A Distributed Time-Fair Scheduling Algorithm for Multi-Rate WLANs

Kwan-Wu Chin
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
Northfields Avenue, NSW, Australia
kwanwu@uow.edu.au

***Abstract*— The performance anomaly experienced by nodes in multi-rate wireless local area networks (WLANs) leads to low throughput and unfairness. To this end, we propose a distributed, token-based approach that allows applications to control nodes channel occupancy time via a single tuning knob. Our approach makes use of local information extensively and does not require any modifications to the IEEE 802.11 MAC nor require explicit signaling. Simulation results show our approach to be effective in ensuring fairness, and isolating high and low rate flows.**

I. Introduction

In multi-rate WLANs, devices have the advantage of adjusting their data rate to combat varying channel conditions. However, this leads to low throughput and unfairness. First, the performance anomaly [3] problem causes all flows to converge to the slowest data rate. This is due to the extended "air-time" taken by the slowest flow to transmit its packets. Second, the binary exponential backoff (BEB) algorithm is unfair in the short term due to a recently transmitted device having a higher probability of recapturing the channel. This becomes critical when low data rate devices recapture the channel frequently.

To illustrate the aforementioned problem, consider an experiment involving the topology shown in Figure 1. The topology has four flows, F1 to F4, and all nodes are within transmission range of each other. During simulations, we move nodes G and H increasingly farther apart, thereby lowering F1's data rate over time.

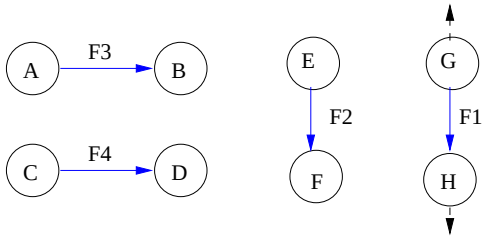

Fig. 1. Topology used in all experiments. Note, all nodes are within transmission range of each other. Nodes G and H are moved increasingly farther apart after each simulation run.

Figures 2 and 3 show significant short and long term throughput unfairness between flows. In Figure 2, F4 consistently obtains a higher throughput whilst F2 has the lowest throughput. In Figure 3, at each 200 milliseconds time intervals, we can see each flow obtains a disproportionate amount of throughput. Later, in Section IV, we show how our algorithms ensure fair throughput for flows F1 to F4.

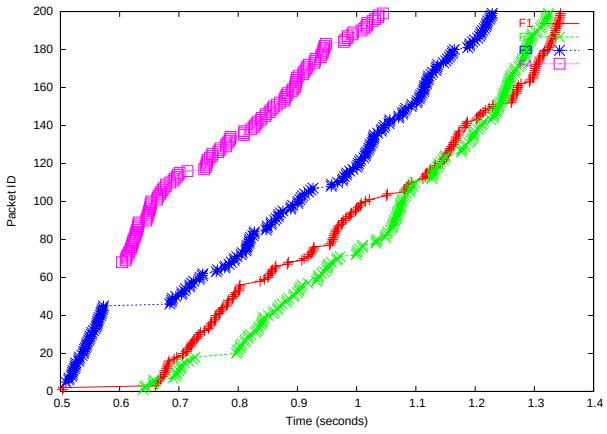

Fig. 2. Throughput of flows F1 to F4.

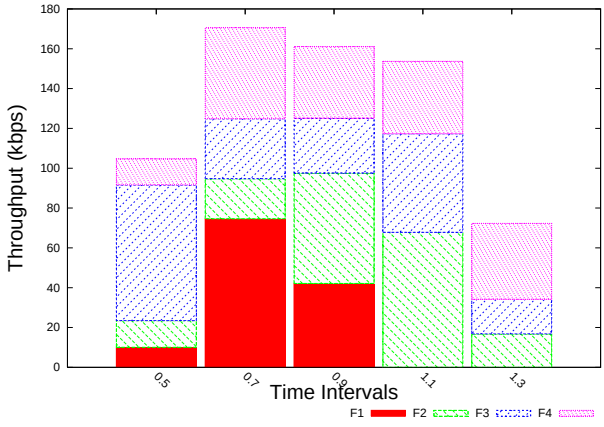

Fig. 3. Throughput proportion of flows F1 to F4 at 200 milliseconds intervals.

In light of the aforementioned problems, researchers have proposed various approaches to improve fairness in multi-rate WLANs. In [2], the author propose a two tiered scheduling architecture that groups devices with similar data rates together. Each group is then viewed as a single rate WLAN and served using a conventional wireless fair queuing scheduler. Tan et al. [11] propose a scheduler that uses a token bucket to ensure time fairness. The scheduler selects the queue with

positive tokens in a round-robin manner. Tokens are consumed by the MAC after packet transmissions. The number of tokens consumed is dependent upon packet size, data rate, protocol overheads and retransmission attempts. Both of these works and ours use tokens. However, theirs require a centralized node to allocate tokens. In [9], each node observes the length of busy periods to determine its maximum transmission time. However, a node that gains access to the channel immediately after transmission has similar performance to an unmodified IEEE 802.11 MAC since it has not had a chance to accumulate transmission time. Lastly, [10] propose to send a burst of packets whenever the channel condition is favorable. However, their work requires modifications to the IEEE 802.11 MAC and does not guarantee short-term fairness.

In the following section, we outline a distributed approach that requires no modifications to the IEEE 802.11 MAC, and guarantees nodes receive a fair share of the wireless channel.

II. The Approach

We propose to regulate channel occupancy and access using tokens. Each node is allocated a number of tokens which determines how much "air-time" it has to transmit packets. Consider Figure 4. Each node currently has four tokens and a bucket that can store up to h tokens. Assume each token is equal to a one millisecond transmission slot, and N_1 wants to transmit a 1024 bytes packet. N_1 first determines whether there are sufficient tokens to transmit the packet. Assuming an average data rate of 6 Mbps, the 1024 bytes packet will take approximately 1.4 milliseconds to transmit, ignoring protocol overheads. This means N_1 can proceed with the transmission. On the other hand, if there are insufficient tokens, N_1 will wait until the required number of tokens arrive.

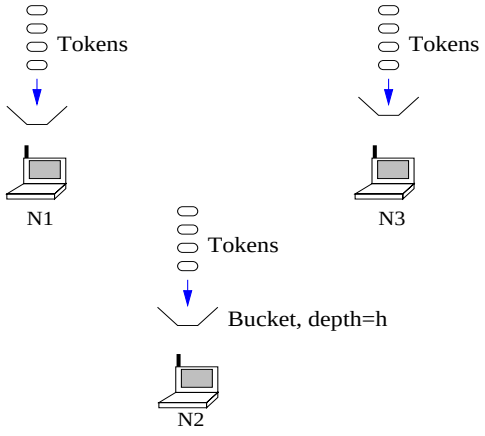

Fig. 4. Overview of proposed solution.

Each node is assigned λ tokens/second, which corresponds to the throughput required by its applications. Consider a device with a 64 kbps flow that is transmitting 100 bytes packets or 80 packets per-second. Assuming an average data rate of 6 Mbps, each packet will take approximately 133 μs to transmit; ignoring protocol overheads. If each token represents a 100 μs time slot, the device needs a λ value of 160 in order to support the 64 kbps flow. Here, 160 tokens/second assumes an error-free channel. In practice, a scheduler may use a slightly higher λ value to account for protocol overheads such as the RTS/CTS handshake, inter-frame spacings and retransmissions.

There are two techniques to determine each node's λ value. First, we can rely on a distributed admission control protocol that ensures the effective capacity of a WLAN is allocated fairly amongst devices in a WLAN. We leave the design of such protocol as future work. Second, λ can be computed dynamically using local information. We will present details of this technique in Section II-A.

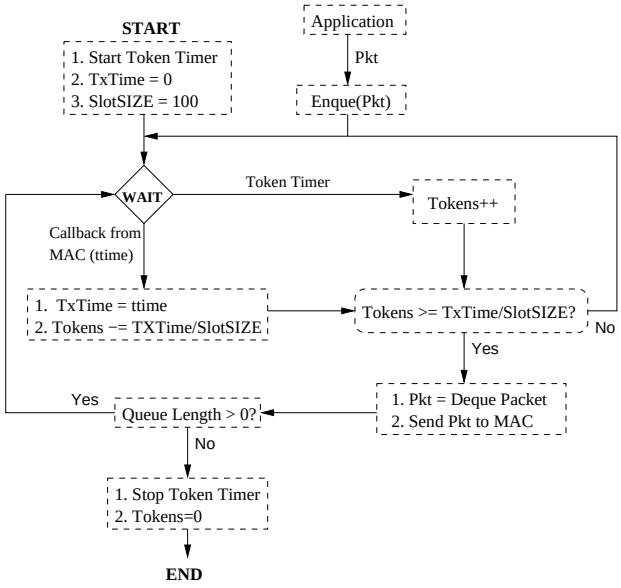

Fig. 5. Proposed algorithm to ensure fair channel access. Each token represents a 100μs slot and is generated at the rate of λ tokens/sec.

Figure 5 depicts our algorithm. The algorithm first starts a timer to generate tokens periodically and sets the variable *TxTime* (transmission time) to zero. It then moves to the wait state. Once the token timer expires, the number of tokens is increased and the algorithm checks whether it has sufficient tokens to transmit a packet. Since *TxTime* is set to zero initially, a packet is scheduled for transmission. After some time, the MAC returns the total transmission time ($ttime$) used to transmit the packet. *TxTime* is then set to $ttime$. Finally, the number of tokens consumed by the MAC is deducted from the variable *Tokens*.

Our algorithm has three key features. First, although the algorithm ensures a device has sufficient tokens before transmission, the MAC may consume more than the budgeted number of tokens due to retransmissions or a lower than anticipated data rate was used to combat worsened channel conditions. In both scenarios, $ttime$ increases. When $ttime$ exceeds the budgeted transmission time, a node experiences a token deficit, which removes it from contention. This behavior is desired because a node that exceeds its token budget has taken an unfair share of the wireless channel, hence it should "backoff" and give other nodes opportunities to transmit. Secondly, failed packet transmissions consume tokens. This is because each transmission, new or old packet, consumes "air-time" and hence needs to be accounted for. Thirdly, the variable $TxTime$ reflects the data rate used to transmit the last

packet and can be viewed as the token budget to a destination device.

To schedule multiple flows originating from a node, λ is set to $\sum_{i=1}^{F(t)} \lambda_i$, where $F(t)$ is the number of flows at time t. Upon token arrival, any scheduling policy can be used. For example, a node may choose the highest rate flow. Thereby, minimizing the number of tokens used. Alternatively, it could choose a queue that has the earliest deadline or simply serves flows in a round-robin manner.

A. Dynamic Token Rate Estimation

We now present a distributed algorithm that enables each node to dynamically compute its λ value.

Each node runs the following algorithm periodically:

1) $Bits^{max} = NAV^{max} \times \bar{R}$. NAV^{max} is the maximum network allocation vector observed by the MAC within a given *period*. $\bar{R}$ is the node's average data rate.
2) $nTokens = \lceil \frac{Bits^{max}}{\bar{L}} \rceil$. $\bar{L}$ is average packet size.
3) $\lambda = \lfloor \frac{nTokens}{NAV^{max} \times N} \rfloor$. N is the number of nodes that are observed transmitting within a given *period*.

Consider the following example. Assume there are three flows in a WLAN, A, B and C. Flows A and B run at 54 Mbps whereas Flow-C runs at 6 Mbps. All flows transmit 1024 bytes packets. In this example, Flow-C will have the longest channel occupancy time, i.e., $NAV^{max} = 1.36ms$. This means flows A and B will have a $Bits^{max}$ value of 73440 bits. At Step-2, flows A and B have $nTokens = 9$, which equates to a λ value of 2206. Flow C has $\lambda = 245$.

There are a few key features to note. First, the algorithm only guarantees per-node fairness, and each node is responsible for dividing the computed λ value to its flows. Second, all required information are computed locally. Third, to estimate N, we can use algorithms such as [1]. However, in our simulations, N is estimated to be the number of unique hosts that transmitted within a given *period*, where *period* is a configurable parameter and it is set to 100 milliseconds in our simulations.

B. Discussions

Our approach addresses several issues that arise from the distributed nature of WLANs, and vagaries of the wireless channel.

First, as mentioned, the BEB algorithm as used by IEEE 802.11 [5] is unfair in the short-term [7]. In multi-rate systems, there is a risk that a low rate flow may recapture the channel continuously, which significantly lowers the throughput of other flows. Ideally, each node should gain access to the channel once every "round". That is, nodes should be given fair access to the channel as well as fair occupancy time. In this respect, a key investigation is to determine whether tokens improve short-term fairness. An observation here is that if a node only has a fixed number of tokens within a short time period, it will only contend and transmit as long it has tokens. On the other hand, if too many tokens are allocated, then short-term unfairness increases; approaching the case where no tokens are used. In this respect, our algorithms aim to improve contention and fairness using tokens.

Second, the vagaries of the wireless channel have a significant impact on token usage. A node does not know how many tokens are required to send a packet. In other words, the channel condition to a neighbor is unknown until it tries to send a packet. The node could probe the channel continually to determine whether it has sufficient tokens to transmit a packet. However, the probing process and overheads such as inter-frame spacings consume precious "air-time" or tokens. This means a node could used up all its tokens probing the channel and not have any left to transmit data packets. For this reason, our approach only requires nodes to determine whether it has sufficient tokens to transmit a packet using the last data rate, and allows nodes to run their tokens into deficit.

Third, as shown in [2], throughput fairness is impractical unless flows with similar data rates are grouped together. For example, a 6 and 54 Mbps flow is unlikely to have the same throughput in the long term, given that the 54 Mbps flow transmits approximately 10 times more data. The scheduler could suspend the 54 Mbps flow to allow the 6 Mbps flow to catch up, but doing so would result in significant packet delays for the 54 Mbps flow. In this regard, the use of tokens effectively abstracts the disparate data rates provided by the physical layer, and allows developers to focus on application level throughput fairness.

Lastly, in terms of fairness, we like to point out there is little difference between allocating tokens on a per-flow or per-node basis. On a per-node basis, a node has the advantage of selecting a flow that best utilizes its tokens. For example, picking the highest data rate flow will minimize the number of tokens used. On the other hand, on a per-flow basis, once a flow captures the channel and finds itself with insufficient tokens, it cannot pass the channel to a different flow. Nevertheless, in both cases, each flow has equal tokens.

III. Simulation

To investigate our approach, we augmented *ns-2* (v2.29) [8] with IEEE 802.11a [5], thus giving us a WLAN with eight data rates; 6 to 54 Mbps. In addition, all nodes accumulate the interference caused by simultaneous transmissions and use it to calculate the signal-to-interference of each packet - hence its packet error rate. We use the shadowing radio propagation model included in *ns-2*. All our simulations use a standard deviation value (*std_db*) of 4.0 and path-loss exponent of 2.0. Apart from that, we have implemented the RBAR [4] link adaptation algorithm.

The simulation scenario/topology is the one described earlier in Section I. We calculate the throughput of each flow and measure its short-term fairness using Jain's fairness index [6] over 100 milliseconds intervals. The λ value of each node is calculated dynamically as per the algorithm in Section II-A. Lastly, all flows transmit at a constant bit rate of 1K packets/second, each packet is 512 bytes in size.

IV. Results

We first show what happens when our algorithm is disabled. Recall that flow F1's data rate is lowered over time by moving nodes G and H increasingly farther apart. Figures 6 and 7

show the throughput obtained for each flow when there are no errors. Flows F2 to F4 initially have to contend with F1. As the distance between nodes G and H increases, flows F2 to F4 begin to observe a decrease in throughput, so called performance anomaly. On the contrary, when we have packet errors, the throughput of flows F2 to F4 increases as nodes G and H move apart. This is due to flows capitalizing on the additional capacity when other flows enter backoff after experiencing errors or collisions.

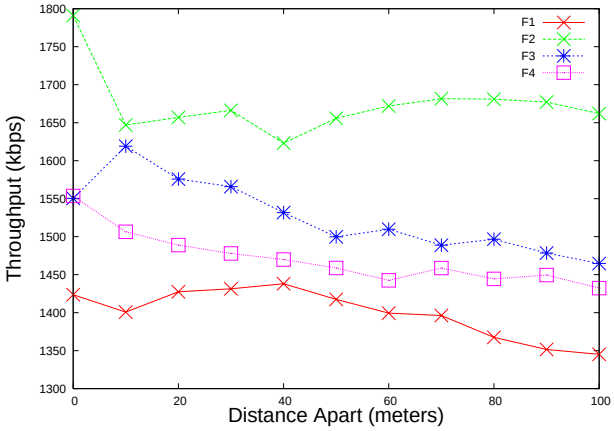

Fig. 6. Throughput (Error-Free) - Proposed algorithm disabled.

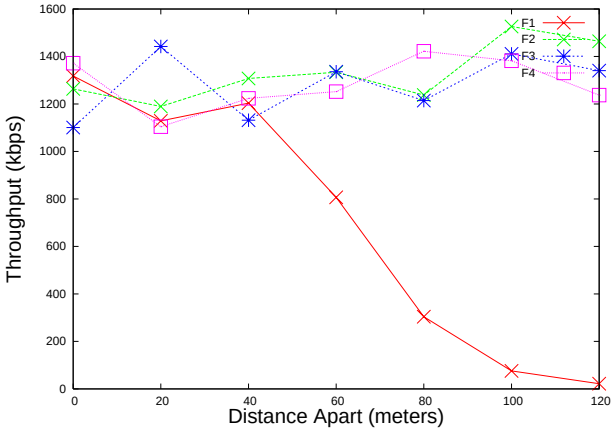

Fig. 7. Throughput (Error Prone) - Proposed algorithm disabled

We now enable our algorithms. Figures 8 and 9 show the throughput achieved by each flow. We see that flows F2 to F4 are able to sustain their throughput in the presence of the low rate flow F1 - thus overcoming the performance anomaly problem. Moreover, flows F2 to F4 have almost equal throughput. To quantify this fact, Figures 10 and 11 show the proportion of throughput received by all flows and the corresponding Jains's fairness index over 100 milliseconds intervals. We see all flows have equal throughput over all periods, as verified by the high Jain's fairness index value. From these results we can conclude that our algorithms can indeed provide short-term and hence long-term fairness.

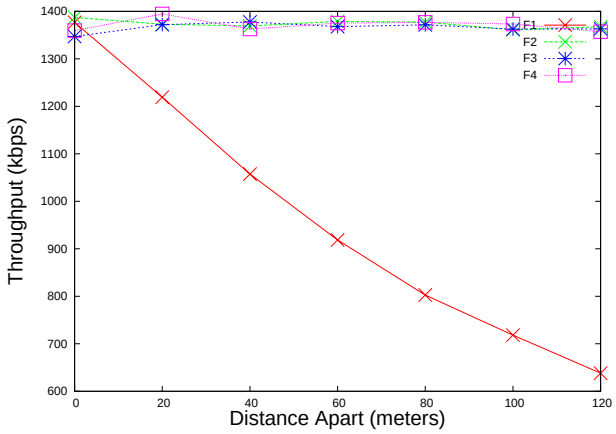

Fig. 8. Throughput (Error-Free) - Proposed algorithm enabled.

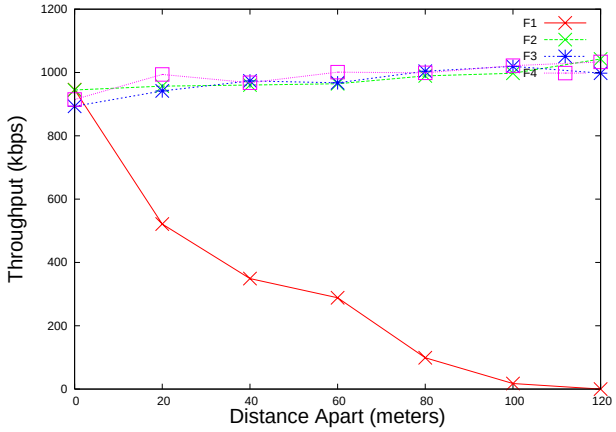

Fig. 9. Throughput (Error Prone) - Proposed algorithm enabled.

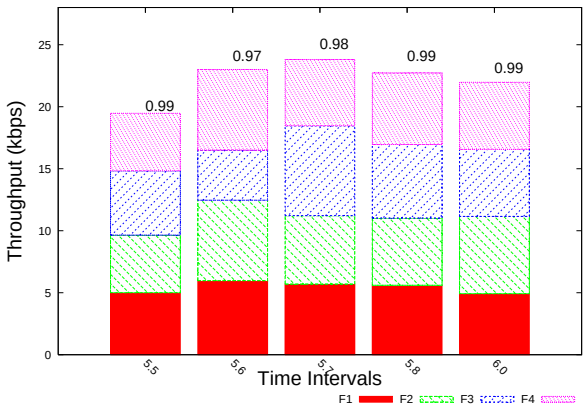

Fig. 10. Fairness (Error-Free).

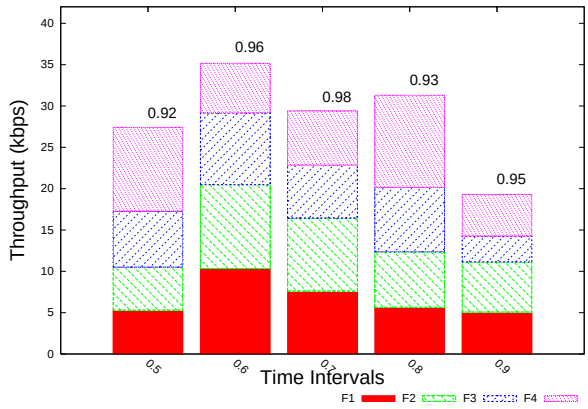

Fig. 11. Fairness (Error Prone).

V. Conclusion

Our algorithm addresses the performance anomaly problem in a distributed manner. Advantageously, it enables protocol designers to control fairness via one tuning knob: *token rate*. Moreover, it operates without any changes to IEEE 802.11 and relies only on local information. In particular, our dynamic token rate estimation algorithm does not require inter-node communication in order to achieve fairness.

Currently, we are investigating whether our algorithm is suitable for multi-hop wireless ad-hoc networks. Preliminary analysis indicates our algorithm is applicable. Results are forthcoming.

References

[1] G. Bianchi and I. Tinnirello. Kalman filter estimation of the number of competing terminals in an IEEE 802.11 network. In *Proceedings of IEEE INFOCOM'2003*, San Francisco, USA, 2003.

[2] K.-W. Chin. T^2-fair: A two-tiered time and throughput fair scheduler for multi-rate WLANs. In *ACM MSWiM'2006*, Torremolinos, Spain, Oct. 2006.

[3] M. Heusse, , F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Proceedings of IEEE INFOCOM'2003*, San Francisco, USA, 2003.

[4] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *ACM MOBICOM'2001*, Rome, Italy, Oct. 2001.

[5] IEEE. Part 11a: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: High speed physical layer in the 5 GHz band, standard specification. IEEE Std 802.11a-1999, 1999.

[6] R. Jain, D.-M. Chiu, and H. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. Digital Equipment Corporation, Technical Report - RR TR-301, 1984.

[7] C. E. Koksal, H. Kassab, and H. Balakrishnan. An analysis of short-term fairness in wireless MACs. In *ACM SIGMETRIC*, Santa Clara, CA, USA, June 2000.

[8] S. McCanne and S. Floyd. ns network simulator-2. http://www.isi.edu/nsname/ns/.

[9] T. Razafindralambo, I. Guerin-Lassous, L. Iannone, and S. Fdida. Dynamic packet aggregation to solve performance anomaly in 802.11 wireless networks. In *ACM MSWiM'2006*, Torremolinos, Spain, Oct. 2006.

[10] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad-hoc networks. In *ACM MOBICOM*, Atlanta, Georgia, USA, Sept. 2002.

[11] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate wireless LANs. In *Proceedings of USENIX*, Boston, USA, June 2004.