Proactive Channel Allocation for Ad-Hoc Networks

Yosi Ben-Asher[1] **and Yehuda Ezra**[1]
[1]CS Department, University of Haifa, Haifa, Israel

Abstract— *In this work we consider the problem of creating multi-edge channels in ad-hoc networks wherein each node can use multiple frequencies for sending and receiving packets by filling the slots of an OFDMA matrix. We consider the problem of how to fill-up the slots of a $frequency \times time$ matrix (FTM) at each node such that maximal collision free $k > 1$ path cover (CFkPC) of the communication graph is obtained. This problem is a variant of edge and node disjoint path cover in graphs extended to include collisions caused by the hidden terminal problem where two nodes A and B which are not in transmission range collide by transmitting at the same frequency and time to an intermediate node C. The proposed solution for filling the FTM is based on using maximum independent set in a suitable conflict graph. Our simulations compare between the proposed CFkPC approach and a version of the DCA protocol [12] that was extended to use multiple send/receive slots. The results show significant advantage of using CFkPC versus the adaptive approach of the extended DCA. We assume that GPS coordinate and GPS time synchronization is available.*

Keywords: Wireless, Ad hoc, multi-channel,routing

1. Introduction

We consider the possibility of sending and receiving packets in multi-frequency channels by the mobile nodes of Ad-Hoc networks. In Ad-Hoc networks, due to movements of users, the connections between nodes frequently change creating different topologies of the communication graph between the nodes of the network. This creates a challenging situation for suitable protocol that can decide, for a given node v, in which frequencies it should transmit and at what time slot such that minimal latency and maximal throughput of packets is obtained. We assume that all nodes use a GPS for location and perfect time synchronization. We model the use of multi-frequencies channels in ad-hoc networks by assuming that each node v needs to fill up slots in a $frequency \times time$ matrix M (called FTM) indicating whether it will transmit $M^v_{f,t} = send$ or receive $M^v_{f,t} = receive$ a packet at this $frequency \times time$ slot. FTM is repeatedly used by each node for sending and receiving packets and thus it is an abstraction of a strategy for using multi frequency channels. Depending on how the FTM at each node are filled, communication channels between neighboring nodes are generated. Thus if $u \longrightarrow v$ are two nodes in communication range and $M^u_{f,t} = send, M^v_{f,t} = receive$, then possibly u can transmit packets to v at this $frequency \times time$ slot. The channel created between u and v by setting $M^u_{f,t} = send, M^v_{f,t} = receive$ is subject to two problems:

- Frequent movements of u and v may move/remove u and v in/out-of communication range.
- Transmissions of other neighboring nodes at the same $frequency \times time$ slot can collide with packets sent by u (known as the hidden terminal problem).

The goal in this work is to find a strategy for filling $M^v_{f,t}$ such that we maximize the number of channels of length $k > 1$ while minimizing packet-lost caused by collisions (due to the hidden terminal problem [2]).

For the purpose of this paper a channel is an agreement between two nodes u and v to use a certain frequency to transmit packets from u to v in a given frequency for a certain period of time or another terminating condition. As such it is possible to generalize the channel from a single hop to a multi-hop channel where u sends packets to v through a sequence of nodes which are all part of the channel agreement. Consider for example a tree like communication graph that has seven nodes given in figure 1. Each node x, u, w, r, z, v, y has the ability to use two frequencies and two time slots (depicted by the 2×2 FTMs in figure 1). Following the previous discussion we would like to configure the slots (transmit/receive) such that multi-hop communication paths will be generated. These communication paths will allow us to pipeline packets of data streams to longer distances faster than what could be obtained had we used single edge channels. Practically, pipelining or streaming of messages implies that packets are transmitted along such a path without acknowledgment from the receiver. The communication paths that have been selected to cover the communication graph of figure 1 contain two paths $p0, p1$ each containing four edges.

We compare two strategies for channel allocation in ad hoc networks:
Extended Dynamic Channel Allocation (EDCA) is a strategy where by using a control channel with request-to-send (RTS) and clear-to-send (CTS) messages, a node u can negotiate with a neighboring node v for a slot $M^{u/v}_{f,t}$ that is currently free. By snooping on other requests on the control channel, both u, v can verify that $M^{u/v}_{f,t}$ is not used by any neighboring node of u or v (possibly preventing collision with other nodes). EDCA is an extension of a well known protocol DCA that have been extensively used in other works. Another extension of DCA is the multi channel

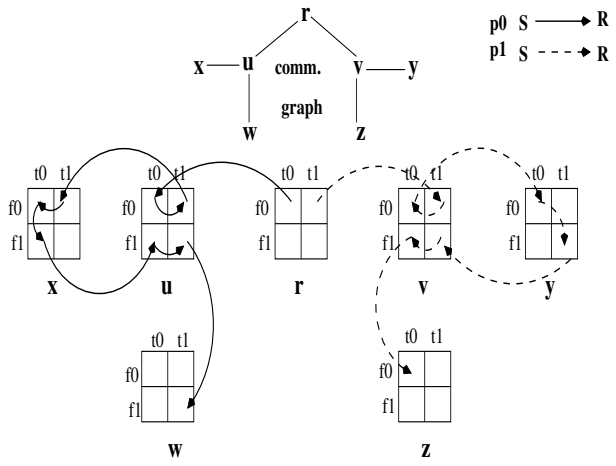

Fig. 1: Configuring $frequency \times time$ slots to create multi-channel paths for streaming of packets

DCA [12] which allowed a node dynamically jumps from one frequency to another according to the traffic of the network, but still do not allow transmitting or receiving on multi channels simultaneously. Thus, in this work we extend the original DCA algorithm [15] to handle receiving and transmitting packets through multi set of channels at same time slot.

Proactive Channel Allocation (PCA) is what we propose here, a strategy based on finding a path-cover by a maximal number of paths of length k such that each path passes through $2k$ transmit/receive slots of the FTMs available at each node. For example the path-cover depicted in figure 1 is a solution to the coverage problem for $k = 4$ as follows:

- There is no legal cover with more than two paths of length $k = 4$ (must be verified by checking all possible cases). This is also a maximal cover since no path, even of length one, can be added to it.
- Each path forms a legal scheduling for pipelining a stream of packets traversing along this path edges (as can be seen there are edges that seems to go back in time but this will be explained later on).
- There are no interferences between the paths, i.e., no node can receive more than one packet at any f_i, t_j slot. For example, node v is in communication range with nodes r, y, z and is scheduled to receive a packet at $f0, t1$ and at $f1, t1$. Indeed only node r broadcasts at $f0, t1$ and only node y broadcasts at $f1, t1$.

As opposed to the EDCA approach the channels that are created by these paths contain more than one edge allowing forwarding of packets to longer distances without dynamically checking which neighbor is "free". In addition unlike the EDCA, for PCA, the FTMs are filled once for a specific graph topology regardless of the communication sessions that are currently going on.

Intuitively the PCA method proposed here significantly differs from current techniques to perform routing and channel allocation in ad hoc networks as follows:

1) It creates multi-channels containing more than one edge, actually transforming the communication graph to a hyper-graphs which is a graph whose edges connects more than two nodes. This stand in contrast to current techniques that create channels only between neighboring nodes. By the term channel we prefer to a group of nodes that agrees on a set of frequency/time slots through which data-packets can be broadcasted.
2) We focus on streams rather than on individual packets, we thus assume that a channel, once created, will be used to transfer a stream of packets. Moreover, we assume that pipelining a stream of n packets through a channel of length k in a pipeline mode is faster than $n \cdot k$ namely sending each packet through k distinct channels. Thus, by creating channels of length $k > 1$ we support pipelining of a stream of packets where packets are being sent along the edges of the path without acknowledgement.
3) Further, all current methods for routing in ad hoc networks use adaptive and dynamic creation of channels tracking the frequent changes in the topology of the communication graph. In contrast, PCA samples the communication graph to create channels and attempts to use them for a relatively long time regardless of the frequent changes in the topology. There are two reasons as to why a sample "road-map" will be useful for a relatively large time before an update of it is needed:
   - There are many changes caused by frequent movement that cancel one another, e.g., a node moves out of a channel but another may take its place listening and transmitting at the right frequencies and time slots.
   - The sampled "road-map" may be wrong about some of its roads (channels) but by attempting to use it a packet will travel at the direction it needs to go. Thus the sampled map is a good approximation of the true road-map that currently exits.

   Thus PCA creates an approximated "road map" which is likely to be useful in most cases even when the "roads" are frequently changing.

Some of the related works are as follows. Dual Busy Tone Multiple Access [7] is a method that divides a common channel into two sub-channels. Wu et al [15], propose MAC protocol called Dynamic The main drawback of this protocol concealed in the fact that RTS, CTS and ACK packets exchange are necessary for every pair nodes on every communication path. Jain et al [8], propose a protocol that achieves throughput improvements by intelligently selecting

the data channel, but also required RTS and CTS and ACK packets exchanged for every pair nodes as described in DCA. Finaly, [12] propose a protocol that uses all channels as data channels where nodes negotiate channels with their destination nodes during a time-window. In this window, every node must listen to the default channel. [14] proposed a Usage-Prediction Based Channel Allocation scheme for GSM networks.

2. The Problem of Finding a Collision-Free Maximal Path Coverage

The problem of finding a maximal FTM scheduling for a given communication graph with communication paths of length k is equivalent to finding maximal coverage of a graph by disjoint paths of length k.

Definition 2.1: Let FTG be an undirected graph resulting from expanding a communication graph G such that each node has been replaced by a clique of internal edges representing the FTM slots as depicted in figure 2. Each communication edge of G have been expanded to a set of external-edges connecting the suitable f,t nodes of two neighboring cliques. By definition FTGs are not general graphs and in particular each path in an FTG consists of alternating external and internal edges (see the FTG graph of figure 2 for an illustration). Let a $collision\ free\ k\ path\ cover$ (CFkPC) of an FTG be any cover of it's edges by a maximal number of directed paths $p_1, p_2, \ldots, p_n$ of length k ($k \geq 1$) such that:

- Each path p_i corresponds to some path of FTG.
- No node/edge is shared by two paths (node/edge disjoint paths).
- All the nodes of a path are distinct (i.e., paths cannot visit a node more than once).
- Each path should start with an external-edge and end with an external edge.
- Paths should not be connected by an external edge that connects two of their external edges. Thus if u, v are two external edges in two different selected paths then there can be no external edge in G that connects them (we later show that this is sufficient to prevent hidden terminal problems).

We can sharpen this definition to include paths of length smaller than k by grading a path-cover of G by a vector with k coordinates $grade = < x_k, \ldots, x_1 >$ where x_i counts the number of paths of length i in the cover and two grades are compared lexicographically. A maximum CFkPC of an FTG is a CFkPC with the maximal number of paths of length k or the one that achieves the maximal grade.

The CFkPC of graphs is similar to the multi-dimensional matching [1] and to $cover - by - disjoint - paths$ in graphs which are all NP-complete problems. Thus, CFkPC is likely to be NP-complete. Figure 2 illustrates how the communication graph with 3×2 FTMs at each node is extend to include the $M_{f,t}$ slots as nodes forming an undirected graph called FTG. The FTG is formed by:

1) For each node v, every slot in its FTM $M^v_{f,t}$ is made a node of FTG.
2) For each edge (v,u) and f,t slot add all edges $\left(M^v_{f,t}, M^u_{f,t}\right)$ (indicating a possible send-receive operation between v and u using this f,t slot).
3) Add an internal edge between every two slots of the same node $M^u_{f,t}$ and $M^u_{f',t'}$ where either $f \neq f'$ or $t \neq t'$ (indicating that a message received at $M^u_{f,t}$ will be sent at $M^u_{f',t'}$ or vice-versa).

A possible cover of the resulting FTG by paths of up to $k = 5$ edges is given at the upper part of figure 2. This cover, does not satisfy the hidden terminal requirement and several paths can collide, e.g., the receive in $M^u_{f0,t1}$ collide with the Send of both $M^w_{f0,t1}$ and $M^z_{f0,t1}$.

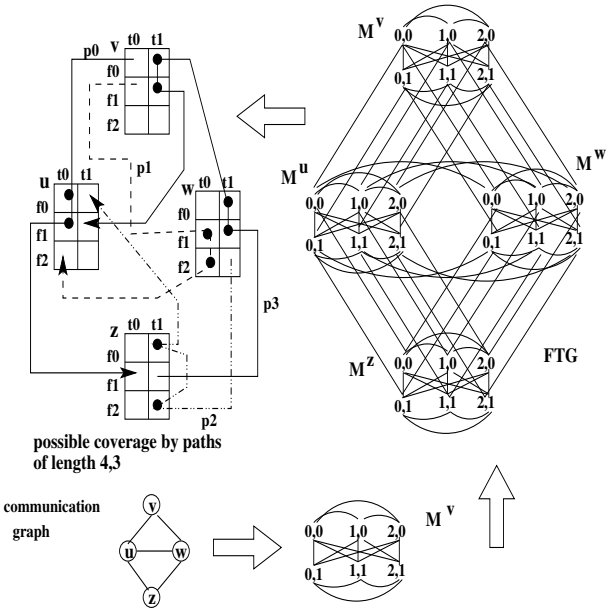

Fig. 2: Building the FTG of a communication graph and a possible $k - path - cover$ for it with 3×2 FTM

3. The algorithm for computing maximal CFkPC

We turn now to the algorithm for computing maximum CFkPC of an FTG. First we show that finding a CFkPC allowing collision free broadcasts:

Lemma 3.1: The conditions of definition 2.1 grantee that if we broadcast packets along the paths of a CFkPC then no two packets will collide.

Proof omitted due to space limitations.

Next consider a possible lower bound on the number of paths in the CFkPC of a given FTG:

Lemma 3.2: Let a conflict graph CG of a given FTG be a graph whose nodes correspond to the edges of the FTG and an edge is inserted between two nodes u, v if the corresponding edges $e_u, e_v \in FTG$ collide under the conditions of definition 2.1. It follows that the number of paths of length k in a maximum CFkPC of the given FTG is smaller than $\frac{|MIS(CG)|}{k}$ where $MIS(CG)$ is the maximum independent set of CG.
Proof omitted due to space limitations.

Finding an MIS of a graph [13] is well known and if we settle for maximal independent set [11] we can use fast distributed algorithms suitable for ad-hoc networks. Let G be a given graph representing a communication graph where each node uses an $F \times T$ FTM. The algorithm for maximizing use of multi-channels of length k uses the following steps:

1) Let each node internally maintains its part of the FTG by representing the FTM as a clique as described in figure 2. This requires that each node will identify its neighbors.
2) Compute conflict graph CG using the two rules of definition 2.1: 1) two connected edges of the same type and 2) three connected external edges. Note that this can be done distributively letting each node acquire information on conflicting edges from its neighbors.
3) Find maximum independent set (MIS) in the resulting CG. Statically this can be computed by an exact exponential algorithm [9], [13] or an approximated by a polynomial approximation algorithm for finding the biggest colored class as the MIS [10]. For a distributed algorithm that can be used in ad-hoc networks it is reasonable to use an algorithm for finding a maximal independent set (an IS that cannot be increased by adding any other node to it) to approximate the MIS. Distributed maximal IS protocols have been used in ad hoc networks extensively to find a connected dominate set in the communication graph [3], [4]. We select the algorithm of [11] that completes in $\log^*$ steps on unit-disc graphs (using GPS addresses as unique node ids). The size of the MIS is also computed and broadcast to every node.
4) Finally multi-channels can be computed using a simple protocol that grows paths randomly:
   a) Each node learns which edges of the FTG and the MIS resides in its neighbors.
   b) External edges of the MIS are selected at each node to be the head of a path. This is done with probability $\frac{k}{2|MIS|}$ so that most edges of the MIS are not selected and are marked as free-edges.
   c) For each path L whose head is currently at node v we randomly select an internal edge from the MIS that will continue it. Next we select the external edge from the MIS that will continue it to the next node u and send a continuing request for L to u.
   d) When a node u receives a continuing request for path L from v it checks to see if $< u, v >$ is a free external edge which is not used for any other path. If so u returns an acknowledgment to v and marks $< u, v >$ as used. Otherwise, u returns a fail message to v which ends the path L.
   e) This is continued for each active path until this path reaches a length of k or reaches a node for which there is no free edge in the MIS that can be added to it.
   f) A backward chain of messages from the last active end-point of each path L (either reached length k or terminated) is sent back through the active end-point of a path/channel to configure the FTMs at each node.
   g) There is a fixed time budget (order of k steps) after which each node assumes that this phase is over and packet routing is performed.

Figure 3 illustrates how the algorithm works. The input FTG is a grid of alternating levels of external and internal edges. The edges are marked with numbers $0-11$ and also by their type $'ex/in'$ for external/internal. Figure 3 right side contains the resulting conflict graph (CG) with nodes corresponding to the FTG edges $0-11$ and edges between any two conflicting edges. The MIS for the CG of figure 3 is marked by circled nodes. In order to create the paths (marked by dashed arrows) for this MIS, we start with edge-0 which can be continued only to edge-6 and finally end this path with edge-10. The second path starts with edge-3 and can be continued with two internal edges (edge-7) and (edge-08) out of which we cannot continue with edge-7 since edge-10 is not free so we attempt to continue it through edge-8. However edge-8 is an internal edge that cannot be continued by an external edge in the MIS so the resulting second path contains only edge-3.

For a given ad hoc network with n nodes, transmission radius r, F frequency channels, T time slots and field-size $L \times L$, we can use a maximal CFkPC of the communication graph as the base for an algorithm to send streams of packet over ad hoc networks. The algorithm is called PCA works as follows:

1) We globally set the value of k to be $2 \cdot \frac{field\_length}{transmission\_range}$ which is the maximal Manhattan distance between any two nodes assuming uniform distribution of the nodes in the given field (also assuming that there are enough nodes to form this density).
2) CFkPC is performed globally and the resulting paths are held in suitable routing table in every node. Thus every node has knowledge of the CFkPC paths that pass through it (including their length, start/end nodes and FTM slots that are used to realize them). Note that the CFkPC may include paths of length 1.

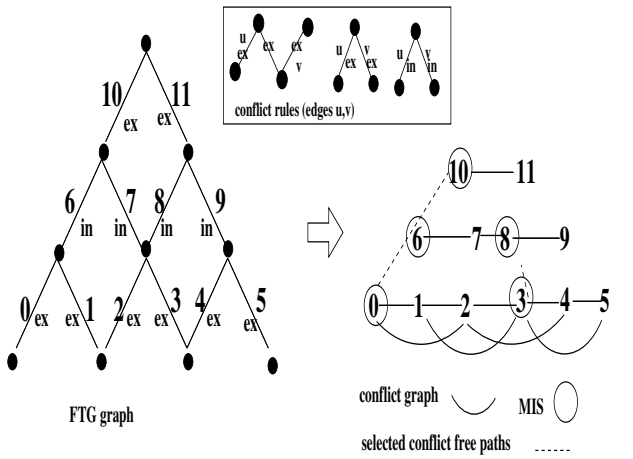

Fig. 3: Conflict graph and the rules to create it for a grid like FTG

3) Streams of packets are generated arbitrary in source nodes where each stream contains a relatively large number of packets which should be sent to the same destination.
4) At each cycle all nodes receive packets through their FTM slots and send them further as follows. The packet is sent to the next node in its current path if that node gets it closer to its destination. A packet will switch to another path if the next hop in its current path does not get it closer but there is another path in the routing table that will and that path is not used. Otherwise the packet proceeds in its current path if there is a next hop for that path. In case that the packet reach to the end of its path and there is no unused path through which it can continue it is moved to a waiting queue to wait until there is an unused path through which it can continue.
5) One slot is reserved for beacon messages where each node transmits its ID, its location, and a list of un-used F-T slots through which packets can be received "outside" the scope of the CFkPC paths. Each node constantly listen and collect beacon messages from its current set of neighbors and collects their un-used slots.
6) When a node u does not detect beacon message from a neighbor v for a certain period of time it assumes that v moved out of transmission range and the set of paths that passed through v are now "broken". Packets that needs to continue through a broken path are sent through an un-used slot to a suitable neighbor. If no un-used slot is available and no un-used path can be used packets from broken paths are added to the waiting queue.
7) A new neighbor v may move to the vicinity of a node u and is thus detected by u through its beacon message. In that case u and v will attempt to join their broken paths by connecting them together.
8) A packet may be dropped if its TTL expires, receiving queue is full, or if it collides at some node with another packet that have been transmitted at the same FT-slot.
9) The CFkPC is recalculated periodically every fix period of time called the re-calculation time.

4. Experimental Results

In here we discuss the experimental evaluation of the proposed PCA algorithm in comparison to the EDCA technique. We first summarize our assumptions: Time is divided to T frames such that for every time frame, F frequencies are available for use and none of the frequencies overlap. transmitted on different frequencies do not interfere with each other. Each host can listen or transmit on more than one channel at a time. Fixed-channel-bandwidth model is used. Each host is equipped with a single full-duplex transceiver [5]. Nodes are synchronized using GPS [6]. Simulations are performed in multi-hop networks scenario of varied nodes that are randomly placed in a 500m $\times$ 500m area. Source and destination nodes for streams are randomly chosen with probability 0.5. A node may be the source for multiple destinations and a node may be the destination for multiple sources. Each simulation was performed for duration of one minute. Packets have been partitioned to streams of $50 \ldots 250$ packets. Each data point in the result graphs is an average of 15 runs. Packet size is 512 bytes. Packet TTL was 10 implying that packets are dropped after 10 hops. Queues for receiving packets were limited to 50 packets while queues for transmission have unlimited capacity. We used geographic routing algorithm where nodes know their geographical coordinates and also their one hop neighbors coordinates. The parameters we vary are: speed of nodes, number of frequency slots F, number of time slots T, size of flows (in packets), transmission range and number of nodes in the network. We use the throughput performance metrics in our simulation:

$$Throughput = \frac{Packet\_Length * Num\_Of\_Received\_Packets}{Total\_Time}$$

where the throughput measures the rate in which packets have been received at their final destination.

It immediately follows that there are too many parameter values that should be considered, since each measurement of the PCA requires setting a value to: n number of nodes, r the transmission range, $L \times L$ size of the field, F number of available frequency slots, and T number of time slots. However, assuming uniform distribution of the nodes at any given time in the field, T can be approximated by $\frac{T \cdot F}{2} = \frac{\beta \cdot \alpha}{4 \cdot n}$ where $\alpha = \frac{\pi \cdot r^2}{L^2} \cdot n$ and $\beta = \frac{\gamma \cdot n}{2} \cdot \delta$. This formula is based on the following assumptions:

- α is the average number of nodes that are in communication range of any node (v).

- β is the approximate number of transmissions in the field such that:
  - γ is the probability that a node will start a stream of packets (0.5 in our experiments).
  - $\frac{n}{2}$ is due to the fact that each stream has two end points.
  - δ is the average number of hops a stream will follow and is approximately equal to $L/r - 1$ assuming $n > (L/r)^2$ and that streams travels in the shortest Manhattan distance between their end-points that have been selected at random.
- The term $\frac{\beta \cdot \alpha}{n}$ is the expected number of transmissions from the nodes that are in the communication range of v. Out of this number we assume that transmissions are sent equally to all four directions out of which only $1/4$ will be sent to v.
- Thus at maximum throughput each node v should have enough $F \cdot T$ slots to pass streams from/to all the nodes that reside in its transmission range yielding eq. 4.

The first experiment we consider is designed to verify the formula of eq. 4. We run the PCA algorithm with $F = 4$, $r = 100m$, $L = 500m$, $n = 60, 90, 120, 150$ and measured the throughput of PCA for $T = 2, 3, 4, 5, 6$ (with random way-point and 50 packets per stream). For this setting of eq. 4 we get that $\beta = \frac{0.5 \cdot n}{2} \cdot 8 = 2 \cdot n$, and $\alpha = 0.12 \cdot n$ yielding that $T \cdot F = 0.12 \cdot n$ and $T \cdot = 0.03 \cdot n$ The results in figure 4 show that for $n = 150$ the throughput ceased to improve for $T > 5$ which is indeed what eq. 4 determines ($0.0314 \cdot 150 = 4.71$). For $n = 120$ throughput in figure 4 ceased to improve for $T > 4$ which agrees with $0.0314 \cdot 120 = 3.76$. Similarly for $n = 90$ the experimental result is $T > 3$ which agrees with $0.0314 \cdot 90 = 2.82$ of eq. 4, and so is the result for $n = 60$ we have $T > 2$ which agrees with $0.0314 \cdot 60 = 1.88$.

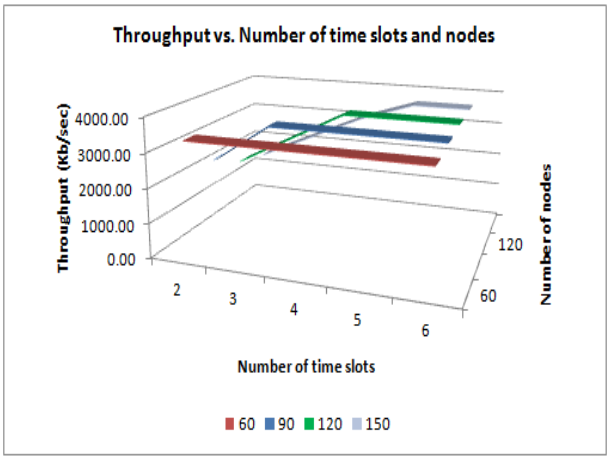

Fig. 4: PCA Throughput vs. Number of time slots T and number of nodes n.

Next we compare between PCA and EDCA selecting F, T values which are bellow the saturation point of eq. 4 $\frac{T \cdot F}{2} \leq \frac{\beta \cdot \alpha}{4 \cdot n}$. In this set of experiments we compared the throughput of PCA and EDCA for $n = 120$, $r = 250m$, $L = 500m, \#stream = 50packets$ and tested for several $< F, T >$ values. Figure 5 presents the results for $F = 4, 8, 12, 16$ and $T = 24, 12, 12, 6$. The results show that when at $F = 12$ EDCA cease to improve while the PCA continues to improve the throughput (about 40%) and maintain the channel utilization. This was also obtained for $\#stream = 250packets$ (Omitted due to space limitations).

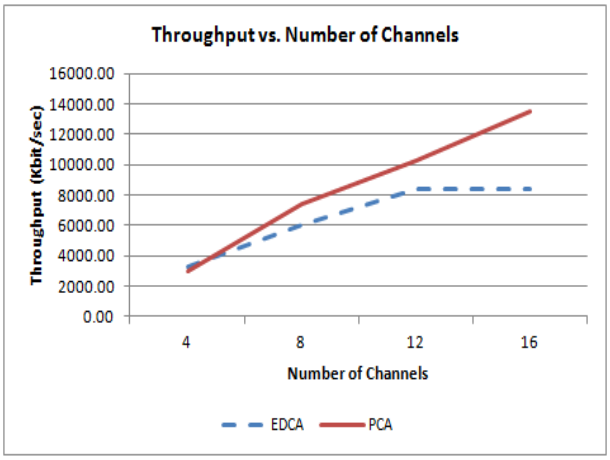

Fig. 5: Throughput vs. increased F, T values of EDCA and PCA with 50 packets per stream

The explanation for this behavior is that when the network load become very high, the control channel becomes a bottleneck for the EDCA but not for PCA. The throughput ratio between EDCA and PCA is significant. When using only 4 channels, the ratio is about 8% for EDCA and increased to 37% in favor of the PCA when using 20 channels. In a different experiment (omitted due to space limtations) we set the number of packets of each flow to 200 and found even more throughput improvement using PCA algorithm. Results of this test show that when using 4 channels, the ratio is about 18% and increased to 40% when using 20 channels. The difference between two tests is because in PCA, long communication paths are already exist and ready for streams transferring without any overhead of RTS/CTS protocols. Clearly without increasing the amount of streams β even PCA will cease to improve (as follows from eq. 4).

We have also tested the effect of the mobility rate on network throughput using the Random Waypoint mobility model. The results show that the more the speed of nodes increases the more the network throughput decreases and that there is a constant gap of 17% more throughput of PCA versus EDCA. Another experiment we done is to compare packets latency between the two algorithms. The latency was

computed for 100 packets that were selected randomly. We varied the number of generated streams in the network from 50 streams up to 500. The results show that for PCA the average latency of a packet is 6 simulation cycles compare to 17 for the EDCA. The last experiment measures the effect of the CFkPC recalculation on the network throughput. A degradation in the throughput is expected when the recalculation time increases however this experiment shows that for certain range of values this degradation does not harm the usefulness of the PCA. Due to space limitations the results of these three sets of experiments can not be included.

References

[1] *3-dimensional matching*. http://en.wikipedia.org/wiki/3-dimensional_matching, 2011.

[2] *Hidden node problem*. http://en.wikipedia.org/wiki/Hidden_node_problem, 2011.

[3] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 134–144. ACM, 1991.

[4] K.M. Alzoubi, PJ Wan, and O. Frieder. Maximal independent set, weakly-connected dominating set, and induced spanners in wireless ad hoc networks. *International Journal of Foundations of Computer Science*, 14(2):287–303, 2003.

[5] D. W. Bliss, P. A. Parker, and A. R. Margetts. Simultaneous transmission and reception for improvedwireless network performance. In *Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th Workshop on*, pages 478 –482, 2007.

[6] I.A. Getting. The global positioning system, 1993.

[7] Zygmunt J. Haas, Senior Member, Jing Deng, and Student Member. Dual busy tone multiple access (dbtma) - a multiple access control scheme for ad hoc networks. In *IEEE Transactions on Communications*, pages 975–985, 2002.

[8] Nitin Jain, Samir R. Das, and Asis Nasipuri. A multichannel csma mac protocol with receiver-based channel selection for multihop wireless networks. In *In IEEE IC3N*, pages 432–439, 2001.

[9] T. Jian. An $o(2^{0.304 \cdot n})$ algorithm for solving maximum independent set problem. *Computers, IEEE Transactions on*, 100(9):847–851, 1986.

[10] T. F. More. *Estimation of sparse hessian matrices and graph coloring problems*. Springer, 1982.

[11] J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, pages 35–44. ACM, 2008.

[12] Jungmin So and Nitin H. Vaidya. Multi-channel mac for ad hoc networks handling multi-channel hidden terminals using a single transceiver. *ACM MobiHoc*, 2004.

[13] R.E. Tarjan and A.E. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6(3):537–546, 1977.

[14] Jinsu Wang, Sharad Mehrotra, and Nalini Venkatasubramanian. Pbca - prediction based channel allocation. In *GLOBECOM*, pages 4801–4806. IEEE, 2007.

[15] Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu. A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks, 2000.