WiBi MAC: Biased Medium Access Control for WiFi

Cameron A. Keith
Computer Science and Engineering Department
Southern Methodist University
Dallas, Texas USA
ckeith@smu.edu

Anna A. Carroll
Computer Science and Engineering Department
Southern Methodist University
Dallas, Texas USA
aacarroll@smu.edu

Dylan C. Fansler
Computer Science and Engineering Department
Southern Methodist University
Dallas, Texas USA
dfansler@smu.edu

Ethan Busbee
Computer Science and Engineering Department
Southern Methodist University
Dallas, Texas USA
ebusbee@smu.edu

ABSTRACT

In this paper, we present WiBi MAC, a biased Medium Access Control (MAC) protocol for WiFi. The standard medium access approach utilized by WiFi relies upon Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) combined with slotted Aloha. This approach achieves a fair packet access scheme that is similar to Processor Sharing (PS). Biased scheduling approaches, such as Shortest Remaining Processing Time (SRPT), have been shown to yield better throughput and faster expected completion times than PS in computer process scheduling. The WiBi MAC for Wifi approximates SRPT in order to achieve greater throughput and utilization of the wireless medium. WiBi MAC operates by having each node choose a maximum potential back off time that is proportional to the number of packets that are ready to send. By favoring packets belonging to short streams over packets belonging to longer streams, we achieve a simulated throughput improvement of XXXX over the standard fair exponential back off approach used by WiFi.

1. INTRODUCTION

Current WiFi MAC protocol is primarily designed around achieving a completely fair protocol where all nodes in the network have an equal probability to interact with the router. This allows for the network to allocate resources fairly between the nodes at the expense of reducing the overall network throughput performance. The standard WiFi protocol works well in environments where all of the data traffic is fairly similar in size and priority, but begins to suffer performance degradation when you have users with starkly different data sizes. A WiFi protocol that instead of optimizing for fairness of access optimizes for near-maximum throughput on the network would cause no negative effects when implemented in a network that has very similar data traffic size, and in the case where a network has different sizes of data, would increase the throughput of the network by having users with smaller data sizes transmit all of their data faster than a user that is sending a larger amount of data.

Current WiFi works in the following way: when a node has a data packet to send, it selects a random number between 1 and a standard minimum value which it uses to establish a back off time for the number of slots to wait to pass before sending its data. This is similar to processor sharing in which programs have a uniformly random chance to be selected for access to the processor. Processor sharing also has functionality similar to WiFi MAC's exponential back off, for when nodes have a collision, which prevent larger programs have an increased delay between times they have access to the processor to ensure that they do not monopolize the processor ensuring that the smaller programs have the ability to use the processor as well.

SRPT achieves the highest throughput, lowest mean completion time, when applied to a single channel processor [12]. It achieves this by always choosing the next program that has the lowest remaining process time, assuming that the time needed to complete processing is known at the start. A similar process can be implemented in the WiFi MAC protocol with the assumption that a node knows the size of the packet stream it needs to send. The node would then base the upper bound of the back off random number selector on the size of the packet stream that the node has left to send. As a node sends data over the network, it gradually lowers its upper bound allowing it to have a higher chance of sending its remaining packets in a shorter time frame. This approach biases the network in favor of packet streams that were small in the beginning as well as streams that are near completion. In the case that the size of the packet stream is unknown, a different way to bias the network towards streams of a smaller size is to base the upper bound of the random back off on the current frame number of the packet in the stream being sent. This approach requires that all streams start with the same range for the back off and gradually increase the range as the node sends more and more packets successfully across the network with larger streams having a higher upper bound than the smaller streams. The final approach to creating a biased protocol that improves the throughput of the network is to dynamically alter the bounds of the back off based on the current level of traffic in the network as well as the size of the stream remaining. If the node detects that the network traffic has increased and that it has a long stream remaining, then the node will increase the delay between its own packets to allow other nodes to communicate.

***** Talking with Ethan over the break to discuss the

results********

The remainder of this paper is organized as follows. In Section 2 we discuss past improvements to WiFi MAC protocols as well as some of the challenges a protocol encounters when operating in a wireless network. In Section 3 we review related work to medium access control. In Section 4 we discuss the results of the four different simulations: standard WiFi MAC, MAC protocol based on the SRPT algorithm, back off based on the current frame number in the stream, and lastly a dynamic back off window changed based on the traffic level in the network. Finally, we draw the relevant conclusions in Section 5.

2. BACKGROUND

This section is an explaination of the history and past improvements to the WiFi MAC protocol as well as some of the challenges of a wireless network. For wireless networks, a principle problem is the hidden terminal problem. In a wired network, it is possible to sense when another node is being sent, and a collision will occur [2]. In a wireless network, however, this cannot be done, resulting in the need for an alternative method of handling collisions [6]. Thus, wireless MAC protocols like Slotted-Aloha were created. In the Slotted Aloha Protocol, a node can only be sent at the beginning of a time slot [5]. This ensures that one node can finish sending before another one is sent, reducing the number of collisions. Under a network with a light load, this approach has a low chance of collision. However, collisions can still occur if two nodes are sent in the same timeslot. Under heavy loads, the probability that a node will be sent in the same timeslot as another node will increase. Most research into this problem is based on reducing collisions while keeping the distribution of sent nodes fair. An improvement on the Slotted Aloha Protocol is the Frameless ALOHA protocol, which uses a random access scheme to decide which nodes should be sent in which spot [14]. Another attempt at improving the Slotted-ALOHA protocol is the Generalized Slotted ALOHA protocol, in which nodes are sent according to their probability of being transmitted successfully. However, an issue with this protocol is that if every node is attempting to maximize its own transmission rate, then the network can jam. [9]. The Multiple Access with Collision Avoidance for Wireless (MACAW) protocol is one attempt at improving on the Slotted-Aloha Protocol and solving the hidden terminal and unfair MAC problems. MACAW introduces fairness, in which every stream sent on the network is treated equally. This contrasts our proposal in that we create an optimal MAC protocol by prioritizing certain streams, thus creating an unfair MAC protocol that optimizes the network.

3. RELATED WORK

In research done by [8] focused on the objective of improving the throughput of Wireless LAN by attempting to find an alterntive MAC protocol to the current protocol. Their researched focused on the creation of a FAIR MAC algorithm [8] that gave all nodes in a network equal priority for accessing the network. They determined that their algorithm worked to increase the throughput of a mixed data network, with VoIP and normal data, versus the current MAC algorithm. They've choosen a different path to achieve an outcome similar to this paper which suggests that there can be other ways to improve the throughput of a WiFi MAC protocol.

Other research involving unfair WiFi MAC protocols has been conducted on how to detect and prevent unfair behavior from a node on the network. Some has focused on merely detecting and blocking the misbehaving node, while others have suggested implementing a punishment of sorts on the misbehaving nodes. Ways that nodes can misbehave in order to increase their own performance is to refuse to forward packets in order to save energy, or to select a smaller backoff in order to increase throughput for their own traffic. These misbehaving nodes can degrade the network throughput for other, well-behaved nodes. [7].

Most research has been done on detecting the manipulation of the random back off [13], detecting if a node is cheating by altering the delay period between SIFS and DIFS [11]. Thise approaches tend to only benefits a single user as they crowd out other users as they maximize their bandwidth [11]. As a result it also decreases the throughput of the network as other user's data is put on hold they think the communication channel is always busy. This research acknowledges the issues that occur in the network when a single user behaves unfairly and attempts to monitor a network and correct misbehaving nodes for the improvement of the network. Their research is benefical to this paper by giving insight to current unfair methods as well as creating additional protocol requirements to be tested against in future work performed on this subject.

A similar task to creating an unfair MAC protocol that everyone follows is detecting when a single user, or a small group of users, is being unfair and the process of handling them. A few approaches to remediate unfair behavior have been to exclude the misbehaving node from routing operations, encourage nodes to cooperate by penalizing misbehavior, or to incentivise good behavior by paying nodes for cooperating. Another protocol detects unfair behavior by having a sender transmit an RTS (Request to Send) after waiting for a randomly selected number of slots in the range [0; CW].After the initial transmission between hosts, the recieving host sends with their acknowledgement: a random value that the sender then uses as the back off counter for each subsequent transmittion during the stream. [7] With this protocol, if a recieving node recieves a packet before the appropriate number of frames has passed, then the sending host is not obeying the Protocol and can then be handled accordingly. Their protocol could possibly work in the protocols suggested in this paper by using the router as the controller for who can send data when effectively turning the network in to a single core processor. The downside to this approach is the large overhead associated with the router as the cooridinator for determining when a user can send data which may negate the gains from using the protocols mention with in this paper.

For a generalized processor sharing approach to flow control the worst-case bounds on delays and backlogs are derived for the purpose of applying constraints to leaky bucket sessions in arbitrary topology networks of Generalized Processor Sharing (GPS) [10] servers. In the Generalized Processor Sharing research, allocating network to users of integrated service networks is viewed with the context of rate-based flow control being used. A major assumption made by GPS, is that the design of the network was one that used virtual circuit connection-based packets. The combination

of GPS alongside Leaky bucket admission control will allow this type of network to make a maximized range of worst-case performance calculations for packet throughput and delay; this is brought about by using a packet-by-packet service discipline called PGPS [3]. There is good flexibility across user usage since each user can get wildly different values based on their environment; however, its efficiency is based solely on the servers conserving work. With this user flexibility of service in mind, an analysis is then made of broader classes of the network. Only a subset of the sessions that are created are leaky bucket constrained, because of this each session is given succinct bounds that are independent of the behavior of any of the other sessions. The bounds are also completely independent of the networks overall topology. However, these bounds are only shown to hold for each session that is guaranteed a backlog. This backlog must also have a clearing rate that exceeds the token arrival rate of its leaky bucket, so that there arent multiple tokens interfering with the process. A much broader class of networks, called Consistent Relative Session Treatment (CRST) networks are analyzed for the case in which all of the sessions suffer from having leaky bucket constraints. The process first begins with an algorithm that has presented with characteristics that show the internal traffic in terms of their average rate and the burstiness of packets; it is shown that all CRST networks are stable. The next step in the process is a method that is presented to yield bounds on a session delay and backlog given this internal traffic characterization. The links of each route are treated collectively, which will yield tighter bounds than those that result from finding the summation of all the worst-case delays (backlogs) at each of the links in the route. The bounds on delay and backlog for each session are efficiently computed from a universal service curve, and it is shown that these bounds are achieved by using staggered greedy regimes when an independent sessions relaxation holds; the staggering comes from the propagation delay that is also incorporated into the model. Processor scheduling implies that the code segments (from here on we will refer to them as tasks) needing to be performed are to be assigned to a particular processor for execution at a particular time. Because many tasks can be potential candidates for execution, it is necessary to represent the collection of tasks in a manner that will conveniently represent the relationships among the tasks. A directed graph or precedence graph representation is probably the most popular and most appropriate representation to map processor scheduling. The nodes in these graphs can represent by independent operations or parts of a single program that are related to each other in time. The directed paths between nodes imply that a partial ordering or precedence relation exists between the tasks. Associated with each node is a second number that refers to the time required by a hypothetical processor to execute the code that is represented by the node itself. If the processors are identical, then any task can essentially run on any processor provided that its precedence requirements are satisfied. The major problem associated with the study of processor scheduling is the amount of computation time needed to locate a suitable schedule. In order to cover definitions, ley us just say that an efficient algorithm is one that requires an amount of time that is bounded in the length of its input by some polynomial and an inefficient algorithm is one which essentially requires an enumeration of all possible solutions before the best solution can be selected [4]. The algorithm whose running times are exponential in the number of tasks to be scheduled can characterize solutions of these types. For most of the problems of interest in processor scheduling, no efficient algorithm is known; in fact, interestingly enough it is actually known that if an efficient algorithm for these problems could be constructed, then an efficient algorithm could be constructed for a large family of seemingly intractable problems. That is, it is at least as difficult to compute as the hardest problem in the family of problems capable of being solved by nondeterministic algorithms in polynomial time. It includes such problems as whether or not a propositional formula can be satisfied, whether or not a graph possesses a clique of a given size, and a version of the well-known traveling salesman problem.

This paper's purpose however, is to prove that implementing an unfair protocol where network traffic is sent according to a certain priority will actually improve throughput for all nodes in the network. The proposed protocol will ideally also maintain its increased performance even when all nodes on the network are not operating under the same protocol.

4. ANALYSIS

In this paper we've decided on a theoretical approach to determining the improvement that a biased MAC protocol can give compared to a fair MAC protocol. The assumptions used in the model are, the data streams being sent follow a heavy-tailed distribution, the largest 1% of jobs comprise at least 50% of the load [1], that the network load does not exceed a value of 1, representative of the decimal percent of the network bandwidth in use, all nodes experience the same amount of delay in the network, the network does not have a central authority that dictates who can send, and finally that the network is free of collisions between packets.

These assumptions allow us to look more directly to compare current WiFi protocols to biased WiFi protocols on through-put with out the interference of outside variables.

5. CONCLUSIONS

6. REFERENCES

[1] Nikhil Bansal and Mor Harchol-Balter. Analysis of srpt scheduling: Investigating unfairness. *SIGMETRICS Perform. Eval. Rev.*, 29(1):279–290, June 2001.

[2] B. Bensaou, Yu Wang, and Chi Chung Ko. Fair medium access in 802.11 based wireless ad-hoc networks. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pages 99–106, 2000.

[3] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM Computer Communication Review*, volume 19, pages 1–12. ACM, 1989.

[4] Mario J. Gonzalez, Jr. Deterministic processor scheduling. *ACM Comput. Surv.*, 9(3):173–204, sep 1977.

[5] Ajay Chandra V. Gummalla and John O. Limb. Wireless medium access control protocols. *Communications Surveys Tutorials, IEEE*, 3(2):2–15, Second 2000.

[6] Peijian Ju, Wei Song, and Dizhi Zhou. Survey on cooperative medium access control protocols. *Communications, IET*, 7(9):893–902, June 2013.

[7] P. Kyasanur and N.F. Vaidya. Detection and handling of mac layer misbehavior in wireless networks. In *Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on*, pages 173–182, June 2003.

[8] D.E. Lucani, R.E. Badra, and C.M. Bianchi. Increasing voip capacity on wifi networks through the use of the fair algorithm for mac. In *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pages 1684–1688, Sept 2007.

[9] R.T.B. Ma, V. Misra, and D. Rubenstein. An analysis of generalized slotted-aloha protocols. *Networking, IEEE/ACM Transactions on*, 17(3):936–949, June 2009.

[10] Abhay K. Parekh and Robert G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Trans. Netw.*, 2(2):137–150, apr 1994.

[11] Maxim Raya, Jean-Pierre Hubaux, and Imad Aad. Domino: A system to detect greedy behavior in ieee 802.11 hotspots. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, MobiSys '04, pages 84–97, New York, NY, USA, 2004. ACM.

[12] Linus E. Schrage and Louis W. Miller. The queue m/g/1 with the shortest remaining processing time discipline. *Operations Research*, 14(4):670–684, 1966.

[13] M. Shanthi and S. Suresh. Detecting mac layer misbehavior in wifi networks by co-ordinated sampling of network monitoring. *International Journal of Innovative Research in Science, Engineering and Technology*, 3(1), Feb 2014.

[14] C. Stefanovic, P. Popovski, and D. Vukobratovic. Frameless aloha protocol for wireless networks. *Communications Letters, IEEE*, 16(12):2087–2090, December 2012.