M. Elhawary and Z.J. Haas, "Busy Tone Multi Channel (BTMC): a New Multi Channel MAC protocol for Ad Hoc Networks," 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'08), poster session, Hong Kong, March 17 - 21, 2008

Busy Tone Multi-channel (BTMC): a New Multi-channel MAC protocol for Ad Hoc Networks

Mohamed Elhawary
Computer Science Department
Cornell University
Ithaca, NY 14850, USA
hawary@cs.cornell.edu

Zygmunt J. Haas
Electrical and Computer Engineering Department
Cornell University
Ithaca, NY 14850, USA
haas@ece.cornell.edu, wnl.ece.cornell.edu

***Abstract*— On one hand, operation of wireless network protocols, such as 802.11, relies on multiple channels, but without exploiting the multiple channels to increase the network throughput. On the other hand, the operation of previously published multi-channel MAC protocols either is based on more than one radio per node or requires node synchronization. In this paper, we propose a new MAC protocol that avoids these shortcomings. Our protocol uses hash functions and busy tones. The former effectively distribute the control overhead over all the channels, while the latter addresses the hidden terminal-problem and the channel *rendezvous*-problem in multi-channel networks. We compared our protocol with the DCA multi-channel MAC protocol and with 802.11. Although the DCA protocol uses multiple radios per node, the performance of our protocol is comparable with DCA and outperforms the 802.11 protocol. As a point of reference, our protocol performs three times better than the IEEE 802.11 protocol with three channels and results in 25% increase in throughput for a 50-node network.**

Keywords- MAC protocols, multi-channel MAC, 802.11, MANET

I. INTRODUCTION

Wireless mobile ad hoc networks (MANET) are formed by a collection of nodes that communicate without using a central control or a fixed infrastructure. All operations performed in this type of networks must be distributed and have a low overhead due to the limited wireless capacity. Indeed, many believe that proliferation of the MANET technology will depend on its ability to deliver improved performance, relative to conventional wireless LANs.

One way to increase the capacity of wireless ad hoc networks is through the simultaneous use of multiple channels [11]. This is a practical alternative, as operation of many wireless protocols rely on the ability to transmit on multiple frequencies (e.g., the IEEE 802.11 [2] standard)[1]. When channels are orthogonal, neighbor nodes that use different channels are not interfering with each others when transmitting simultaneously.

A protocol designed for the multi-channel environment needs to solve the *hidden-terminal* problem associated with the multi-channel environment and specify how channel *rendezvous* is done. The *hidden terminal* problem in the multi-channel environment ([16], [11]), occurs due to the fact that nodes can listen to only one of the available channels at any particular time. Thus, a node using one channel is unaware of the communication on the other channels. For example, node ***A*** which resides on channel f_1, tries to establish communication with node ***B*** over channel f_2, without being aware that node ***B*** has been already communicating with a third node, ***C***, on channel f_2.

* This work was sponsored in part by NSF grant number CNS-0626751.

[1] The IEEE 802.11a standard [3] defines 12 channels and IEEE 802.11b [4] defines 14 channels. However, out of these 14 channels only 3 are orthogonal.

Another problem in the multi-channel environment is the *channel-rendezvous* problem between communicating nodes. To communicate, the sender and the transmitter nodes need to be assigned the same channels. The MAC protocol should specify how this assignment is made, when the protocol is implemented without a central controller.

The previously published multi-channel MAC protocols are subject to a number of limitations. Some protocols use only one channel for control (i.e., to assign the channels to nodes) and as the single channel becomes saturated [16], the capacity of the protocol is limited. Other protocols assume multiple radios in the nodes [8], increasing the implementation cost of the network. Yet, other protocols [11], [5] assume costly and hard-to-implement clock synchronization across the network nodes. Consequently, in our work, we assume that the MAC protocol possesses the following attributes:

- based on a single radio per node,
- solves the *hidden-terminal* problem in the multi-channel communication environment,
- requires no synchronization,
- does not dedicate channels for data or for control exchange.[2]

In this paper, we propose a new distributed multi-channel MAC protocol that benefits from multiple channels available in ad hoc networks. In our protocol, sender nodes use an ordered list of hash functions to calculate the channel numbers to be used for their communication. Each channel is associated with a single busy tone.[3] Receiver nodes set up busy tone signals on the channels which they currently use to declare that the channels are unavailable. A sender node failing to *rendezvous* with the receiver on a particular channel uses the next hash function on the list to calculate the next channel to try. The protocol solves the hidden-problem and the *rendezvous*-problem in the multi-channel environment.

To the best of our knowledge, this is the first MAC protocol with a single radio per node that does not require synchronization or a dedicated control channel and that does not cause network partitioning. The protocol increases the capacity of wireless networks for the single- and the multi-hop communication scenarios.

The extra hardware required by our protocol is a busy-tone transmitter/receiver in the network node. Of course, a single-tone transceiver is significantly less expensive than a complete radio transceiver [16]. The bandwidth consumption of a busy tone is negligible compared to the bandwidth of a data channels [14].

II. THE BUSY TONE MULTI-CHANNEL PROTOCOL (BTMC)

The proposed BTMC protocol uses an ordered list of k hash functions ($h_0, h_1, \ldots, h_{k-1}$) and when given the MAC address of

[2] There is no single control channel that can get saturated.

[3] A "busy tone" is a single frequency. Detection of a busy tone is done by sensing the energy at the particular frequency, for example through a simple notch filter with a threshold.

a node, the protocol returns a channel number from the range [*0, m-1*], where *m* is the total number of channels available. We assume that nodes know the MAC addresses of their neighbors,[4] and all the nodes have the same set of the hash functions.

The main idea behind our protocol is that a sender node applies in sequence the hash functions (one after another and starting from h_0) to the MAC address of the receiver (neighbor) node, with which it wants to communicate, until one of the hash functions returns a number of a channel which is available for transmission. Such a channel is referred to as a *current default channel* and such an operation as *hash function search.* A channel is said to be *available* or *free* if its busy tone is down. A node which is not communicating, and thus available for reception, performs the same *hash function search* on its own MAC address, and tunes to the first available channel, h_i. The node then passively listens on the h_i channel for any attempt to establish a communication by its neighbor nodes. Since both, the sender and the receiver nodes perform the same operation, except for special situations, the two nodes will *rendezvous* on the same channel. The sender then starts the RTS/CTS dialogue with the receiver, during which the receiver sets up the busy tone of the h_i channel, preventing other nodes from using the channel. The receiver clears the busy tone after a successful transmission or on *abort*.

When an idle node waiting on a free channel h_i receives a control packet not intended to it, it extracts from the RTS/CTS exchange the duration of time that the channel will be occupied for the other communication. The node then stores the time, t_i, when h_i will become available again, and performs the *hash function search*, but starting from the next hash function on the list. At time t_i, the node switches back to channel h_i. Thus, the node will end up on an available channel h_j, with j being the smallest index.

For purpose of analysis, we define the following variable:

δ: data packet transmission time [sec]
γ: transmission time of the RTS or CTS control packets [sec]
τ: max one-way sender-receiver propagation delay [sec]
t_d: busy tone detection delay [sec]
N: the number of nodes
L_c: length of control packets
R and Rd: the total system bandwidth and the bandwidth of a data channel, respectively.

By way of an illustrative example, we explain the operation of the protocol. Assume sender node ***A*** attempts to communicate with receiver ***B***. It sends RTS on the first (based on the hash functions list) free channel assigned to ***B***. After sending RTS, node ***A*** waits for CTS on the same channel for duration of $2\tau + \gamma$. When node ***A*** senses the busy tone of the used channel during the RTS transmission or when ***A*** does not receive CTS within the above waiting time, ***A*** *abort*s the transmission, starting a new *hash function search.*

The receiver node ***B***, if idle, replies with CTS and sets up the busy tone on its current channel. Next, it sets a timer for time $2\tau + \delta + t_d$ and waits to start receiving data from node ***A***.

After receiving the CTS message from the receiver node ***B,*** node ***A*** senses the busy tone of the channel. If node ***A*** does not sense the busy tone, ***A*** *abort*s the transmission, starting a new *hash function search.* Otherwise, ***A*** transmits the data packet.

After node ***B*** receives the data packet or if ***B*** does not receive the data before timeout, node ***B*** clears the busy tone of the channel and becomes idle.

If node ***B*** senses the busy tone of the channel before it sends the CTS, node ***B*** switches to the channel determined by the *hash function search*, which starts from the next hash function on the list. Node ***B*** waits on the new channel for $2\tau + 2\delta$, which is the time needed for the sender ***A*** to discover that the receiver cannot use this channel and to send a new RTS on the next channel.

Note that the above waiting times do not take into account hardware characteristics. In practice, hardware delays need to be added.

Nodes become idle when they are initialized or powered up and when they finish transmitting or receiving a packet. Also, a node becomes idle when it times out while waiting for data. The "deafness problem" arises when a node is transmitting and hence cannot listen to RTS sent by other nodes. As a transmitter fails to reach a receiver on receiver's default channels, as determined by the *hash function search, the* transmitter determines that the receiver node is busy and enters the *backoff state.* It stays in the backoff state for a duration determined by the simple binary exponential backoff algorithm, at the end of which, it repeats the communication attempt based on the *hash function search.* An alternative would be for the receiver node after it finishes receiving a data packet, to inform, through a "null RTS"[5] packet, all its neighbor nodes on the current default channel that it is free again.

III. Performance Analysis

To evaluate the performance of the protocol, we make the following assumptions used in our analytical model:

- Packet collisions are the only source of packet errors.
- Data processing time, channel switching time, and the transmit/receive turnaround time are negligible; we relax this assumption at the end of this section.
- The nodes in the network collectively generate (including retransmissions) Poisson control packet arrival with mean aggregate rate of G [requests/sec].
- Bandwidth of a busy tone is negligible compared to the bandwidth of a data channel [14].
- The network is fully connected.
- The arrival rate of packets to all the channels is equal.
- Exponentially distributed data packets with mean L_d [bits]

The successful control packets exchange for reserving a channel results in transmission of a data packet on one of the channels. We use the queuing model ***M/M/m/m***, *m* being the number of data channels, where a successful RTS/CTS exchange is modeled as a Poisson arrival of a data packet in the queuing model. The transmission of a data packet on one of the channels is modeled as the service done by on one of the servers. Note that there are no arrivals when all the servers are busy as control packets are only sent on a channel when the channel is free.

Nodes compete on the channels and senders send RTS to idle receivers only. Successful RTS/CST handshake leads to a data packet transmission on one of the channels.

A couple of RTS and CTS packets will be transmitted successfully if no other packets are transmitted in the first $2\tau + \gamma + t_d$ seconds. Then, all nodes within the transmission range of

[4] A *neighbor* of node ***A*** is another node who can directly communicate with node ***A***.

[5] The "null RTS" is a control packet stating "I'm available."

the receiver sense the busy tone. The probability of success of RTS followed by CTS is:

$$P_s = \exp\left(-\left(\lambda/m\right)\left(\gamma + 2\tau + t_d\right)\right). \quad (1)$$

A successful handshake time consists of RTS transmission, propagation delay, setting up the busy tone signal, CTS transmission time, propagation delay, and busy tone detection delay:

$$T_s = 2\gamma + 2\tau + t_d. \quad (2)$$

As no new RTS packets will be sent out in $2\tau + \gamma + t_d$ seconds after the start of node's RTS packet, then, the longest failed busy period is $2\gamma + 2\tau + t_d$. The shortest busy period is when more than one RTS is sent at the same time, this leads to a busy period equals to $\gamma + \tau$. Assuming the colliding packets arriving uniformly in $[0, \gamma + \tau + t_d + \tau]$, then the average time of the longest busy period and shortest busy period is $(2\gamma + 2\tau + t_d + \gamma + \tau)/2$. Hence the average duration of a failing busy period is:

$$T_f = \frac{3}{2}\gamma + \frac{3}{2}\tau + \frac{t_d}{2} \quad (3)$$

The duration of the contention interval of the control packets is:

$$W = \left(\frac{1}{P_s} - 1\right)T_f + T_s + \frac{m}{G} \quad (4)$$

Next, we define the R and R_d relation as $R = mR_d$. Nodes do not track the status of each other (due to lack of a single control channel), so a sender might contend for a receiver who is unavailable. Also, there is a chance that the current channel is busy. Consequently, two factors need to be added to the packet arrival rate of the queuing model λ: $\frac{N-2k-1}{N-1}$ and $\frac{m-k}{m}$, which account for the two above-stated issues., respectively, where there are k pairs of nodes currently communicating.

Every contention period W there is a new data packet arrival. This defines the data packet arrival rate per channel (λ/m) in data packets/sec in the queuing model as:

$$\lambda/m = \frac{1}{W}\frac{N-2k-1}{N-1}\frac{m-k}{m} \quad (5)$$

The average service rate, μ, of the queuing model that models the mean data packet transmission time is defined as:

$$\frac{1}{\mu} = \delta + \tau = \frac{L_d}{R_d} + \tau \quad (6)$$

The steady state probabilities in the ***M/M/m/m*** model, P_k are:

$$\frac{1}{W}\frac{N-2k-1}{N-1}(m-k)P_k = (k+1)\mu P_{k+1}, \quad k < m$$

Simplifying and using the fact that $\sum_{k=0}^{m} P_k = 1$, we get:

$$P_0 = \frac{\left[(N-2k+1)(N-2k+3)\cdots(N-1)\right]^{-1}}{\sum_{k=0}^{m}\left[\frac{1}{W(N-1)\mu}\right]^k\binom{m}{k}}$$

Thus the throughput S can be calculated as:

$$S = R_d\left(\sum_{n=1}^{m} nP_n\right) \quad (7)$$

The average packet waiting time *AWT* is defined as waiting time for a data channel to be free, plus waiting time for a successful handshake, plus the data transmission time:

$$AWT = P_m\frac{1}{m\mu} + W + \frac{L_d}{R_d} + \tau \quad (8)$$

The comparison of the simulation results with analytical results confirms the validity of our analytical model.

IV. Experimental Results

We used simulation to evaluate the performance of our protocol by comparing it with the commonly used single channel IEEE 802.11 protocol [2] and with the Dynamic Channel Assignment protocol (DCA) [16]. DCA, described in details in section V, is a multi-channel MAC protocol that uses two packet radios per node and does not require synchronization. DCA uses one channel for control packets and the remaining channels for data packets.

We have used the JiST [1] simulation package to compare the three protocols in the single-hop and the multiple-hop scenarios. We assume channel bandwidth of 1 Mbps, the length of the data packets of 4 Kbytes, and m=3, unless otherwise stated. The queue size at each node can hold up to 50 packets and the generated traffic is based on UDP flows. The number of hash functions is equal to the number of channels. For simplicity, we used the simple mod operator as our first hash function; $h_0(\textit{MAC-address}) = (\textit{MAC-address}) \bmod m$. Also, $h_1 \ldots h_{k-1}$ were replaced by linear probing where h_i(MAC-address) = h_0(MAC-address) + $i \bmod m$. Each of our simulation results represents an average of 10 random runs, and each simulation run represents a real time of 100 seconds.

The parameters we vary are: the *packet arrival rate*, the *number of nodes*, and the *number of channels*. The metrics that we use for evaluation are: the *aggregate throughput* (sometimes called *system throughput*), the *average packet delay*, the *number of routes discovered*, and the *average number of hops per each route*, for evaluation of the performance of a routing protocol relative to our proposed protocol. The *system throughput* measures the total throughput of all the nodes in the network. Lost packets are ignored in calculation of the *average packet delay*, which measures the delay from receiving the packet in the MAC layer of the sender node until receiving the packet at the receiver node.

Next, we present the results of the single hop experiments. Nodes are picked randomly as source or destination (can be a source and destination at the same time) and each node can be a source or destination for more than one flow.

The first set of figures,[6] Figure 1 – Figure 3 shows the system throughput versus the packet arrival rate which serves as the load to the network. Each figure represents a different number of nodes. When the network load is low, the overhead of channel *rendezvous* is relatively high and one channel is enough to handle this workload in IEEE 802.11. But, when the load is high, or near saturation, BTMC does much better than IEEE 802.11 as the *rendezvous* overhead becomes insignificant. BTMC has comparable throughput when the

[6] Curves labeled with "BTMC" refer to our proposed protocol, "DCA" refers to the DCA protocol, and curves labeled with 802.11 refer to the single channel IEEE 802.11 protocol.

number of nodes is small, but it has better throughput than DCA when the number of nodes gets larger and the load is high. The reason is that DCA uses one channel for control packets and only two channels for data packets. BTMC uses all the three channels for data packets. At high load, the collision between control packets increases which causes control channel saturation in DCA. Note that BTMC achieves better results while using less expensive hardware (one radio instead of two in DCA). The results also show that our protocol increases the capacity of the network as it handles better the high load.

In Figure 4, we vary the number of channels in BMTC from 3 to 6 and measure the system throughput in a single hop experiment with 30 nodes; all nodes are within transmission range of each other.[7] The results show that BTMC scales well with increasing the number of channels: the system throughput increases with the number of channels. This is a result of using all the channels as data channels. DCA does not have this channel scalability due to control channel saturation at high load. Also, in high load BTMC performs much better than IEEE 802.11 regardless of the number of channels available.

In the multi-hop experiments, we have a 400[m] by 400[m] area within which 50 nodes are randomly placed. The transmission range per node is 100[m]. Nodes are picked randomly as source and destination (can be a source and destination at the same time), and each node can be a source or destination for more than one flow.

In Figure 5, we measure the system throughput of a multi-hop network while varying the network load. The overhead of channel *rendezvous* is high when the network load is low and that is why IEEE 802.11 and DCA perform slightly better in this case. Some nodes are destination of more than one sender; these senders use the same channel for data channel and hence do not benefit from having more than one channel. But, when the load is high or near saturation, BTMC performs slightly better than DCA as the three channels are utilized in BTMC and the network benefits from using the extra channel as compared to DCA. In addition, DCA suffers from control channel saturation at high load. Also, as the channel *rendezvous* overhead becomes insignificant in BTMC, the scheme performs significantly better than IEEE 802.11. The results also show that our protocol increases the capacity of the network as it better handles high load.

Next we evaluate the average packet delay vs. the system throughput in multi hop networks with different network loads. The results in Figure 6 show that BMTC performs significantly better in both metrics relative to IEEE 802.11. When the load is high, the system throughput in BTMC becomes more than three times larger than that of IEEE 802.11 due to the ability of BTMC to distribute the flows across different channels and efficiently sharing the bandwidth among different flows.

Next we present the results of running the routing protocol AODV [9] over BMTC and IEEE 802.11. We used a routing protocol because all routing protocols rely heavily on broadcast messages. A broadcast message might not reach all the neighbors in case they are on different channels. We wanted to test the effect of the retransmission of broadcasts on different channels. Figure 7 shows that the average number of hops per route increases at high loads when we used BTMC. This is due to the fact that a node needs to resend the broadcast on three channels before returning to its first channel. There is a possibility that a neighbor tries to reply back before the node gets back to its own channel. Such a reply is missed in high load operation which causes to miss some shorter routes. On the other hand, the results in Figure 8 which shows the number of routes discovered by the routing protocol, tell us that in light load both protocols perform about the same. But, in high load, AODV with BTMC finds more routes due to BTMC's larger system throughput.

To sums up, our results show that the BTMC protocol has lower overhead and always outperforms IEEE 802.11 in highly loaded networks with multiple flows scenarios. BTMC either produces comparable results to DCA or does slightly better than DCA at high load. The main reason of this improved performance is the ability to explore the large number of channels available while increasing the capacity without saturating any one channel with control packets. The *rendezvous* mechanism in BTMC, based on hash functions, does not require a node to store much information about the nodes in the neighborhood, since the hash function list is common to all the nodes. Also, the *rendezvous* is scalable when we increase the number of channels. Busy tone signals, essential part of BTMC, are used in solving the hidden terminal problem in the multi-channel environment.

V. RELATED WORK

There are various problems with previous multi channel MAC protocols. Some protocols use only one channel for control messages and that channel gets saturated like RBSC [6], AMCP [10], and DCA [16]. Other protocols use more than one packet radio per node which is a costly hardware to support like multi-channel CSMA [8], DCA [16]. Other protocols like MMAC [11], TMMAC [17], SSCH [5], McMAC [12], HRMA [13] and CHMA [15] are impractical as they assume clock synchronization which is hard to be implemented in practice and increases the overhead of the protocol and hence decreases its performance. Some protocols need nodes to be idle for maximum packet transmission time after each transmission like AMCP [10] and others lead to network partitioning and starvation like LCM MAC [7].

The xRDT protocol in [7] uses busy tones and each node has one default channel to use and if busy the communication can not happen. This protocol fails to use free channels and depend heavily on the channel assignment which is not part of the protocol. Our protocol solves this problem by allowing rendezvous to happen on an ordered list of channels and benefit from all the available channels.

VI. SUMMARY AND CONCLUDING REMARKS

We proposed a new multi-channel MAC protocol BTMC for multiple-channel communication in ad hoc networks. Our results show that our protocol has a low overhead and always outperforms IEEE 802.11 in highly loaded networks with multiple flows scenarios. BTMC either produces comparable results to DCA, or does slightly better than DCA in high load, while using only one radio per node. BTMC uses hashing functions to assign channels and busy tones to reserve channels. The proposed protocol is practical to implement in the sense that it does not need synchronization and that it uses a single radio per node. In this sense, BTMC is unique among the other proposed protocols. Finally, the BTMC protocol does not require a separate control channel; thus (dynamic) optimization of the control channel is not necessary.

For future work, we plan to study the effect of using more sophisticated hash functions on the performance of the protocol. Also, we intend to include power-saving mechanisms, similar to IEEE 802.1.

[7] In Figure 4, BTMC-X means BTMC with X channels available.

REFERENCES

[1] JiST, http://jist.ece.cornell.edu/

[2] IEEE 802.11 Working group, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1997

[3] IEEE 802.11a Working group, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications – Supplement: High-speed Physical Layer in the 5 GHz Band," 1999

[4] IEEE 802.11b Working group, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) - specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band," 1999

[5] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," ACM Mobicom 2004

[6] N. Jain, and S. Das, "A Multichannel CSMAMAC Protocol with receiver-based channel selection for Multihop Wireless Networks," IEEE International Conference on Computer Communications and Networks (IC3N) 2001

[7] R. Maheshwari, H. Gupta, and S. Das, "Multichannel MAC Protocols for Wireless Networks," IEEE Conference on Sensor, Mesh and Ad Hoc Comm. and Networks (SECON) 2006

[8] A. Nasipuri, J. Zhang, and S. Das, "A multichannel CSMA MAC protocol for multihop wireless networks," IEEE Wireless Communications and Networking Conference (WCNC) 1999.

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

[10] J. Shi, T. Salonidis, and E. Knightly, "Starvation Mitigation through Multi-Channel Coordination in CSMA Multi-hop Wireless Networks," ACM MobiHoc 2006

[11] J. So, and N. Vaidya, "Multi-Channel MAC Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver," ACM MobiHoc 2004

[12] H. So, and J. Walrand, "McMAC: A Multi-Channel MAC Proposal for Ad-Hoc Wireless Networks," Technical Report, April 2005

[13] Z. Tang, and J. Garcia-Luna-Aceves, "Hop-Reservation Multiple Access (HRMA) for Ad-Hoc Networks," IEEE INFOCOM 1999

[14] F. Tobagi, and L. Kleinrock, "Packet switching in Radio Channel: Part II-The Hidden Terminal Problem in Carrier Sense Multiple Access and the busy tone solution," IEEE transactions on communications, 23(12):1417-1433, December 1975

[15] A. Tzamaloukas and J. Garcia-Luna-Aceves, "Channel-Hopping Multiple Access," IEEE International Conference on Communications (ICC) 2000

[16] S. Wu, C. Lin, Y. Tseng, and J. Sheu, "A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Mobile Ad Hoc Networks," Int'l Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN) 2000

[17] J. Zhang, G. Zhou, C. Huang, S. Son, and J. Stankovic, "TMMAC: An Energy Efficient Multi-Channel MAC Protocol for Ad Hoc Networks," IEEE International Conference on Communications (ICC) 2007

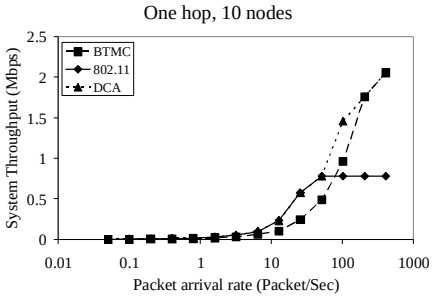

Figure 1 Throughput vs. load, 10 nodes

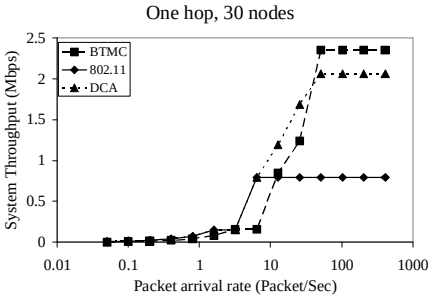

Figure 4 Throughput vs. load, 30 nodes

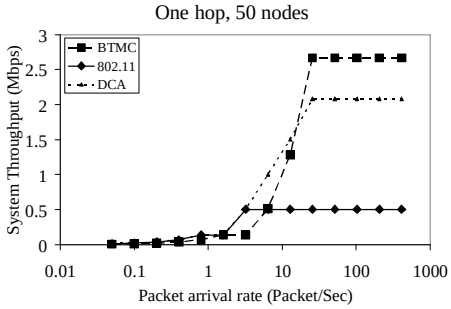

Figure 8 Throughput vs. load, 50 nodes

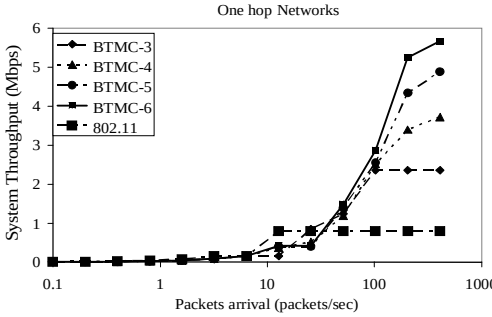

Figure 2 Effect of # channels , 30 nodes

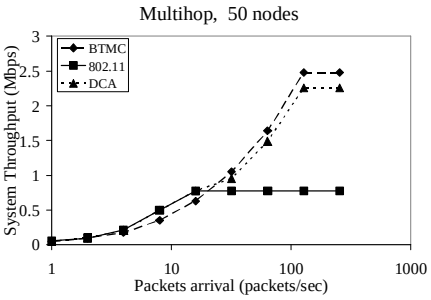

Figure 6 Multi-hop throughput vs. load

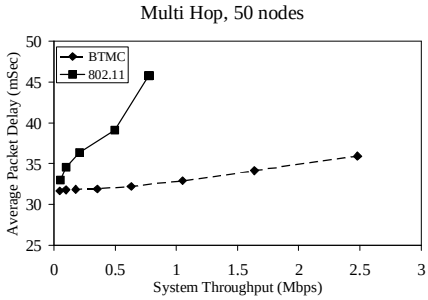

Figure 5 Multi-hop delay vs. throughput

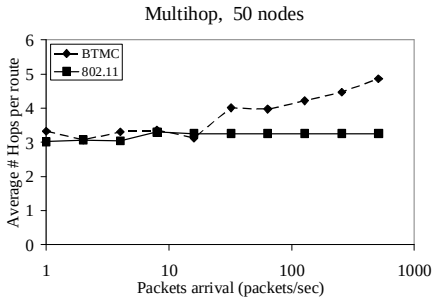

Figure 3 Average # hops per route vs. load

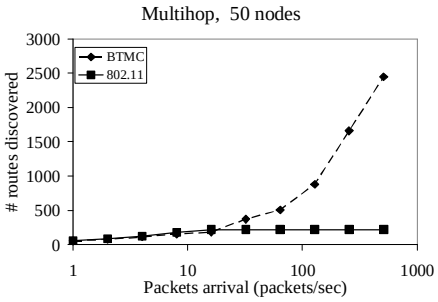

Figure 7 # Routes discovered vs. load