that for each channel c available at a node i in the network, any deterministic algorithm to solve the neighbor discovery problem has to allocate at least $N - n$ timeslots during which i listens on c. Thus, any algorithm to solve the neighbor discovery problem has to use at least $(N - n) \times A$ timeslots, where A denotes the maximum size of a channel availability set among all the nodes in the network. This result arises from the *ambiguity* concerning the exact number of nodes that are present in the network and the channels that are available at each of these nodes. It may appear that one of the reasons for a high lower bound on the time-complexity of neighbor discovery is the requirement that the algorithm be collision-free. However, we have recently proved in [20] that a *similar lower bound exists even if collisions are allowed*. Before we proceed to the formal proof, we give some definitions that will be used in the proof.

[5] For oblivious algorithms, the decision whether a node transmits in a given timeslot is solely determined by the node label and the timeslot number [12]. For CR nodes, the identity and availability of the channel also has to be taken into consideration.

Configuration: A *configuration* $\mathscr{C}$ is defined by the set of nodes that are actually present in the network, the channel availability sets at each one of these nodes and the neighborhood relation among these nodes. Formally, a configuration $\mathscr{C}$ is given by a triplet $\langle \Pi, \Lambda, \Gamma \rangle$ where

- Π denotes the set of nodes present in the network,
- Λ denotes the neighborhood function, that is, $\Lambda : \Pi \to 2^{\Pi}$, and
- Γ denotes the channel availability set function, that is, $\Gamma : \Pi \to 2^{\mathscr{A}_{\text{univ}}}$.

Note that, for each node $i \in \Pi$, $\Gamma(i) = \mathscr{A}_i$.

State: The knowledge that a node has gained about the network constitutes the *state* of that node. Thus, the initial state of a node i depends on M, N, $\mathscr{A}_{\text{univ}}$, i and $\Gamma(i)$ only. A node changes state as it learns about its neighbors and other nodes in the network.

Action: The action of a node i in a timeslot t (transmit on a channel c or listen on a channel c or do nothing) is determined solely by the state of the node at the beginning of t and the deterministic algorithm $\mathscr{NDA}$ executing at i.

Execution: A *state assignment* of a network is defined to be an assignment of a state to each node in the network. Also, an action assignment is an assignment of an action to each node in the network. An execution α is then defined by a finite alternating sequence of state assignments and action assignments $S_0, A_1, S_1, A_2, \ldots, S_T$, where each S_r is a state assignment after r timeslots, A_r is an action assignment for timeslot r, and T is the number of timeslots required for the execution to complete.

For a node i, an execution α and a timeslot t, we use $\text{state}(i, \alpha, t)$ to denote the state of node i immediately after timeslot t in execution α. Further, we use $\textit{action}(i, \alpha, t)$ to denote the action of node i during timeslot t in execution α. Clearly, if an algorithm is deterministic then the action of a node during a timeslot only depends on its state at the end of the previous timeslot. Formally,

$$\begin{aligned} \text{state}(i,\alpha,t) = \text{state}(i,\alpha',t) \\ \Rightarrow \quad \text{action}(i,\alpha,t+1) \\ = \text{action}(i,\alpha',t+1). \end{aligned} \tag{1}$$

Let $\text{view}(i, \alpha, t)$ denote the view of node i during timeslot t of execution α. If node i transmits during the timeslot t or listens but does not hear anything or does nothing, then $\text{view}(i, \alpha, t)$ is defined to be $\perp$. Otherwise, $\text{view}(i, \alpha, t)$ is defined to be the message received during the timeslot t. The state of a node at the end of a timeslot only depends on its state at the beginning of that timeslot and its view during that timeslot. Formally,

$$\begin{aligned} (\text{state}(i,\alpha,t) = \text{state}(i,\alpha',t)) \wedge (\text{view}(i,\alpha,t+1) \\ = \text{view}(i,\alpha',t+1)) \\ \Rightarrow \text{state}(i,\alpha,t+1) \\ = \text{state}(i,\alpha',t+1). \end{aligned} \tag{2}$$

Indistinguishability: If α and α' are two executions, we say that α is *indistinguishable* from α' with respect to a node i, denoted by $\alpha \overset{i}{\Longleftrightarrow} \alpha'$, if i has the same initial state and the same sequence of views in α and α'. We also say that α and α' are *indistinguishable to node i up to timeslot t*, denoted by $\alpha \overset{(i,t)}{\Longleftrightarrow} \alpha'$, if i has the same initial state and the same sequence of views up to (but not including) timeslot i in α and α'. Note that if executions α and α' are indistinguishable to node i up to timeslot t, then i has the same action during timeslot t of executions α and α'. Formally,

$$\alpha \overset{(i,t)}{\Longleftrightarrow} \alpha' \quad \Rightarrow \quad \text{action}(i,\alpha,t) = \text{action}(i,\alpha',t). \tag{3}$$

Two executions α and α' are said to be *distinguishable* if they are *not* indistinguishable. We now show the following:

Lemma 4.1. *Any deterministic algorithm for neighbor discovery has a unique execution for every configuration.*

Proof. Assume, on the contrary, that the algorithm has two different executions α and α' for some configuration $\mathscr{C}$. Let t be the first timeslot where the two executions differ, that is, the state assignments at the end of timeslot t with $t \geqslant 1$ in the two executions are different. By definition of t, we have:

$$\forall i : i \in \mathscr{C} : \text{state}(i,\alpha,t-1) = \text{state}(i,\alpha',t-1). \tag{4}$$

From (1), we have:

$$\forall i : i \in \mathscr{C} : \text{action}(i,\alpha,t) = \text{action}(i,\alpha',t) \tag{5}$$

which, in turn, implies that:

$$\forall i : i \in \mathscr{C} : \text{view}(i,\alpha,t) = \text{view}(i,\alpha',t). \tag{6}$$

Combining (4), (6) and (2), we have:

$$\forall i : i \in \mathscr{C} : \text{state}(i,\alpha,t) = \text{state}(i,\alpha',t).$$

This contradicts the assumption that α and α' have different state assignments at the end of timeslot t. □

We now establish the lower bound on the number of timeslots that any collision-free deterministic algorithm, say $\mathcal{NDA}$, has to use for neighbor discovery. Consider a configuration $\mathscr{C} = \langle \Pi, \Lambda, \Gamma \rangle$, a node $i \in \Pi$, a channel $c \in \Gamma(i)$ and a node $j \notin \Pi$. We use $\mathscr{C}\langle i, c, j \rangle$ to denote the configuration that is similar to $\mathscr{C}$ except for the additional node j that is a neighbor of i on channel c. Formally, $\mathscr{C}\langle i, c, j \rangle$ is the configuration $\langle \Pi', \Lambda', \Gamma' \rangle$ with:

$$\Pi' \triangleq \Pi \cup \{j\}$$

$$\Lambda'(k) \triangleq \begin{cases} \Lambda(k) : k \in \Pi - \{i\} \\ \Lambda(k) \cup \{j\} : k = i \\ \{i\} : k = j \end{cases}$$

$$\Gamma'(k) \triangleq \begin{cases} \Gamma(k) : k \in \Pi \\ \{c\} : k = j. \end{cases}$$

Let α denote the unique execution of the algorithm $\mathcal{NDA}$ for the configuration $\mathscr{C}$. We use $\alpha\langle i, c, j \rangle$ to denote the unique execution of the algorithm $\mathcal{NDA}$ for the configuration $\mathscr{C}\langle i, c, j \rangle$. Note that $\alpha\langle i, c, j \rangle$ must contain at least one timeslot during which j transmits on c and i listens on c. Let $t\langle i, c, j \rangle$ denote the *first* such timeslot. It can be verified that α and $\alpha\langle i, c, j \rangle$ are indistinguishable to node i up to timeslot $t\langle i, c, j \rangle$, that is,

$$\alpha \overset{(i, t\langle i,c,j \rangle)}{\Longleftrightarrow} \alpha\langle i, c, j \rangle$$

By definition, node i listens on channel c during timeslot $t\langle i, c, j \rangle$ in execution $\alpha\langle i, c, j \rangle$. It follows using (3) that node i listens on channel c during timeslot $t\langle i, c, j \rangle$ in execution α as well.

Node i listens on channel c during timeslot $t\langle i, c, j \rangle$ in execution α.

We show using the collision-free nature of the algorithm that:

Lemma 4.2. *Consider a configuration $\mathscr{C} = \langle \Pi, \Lambda, \Gamma \rangle$, a node $i \in \Pi$, a channel $c \in \Gamma(i)$, and nodes j and k with $\{j, k\} \cap \Pi = \emptyset$. Let α denote the execution of the algorithm $\mathcal{NDA}$ for $\mathscr{C}$. The first timeslot during which j transmits on c and i listens on c in $\alpha\langle i, c, j \rangle$ (for configuration $\mathscr{C}\langle i, c, j \rangle$) is not the same as the first timeslot during which k transmits on c and i listens on c in $\alpha\langle i, c, k \rangle$ (for configuration $\mathscr{C}\langle i, c, k \rangle$) unless j and k are the same nodes. Formally,*

$$t\langle i, c, j \rangle = t\langle i, c, k \rangle \quad \Rightarrow \quad j = k$$

Proof. Assume, on the contrary, that $t\langle i, c, j \rangle = t\langle i, c, k \rangle$ but $j \neq k$. Consider the configuration $\mathscr{D}$ obtained from $\mathscr{C}$ by adding the two nodes j and k as neighbors of node i on channel c. (Intuitively, $\mathscr{D}$ is the "union" of the two configurations $\mathscr{C}\langle i, c, j \rangle$ and $\mathscr{C}\langle i, c, k \rangle$.) Let β be the execution of the algorithm $\mathcal{NDA}$ for the configuration $\mathscr{D}$. Clearly, β contains at least one timeslot during which either node j or node k transmits on channel c and node i listens on channel c during the same timeslot. Let t denote the *first* such timeslot, and let $t_{\min} = \min\{t, t\langle i, c, j \rangle\}$. Since neither j nor k transmits on c while i listens on c until t in all the three executions – $\alpha\langle i, c, j \rangle$, $\alpha\langle i, c, k \rangle$ and β, it can be shown that:

$$\alpha\langle i, c, j \rangle \overset{(i, t_{\min})}{\Longleftrightarrow} \beta \overset{(i, t_{\min})}{\Longleftrightarrow} \alpha\langle i, c, k \rangle, \quad \alpha\langle i, c, j \rangle \times \overset{(j, t_{\min})}{\Longleftrightarrow} \beta \quad \text{and} \quad \alpha\langle i, c, k \rangle \overset{(k, t_{\min})}{\Longleftrightarrow} \beta. \tag{8}$$

Without loss of generality, assume that it is j that transmits on c during t in β. It can be proved using (8) that j transmits on c during $t_{\min}$ in $\alpha\langle i, c, j \rangle$ as well as β. Further, i listens on c during $t_{\min}$ in $\alpha\langle i, c, j \rangle$ as well as β. This implies that $t_{\min} = t\langle i, c, j \rangle = t$, which, in turn, means that k transmits on c during t in β as well. This causes transmissions by j and k on c to collide with each other while i is listening on c during t in β. □

By using the fact that each node has a single transceiver, we show that:

Lemma 4.3. *Consider a configuration $\mathscr{C} = \langle \Pi, \Lambda, \Gamma \rangle$ and a node $i \in \Pi$. Let α denote the execution of the algorithm $\mathcal{NDA}$ for the configuration $\mathscr{C}$. Then α contains at least $(N - n) \times |\Gamma(i)|$ timeslots.*

Proof. From Lemma 4.2 and (7), for each channel $c \in \Gamma(i)$, α contains at least $(N - n)$ timeslots during which node i listens on channel c. Since each node has a single transceiver, the set of timeslots for different channels are mutually exclusive. □

By choosing node i to be the node with the largest availability set among all nodes in the network, we obtain:

Theorem 4.4. *Any collision-free deterministic algorithm that solves the neighbor notification problem*

has to use at least $(N - n) \times A$ *timeslots, where N is the maximum number of nodes in the network, n is the actual number of nodes in the network, and A is the maximum size of an availability set among all nodes in the network.*

When $n = o(N)$ and $A = \Omega(M)$, we obtain a lower bound of $\Omega(MN)$ timeslots for any collision-free deterministic algorithm solving the neighbor discovery problem in the single transceiver model.

Note that every configuration that we have constructed during the course of our proof contains at most two more nodes in addition to the nodes in the given configuration. Therefore, the lower bound holds even if all nodes know a better (but not exact) upper bound n_u than N on the actual number of nodes n in the network, that is, $n + 2 \leqslant n_u \leqslant N$. For example, assume that each node in the network is assigned a unique identifier from the range $[1, \ldots, 1000]$. However, the network contains only 10 nodes and each node knows that there are at most 12 nodes in the network. Moreover, instead of assuming that each node is aware of an upper bound on the number of nodes in the network, we can assume that each node is aware of an upper bound on the number of nodes in its neighborhood, that is, its *degree*. It can be verified that as long as the known upper bound on the degree is imprecise, the lower bound proof in this section of $\Omega(MN)$ timeslots is still applicable.

5. Speeding up neighbor discovery and configuration with limited channel divergence

So far, we have implicitly assumed that channel availability sets of two neighboring nodes may differ arbitrarily. In case it is possible to bound the extent by which channel availability sets of two neighboring nodes can differ, we can devise a faster algorithm for neighbor discovery and configuration (first phase). Such a situation of limited divergence between neighbors may arise in scenarios where the CR network is deployed in regions that have a scarce population of primary users and/or other sources of interference. Assume that the sets of channels available to any pair of neighboring nodes i and j differ by at most Δ, that is, $|\mathscr{A}_i - \mathscr{A}_j| \leqslant \Delta$. (The set difference $A - B$ is defined as: $A - B = \{x : x \in A \text{ and } x \notin B\}$.) Each node i is assumed to be aware of the parameter Δ in addition to N, M, $\mathscr{A}_{\text{univ}}$, and $\mathscr{A}_i$. For example, consider the sample network shown in Fig. 6. The channels available at each node are as indicated in the figure. Clearly, $|\mathscr{A}_i - \mathscr{A}_j| \leqslant \Delta = 2$ for all neighbor pairs i and j (as illustrated in the figure).

We now describe our algorithm. Consider two neighboring nodes i and j. Since $\mathscr{A}_i$ and $\mathscr{A}_j$ differ by at most Δ channels, if i transmits on any subset of $(\Delta + 1)$ channels of $\mathscr{A}_i$, there is at least one channel on which j can receive the message *provided* j tunes to the correct channel(s) in the appropriate timeslot(s). We now present a scheme that ensures that the above condition is satisfied.

A round is divided into N blocks, one per node. Each block is further divided into $(\Delta + 1)$ segments, each of which contains $(\Delta + 1)$ timeslots. During the ith block of a round, node i is allowed to transmit on $(\Delta + 1)$ *lower-most* channels sorted by frequency. Specifically, in the xth segment of the ith block, node i transmits on the xth lower-most channel $(\Delta + 1)$ times. On the other hand, in the xth segment of the ith block, nodes other than i listen on their $(\Delta + 1)$ lower-most channels, one by one.

Let $\{c_i^1, c_i^2, \ldots, c_i^{\Delta+1}\}$ be the set of $(\Delta + 1)$ lower-most channels available at some node i. Let $\{c_j^1, c_j^2, \ldots, c_j^{\Delta+1}\}$ be the set of $(\Delta + 1)$ lower-most channels available at a neighbor j of i. Since the difference in the availability sets is at most Δ, the two channel

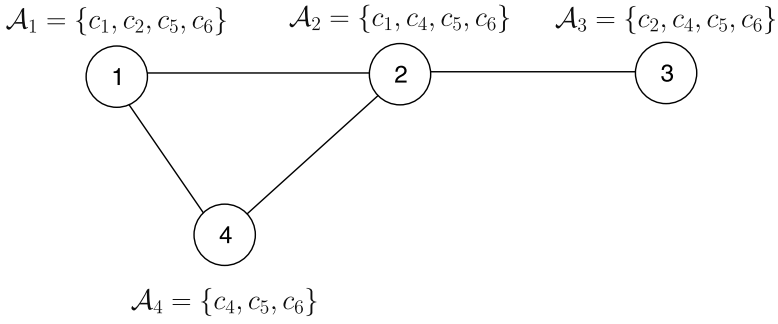


Fig. 6. A sample CR network with $N = 5$, $M = 6$ and $\Delta = 2$.

		Timeslots								
		1	2	3	4	5	6	7	8	9
Blocks	1	**(R,1)**	(R,4)	(R,5)	(R,1)	(R,4)	(R,5)	(R,1)	(R,4)	**(R,5)**
	2	**(T,1)**	(T,1)	(T,1)	**(T,4)**	**(T,4)**	(T,4)	(T,5)	**(T,5)**	**(T,5)**
	3	(R,1)	(R,4)	(R,5)	(R,1)	**(R,4)**	(R,5)	(R,1)	(R,4)	**(R,5)**
	4	(R,1)	**(R,4)**	(R,5)	(R,1)	(R,4)	**(R,5)**	(R,1)	(R,4)	(R,5)
	5	(R,1)	(R,4)	(R,5)	(R,1)	(R,4)	(R,5)	(R,1)	(R,4)	(R,5)

(T,x): transmission on channel c_x

(R,x): reception on channel c_x

Fig. 7. Communication schedule for node 2 in the sample network in Fig. 6. Successful communication is shown in bold.

sets, namely $\{c_i^1, c_i^2, \ldots, c_i^{\Delta+1}\}$ and $\{c_j^1, c_j^2, \ldots, c_j^{\Delta+1}\}$ have at least one common channel. Let that common channel be c_i^x. It can be verified that, in the xth segment of the ith block, at least one transmission by i will be *successfully* received by j.

As an example, consider node 2 in Fig. 6. The transmission schedule for 2 is illustrated in Fig. 7. A tuple of the form (T,x) is used to indicate that a node is transmitting on a channel c_x in a given timeslot. A tuple of the form (R,x) indicates that a node is receiving on a channel c_x. We have used only the indices of the channels for simplicity of representation. Transmissions that are received by at least one neighbor are marked in bold. It is easy to see that there is at least one successful transmission from 2 to all of its neighbors in Fig. 6.

Clearly, each round of the first phase consists of $(\Delta+1)^2 N$ timeslots instead of MN timeslots. For $\Delta \ll M$, this is a significant improvement over the algorithm in Section 3.2.1.

6. Discussion

6.1. Comments on empty globally common channel set

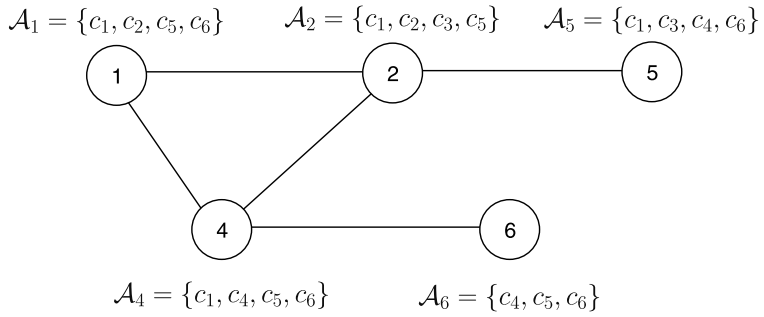
The globally common channel set $\mathcal{G}_0$ may be empty in certain situations. However, that does not preclude communication among the nodes since the only condition that needs to be satisfied for a node i to communicate with its neighbors is:

$$\mathcal{A}_i \bigcap \left(\bigcap_{j \in \mathcal{NBR}_i} \mathcal{A}_j \right) \neq \emptyset.$$

Consider the sample network shown in Fig. 8. This network differs from the network illustrated in Fig. 2 only in the availability set at node 5 ($c_5 \notin \mathcal{A}_5$). This minor change however renders the network with an empty globally common channel set since c_5 was the only common channel in Fig. 2. The progress of the configuration algorithm for the network is also illustrated in Fig. 8. At the end of phase 2, each node i will update its $\mathcal{G}_i$ and find that it is empty. If $\mathcal{G}_i$ at the end of phase 2 is empty, node i can always revert back to the last non-empty $\mathcal{G}_i$ that was computed at the end of a round. This can be done by maintaining a record of all the $\mathcal{G}_i$ computed during execution of the algorithm. The $\mathcal{G}_i$ computed at the kth round is represented by $\mathcal{G}_i^k$. Consider node 5 in the example shown in Fig. 8. Here, $\mathcal{G}_5^1 = \{c_1, c_3\}$, $\mathcal{G}_5^2 = \{c_1\}$ and $\mathcal{G}_5^3 = \{\}$. Thus, node 5 can deduce that channel c_1 is common to itself and all other nodes that are at most two hops away. Similarly, node 6 can deduce that c_5 is common to itself and all other nodes that are at most two hops away. Nodes 1, 2, 4 and 5 can form a cluster and communicate among themselves on c_1 while nodes 1, 2, 4 and 6 can form a cluster and communicate among themselves on c_5. For inter-cluster communication, node 2 or node 4 can act as a gateway node as they can communicate with both clusters.

6.2. Comments on configuration overheads

Layer-2 configuration operation and normal operation are repeated periodically every T time units. Alternating between layer-2 configuration and normal operation requires that the normal



			Timeslots						
			1	2	3	4	5	6	7
Frames	Round 1	1	$\{c_1, c_2, c_5, c_6\}$	$\{c_1, c_2, c_3, c_5\}$		$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_6\}$		
		2	$\{c_1, c_2, c_5, c_6\}$	$\{c_1, c_2, c_3, c_5\}$					
		3		$\{c_1, c_2, c_3, c_5\}$			$\{c_1, c_3, c_4, c_6\}$		
		4				$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_6\}$	$\{c_4, c_5, c_6\}$	
		5	$\{c_1, c_2, c_5, c_6\}$	$\{c_1, c_2, c_3, c_5\}$		$\{c_1, c_4, c_5, c_6\}$		$\{c_4, c_5, c_6\}$	
		6	$\{c_1, c_2, c_5, c_6\}$			$\{c_1, c_4, c_5, c_6\}$	$\{c_1, c_3, c_4, c_6\}$	$\{c_4, c_5, c_6\}$	
	Round 2	1	$\{c_1, c_5\}$	$\{c_1\}$		$\{c_5\}$	$\{c_1, c_3\}$		
		2	$\{c_1, c_5\}$	$\{c_1\}$					
		3		$\{c_1\}$			$\{c_1, c_3\}$		
		4				$\{c_5\}$	$\{c_1, c_3\}$	$\{c_4, c_5, c_6\}$	
		5	$\{c_1, c_5\}$	$\{c_1\}$		$\{c_5\}$		$\{c_4, c_5, c_6\}$	
		6	$\{c_1, c_5\}$			$\{c_5\}$	$\{c_1, c_3\}$	$\{c_4, c_5, c_6\}$	
	Round 3	1	$\{\}$	$\{\}$		$\{\}$	$\{c_1\}$	$\{c_5\}$	

Fig. 8. Algorithm execution when set $\mathcal{G}_0$ is empty.

operation be suspended for some time every T time units. This may disrupt ongoing higher layer communication (for example, TCP connections), which may be a high penalty to pay, especially for networks with low mobility. So, instead of alternating between these two modes of operation, it would be better to interleave the configuration rounds between the normal operation so that configuration process is continuously ongoing. For this,

- The normal operation may also need to have a slotted and framed structure. This is to ensure that the context switching (which is more frequent here) may be done in a manner that is independent of the communication algorithm used for normal operation.
- Traffic-bearing slots during normal operation may be longer than the timeslot duration of the proposed configuration algorithm. One could possibly pack several configuration slots into "borrowed" traffic slots.

6.3. Comments on changes to the globally common channel set

Some of the factors that affect the integrity of the globally common channel set $\mathcal{G}_0$ computed by the proposed layer-2 configuration algorithm are:

- Nodes may not turn on their radios at the same time, and hence, may invoke the configuration algorithm at different times.

- Network topology changes. New nodes in the network could arrive or the existing nodes from the network could depart at any time. Thus, it is possible that a single network could become partitioned and one or more such partitions could merge later to form a single network.
- Changes to the channel availability set maintained by individual nodes (possibly due to arrival of the primary user of the globally common channel, c_{global}) will also trigger re-computation of the globally common channel set.

To address changes to the set $\mathscr{G}_0$ due to all the above factors, the configuration algorithm may have to be re-invoked. Consider a newly arrived node. If the newly arrived node decreases the cardinality of the set $\mathscr{G}_0$ by at least one, then we term it a *contributing node*. Suppose a run of the configuration algorithm resulted in the selection of c_{global} for communication among existing nodes. When a contributing node (say j) arrives, it would not be able to communicate with the existing nodes in the network if $c_{\text{global}} \notin \mathscr{A}_j$. Thus, nodes in the neighborhood of j would remain unaware of j's arrival. Due to this lack of knowledge, they would have to scan through all the M channels in $\mathscr{A}_{\text{univ}}$. Also, the total number of new nodes joining the network is not known a priori. Thus, a time-slotted mechanism appears to be needed, whereby each node transmits in its pre-assigned timeslot. A total of MN timeslots would be required to detect a newly arrived node and learn about its availability set. After this, $O(D)$ rounds would be required to propagate this information throughout the network. This is equivalent to re-invoking the configuration algorithm. Thus, in order to handle changes to globally common channel c_{global} and/or the set $\mathscr{G}_0$, every node in the network re-invokes the layer-2 configuration algorithm every T time units (see Fig. 1), where T is significantly larger than the time required for the configuration algorithm to terminate. For example, we may choose T to be equal to $20 \times$ time taken by the configuration algorithm.

6.4. Comments on lossy channels

We have assumed so far that the channels are reliable: if a node i transmits a message on a channel on which a neighboring node j is listening, then j receives the message sent by i if no other neighbor of j is transmitting on the same channel simultaneously. However, messages may also be lost because of transient noise in the medium. Typically, two neighboring nodes are likely to share many channels. Therefore, in the first phase of the layer-2 configuration algorithms described in Section 3, every node should be able to successfully receive a message from each of its neighboring nodes on at least one of the channels shared by them with reasonably high probability. To improve the robustness of the second phase, we can simply execute a few additional rounds (the number will depend on how lossy the channels are), to increase the likelihood that all nodes have correctly computed the globally common channel set. Also, instead of making one transmission (as in the algorithm proposed by us) during a timeslot, the transmitting node can be programmed to transmit a fixed number of transmissions of the same packet (interleaved with a very short silence period) during a timeslot. This increases the duration of the timeslot by a constant factor, independent of the number of nodes or number of channels, but the robustness of the reception is substantially increased without any increase in the asymptotic complexity. When using the neighbor discovery and configuration algorithm that assumes limited channel divergence, we can again choose a sufficiently high value for Δ such that two neighboring nodes are likely to share many channels in their availability sets.

6.5. Comments on stable network topology

The algorithm described in Section 3 allows only one transmission per timeslot, the primary reason being that since the nodes know nothing about the network topology, it will not be possible to assign a deterministic conflict-free transmission schedule. At the end of a single execution of the algorithm, the nodes become fully aware of the global network topology. In situations where the network topology is fairly stable (channel availability is still subject to change), repeating the algorithm will be inefficient as the nodes can potentially take advantage of the available network topology. Note that two nodes i and j that are not within each other's radio range as well have no nodes in common among their neighbors can transmit simultaneously without any conflicts. The problem of assigning such timeslots is equivalent to the *distance-2 node coloring* problem on graphs, which is known to be *NP-complete* [18]. A number of heuristics and approximation algorithms have been proposed for distance-2 coloring-based timeslot/channel assignment [15,24,29].

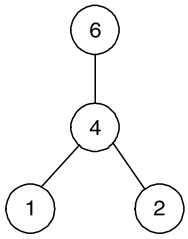


Fig. 9. A star topology.

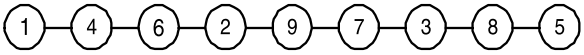


Fig. 10. A linear topology.

Although, distance-2 coloring-based timeslot assignment in a topology-aware setting may not yield an algorithm that runs faster in the worst-case (*e.g.*, star topology), it will perform better for most arbitrary graphs and will in fact be linear in M for certain graphs (*e.g.*, linear topology). To elaborate, consider the star network shown in Fig. 9. When the node at the center of the star (4) transmits, none of the other nodes can transmit because it will result in a collision. Similarly when two or more nodes in the periphery of the network (1, 2 and 6) transmit simultaneously, collision occurs at node 4. Thus, only one node can transmit in a single timeslot and knowledge of the topology does not yield a shorter schedule. Now consider the linear chain illustrated in Fig. 10. One can easily observe that nodes 1, 2 and 3 can transmit simultaneously in a timeslot without any collisions and so can nodes 4, 9 and 8. Thus, up to $\frac{N}{3}$ transmissions are possible per timeslot yielding a running time of $6M$ (linear in M only) for phase 1 instead of $2MN$.

7. Conclusion

In this paper, we addressed the layer-2 auto-configuration problem in a CR network and presented a distributed algorithm for finding a globally common channel set wherein nodes have no prior knowledge of their neighborhood. Our algorithm consists of two phases. In the first phase, every node learns its neighborhood information and selects a preferred channel for transmission in its 1-hop neighborhood. In the second phase, nodes exchange messages on the preferred channels to compute the globally common channel set ($\mathscr{G}_0$). We showed that all nodes in the network determine the $\mathscr{G}_0$ in $2MN + O(DN)$ timeslots for both diameter-aware and diameter-unaware versions of the algorithm. For reasonable network deployment scenarios, the time taken for the diameter-aware and diameter-unaware versions of the algorithm is approximately 8 s and 12 s, respectively. The proposed solution also provides every node the set of channels that are common to itself and all other nodes that are k-hops away. This information is particularly useful when the globally common channel set is empty to facilitate a communication infrastructure among clusters of nodes connected through gateways. We further showed that the proposed algorithm is optimal for neighborhood discovery by proving a lower bound of $\Omega(MN)$ timeslots in the worst-case even when the neighbor discovery algorithm is adaptive in nature. Finally, we presented a faster algorithm for neighbor discovery when channel availability sets between neighboring nodes cannot differ arbitrarily.

In the future, we plan to focus on proving lower bounds for phase 2, and developing algorithms/heuristics that may execute faster for various special scenarios (*e.g.*, stable network topology). We also plan to investigate the layer-2 configuration problem when nodes can distinguish between background noise and collision noise. In Section 6.3, we described a simple but probably inefficient method to address changes in $\mathscr{G}_0$. We propose to develop asynchronous algorithms that can run in conjunction with normal operation of the network to handle changes in network topology and channel availability and the resulting changes in $\mathscr{G}_0$ in a more efficient manner.

Acknowledgement

We would like to thank Jeff Barton of Rockwell Collins Inc., and anonymous referees for their valuable inputs and suggestions.

References

[1] I.F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Computer Networks 47 (4) (2005) 445–487.
[2] G. Alonso, E. Kranakis, C. Sawchuk, R. Wattenhofer, P. Widmayer. Probabilistic protocols for node discovery in ad hoc multichannel broadcast networks, in: Proceedings of the Second Annual Conference on Adhoc Networks and Wireless (ADHOCNOW), 2003, pp. 101–115.
[3] R.W. Broderson, A. Wolisz, D. Cabric, S.M. Mishra, D. Willkomm. CORVUS: A Cognitive Radio Approach for Usage of Virtual Unlicensed Spectrum. Available at <http://

bwrc.eecs.berkeley.edu/Research/MCMA/CR_White_paper_final1.pdf/>.
[4] S. Corson, J. Macker. Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations, RFC 2501, IETF Network Working Group, January 1999.
[5] Federal Communications Commission, FCC Spectrum Policy Task Force Report, ET Docket No. 02-135, November 2002.
[6] Federal Communications Commission. FCC Spectrum Policy Task Force Report, ET Docket No. 02-155, November 2002.
[7] M. Grossglauser, D.N.C. Tse, Mobility increases the capacity of ad hoc wireless networks, IEEE/ACM Transactions on Networking 10 (4) (2002) 477–486.
[8] J. Haartsen, Bluetooth Baseband Specification version 1.0. <http://www.bluetooth.com/>.
[9] Institute of Electrical and Electronics Engineers, IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical (PHY) Layer Specifications, 1999, IEEE 802.11b.
[10] N. Jain, S. Das, A. Nasipuri. A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks, in: Proceedings of the IEEE International Conference on Computer Communications and Networks (IC3N), Phoenix, Arizona, USA, October 2001.
[11] M. Kodialam, T. Nandagopal. Characterizing the capacity region in multi-radio, multi-channel wireless mesh networks, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), August 2005.
[12] D.R. Kowalski, A. Pelc, Time complexity of radio broadcasting: adaptiveness vs. obliviousness and randomization vs. determinism, Theoretical Computer Science 333 (3) (2005) 355–371.
[13] S. Krishnamurthy, M. Thoppian, S. Kuppa, S. Venkatesan, R. Chandrasekaran, N. Mittal, R. Prakash. Time-efficient layer-2 auto-configuration for cognitive radios, in: Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and Systems, Phoenix, Arizona, USA, November 2005, pp. 459–464.
[14] S. Krishnamurthy, M. Thoppian, S. Venkatesan, R. Prakash. Control channel based MAC-layer configuration, Routing and situation awareness for cognitive radio networks, in: Proceedings of the IEEE Military Communications Conference (MILCOM), October 2005.
[15] S.O. Krumke, M.V. Marathe, S.S. Ravi, Models and approximation algorithms for channel assignment in radio networks, Wireless Networks 7 (6) (2001) 575–584.
[16] C. Law, A.K. Mehta, K.-Y. Siu. Performance of a new bluetooth scatternet formation protocol, in: Proceedings of the Second ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2001, pp. 183–192.
[17] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee, R. Morris. Capacity of ad hoc wireless networks, in: Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), July 2001, pp. 61–69.
[18] S.T. McCormick, Optimal approximation of sparse hessians and its equivalence to a graph coloring problem, Mathematical Programming 26 (2) (1983) 153–171.
[19] J. Mitola III. Cognitive radio: an integrated agent architecture for software defined radio, Ph.D. Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, May 2000.
[20] N. Mittal, S. Krishnamurthy, R. Chandrasekaran, S. Venkatesan. A fast deterministic algorithm for neighbor discovery in multi-channel cognitive radio networks. IEEE Transactions on Parallel and Distributed Systems (TPDS), submitted for publication.
[21] A. Nasipuri, S.R. Das. Multichannel CSMA with signal power-based channel selection for multihop wireless networks, in: Proceedings of the Vehicular Technology Conference, September 2000.
[22] A. Nasipuri, J. Zhuang, S.R. Das, A multichannel CSMA MAC protocol for multihop wireless networks, in: Proceedings of the IEEE Wireless Communications and Networking Conference, September 1999.
[23] D. Peleg, Time-optimal leader election in general networks, Journal of Parallel and Distributed Computing (JPDC) 8 (1) (1990) 96–99.
[24] R. Ramaswami, K.K. Parhi, Distributed scheduling of broadcasts in a radio network, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), 1989, pp. 497–504.
[25] A. Raniwala, T. Chiueh, Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), March 2005.
[26] A. Raniwala, K. Gopalan, T. Chiueh, Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks, ACM SIGMOBILE Mobile Computing and Communications Review 8 (2) (2004) 50–65.
[27] T. Salonidis, P. Bhagwat, L. Tassiulas. Proximity awareness and fast connection establishment in bluetooth, in: Proceedings of the First ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2000, pp. 141–142.
[28] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire. Distributed topology construction of bluetooth personal area networks, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), April 2001, pp. 1577–1586.
[29] A. Sen, M.L. Huson, A new model for scheduling packet radio networks, Wireless Networks 3 (1) (1997) 71–82.
[30] F. Siegemund, M. Rohs, Rendezvous layer protocols for bluetooth-enabled smart devices, in: Proceedings of the First International Conference on Architecture of Computing Systems (ARCS) – Trends in Network and Pervasive Computing, 2002, pp. 256–273.
[31] J. So, N.H. Vaidya, Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver, in: Proceedings of the Fifth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2004, pp. 222–233.
[32] J. So, N.H. Vaidya. Routing and channel assignment in multi-channel multi-hop wireless networks with single network interface, in: Proceedings of the Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine), August 2005.
[33] TCI International, Incorporated. TCI 8067 Spectrum processor data specification, 2000. Available at <http://www.tcibr.com/PDFs/8067webs.pdf/>.

[34] M. Thoppian, S. Venkatesan, R. Prakash, R. Chandrasekaran. MAC-Layer scheduling in cognitive radio based multihop wireless networks, in: Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2006, pp. 191–202.
[35] S. Wu, C. Lin, Y. Tseng, J. Sheu. A new multi-channel MAC protocol with on-demand channel assignment for multihop mobile ad hoc networks, in: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), 2000, pp. 232–232.

Srinivasan Krishnamurthy received his Ph.D. in Computer Engineering from The University of Texas at Dallas in December 2006. Since that time, he has been with Cisco Systems, Inc., where is currently a Software Engineer. His research interests include design and analysis of algorithms, cognitive radio networks, fault tolerance in telecom networks and mobile computing.

Mansi Thoppian received the B.Tech. degree in computer science and engineering from the Regional Engineering College, Calicut, India, in June 1998 and the M.S. and Ph.D. degrees in Computer science from The University of Texas at Dallas in 2002 and 2006, respectively. She is currently employed at Cisco Systems Inc, San Jose. Her areas of interest include wireless ad hoc networks, mobile computing, and distributed computing.

Srikant Kuppa received the Bachelor of Technology degree in Computer Technology from Nagpur University, Nagpur, India, in August 1999 and the Master of Science degree in Computer Science from University of Texas at Dallas, Richardson, USA, in December 2001. He worked for Nortel Networks from January to May 2001. He was also a graduate research intern in Intel Labs (Corporate Technology Group) from May to December 2005. In May 2006, he received his Doctorate degree from the University of Texas at Dallas for the work he did towards characterizing the expected performance of IEEE 802.11 DCF and its QoS enhancements. He is currently working for Wireless Networking Business Unit of Cisco Systems Inc., Richfield, OH, USA. His research interests include performance modeling and analysis of network protocols. Other interests include IEEE 802.11 flavors, wireless mesh networks, sensor networks, cognitive radios and Linux kernel hacking.

R. Chandrasekaran has a B.Tech. from Indian Institute of Technology, Bombay and a Ph.D. from UC Berkeley. He has been a faculty member at Case Western Reserve, Northwestern and UT Dallas where he is a Professor of Computer Science and Operations Research. His interests include mathematical programming, combinatorial optimization, scheduling and computational geometry.

Neeraj Mittal received his B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi in 1995 and the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin in 1997 and 2002, respectively. He is currently an assistant professor in the Department of Computer Science at the University of Texas at Dallas and a co-director of the Advanced Networking and Dependable System Laboratory (ANDES). His research interests include distributed systems, wireless networks, and security.

S. Venkatesan received his B.Tech. degree in Civil Engineering and M.Tech. degree in Computer Science from the Indian Institute of Technology, Madras in 1981 and 1983, respectively. He completed his Ph.D. degree in Computer Science from the University of Pittsburgh in December 1988. In January 1989, he joined the University of Texas at Dallas where he is currently an Associate Professor of Computer Science and Head of Telecommunications Engineering Program. His research interests are in Distributed Systems, Sensor Networks, Cognitive Radio Networks, Mobile and Wireless Networks and Testing and Debugging Distributed Programs. He has been the Chief Architect of IPmobile (a startup acquired by Cisco Systems in September 2000) and a member of the advisory board of Jahi Networks (a startup acquired by Cisco Systems in December 2004).

Ravi Prakash received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi in 1990 and the M.S. and Ph.D. degrees in computer and information science from The Ohio State University, Columbus, in 1991 and 1996, respectively. He joined the Department of Computer Science, University of Texas at Dallas in July 1997, where he is an Associate Professor. During 1996–1997 he was a Visiting Assistant Professor in the Computer Science

Department, University of Rochester. His areas of research are mobile computing, distributed computing, wireless networks, and sensor networks. He has published his results in various journals and conferences and has been involved in the organization of various IEEE and ACM conferences and workshops as technical program chair, technical program committee member, etc. He is a recipient of the National Science Foundation CAREER grant. He is currently an associate editor of the IEEE Transactions on Mobile Computing.