A simple congestion-resilient link adaptation algorithm for IEEE 802.11 WLANs

Hyogon Kim, Sangki Yun, Heejo Lee, Inhye Kang, and Kyu-Young Choi

***Abstract*— Algorithmic approach to link adaptation for IEEE 802.11 networks such as Automatic Rate Fallback (ARF) is known to suffer from the inability to differentiate between collision and channel-induced error. In this paper, we propose a novel algorithm called COLA that overcomes the shortcoming and achieves near-optimal throughput over wide range of channel and load conditions. The result is significant since the throughput is achieved without any hardware support. Moreover, COLA does not require any optional or extra-protocol mechanisms support, either, such as RTS/CTS exchange, Clear Channel Assessment (CCS), and promiscuous channel monitoring. Finally, the COLA algorithm has a short critical path of just 10 instructions, and it is free of heuristic parameters, which will facilitate practical use.**

I. Introduction

The IEEE 802.11 physical layer (PHY) supports multiple transmission rates with varying modulation and channel coding scheme [1]. This feature allows the 802.11 wireless LAN (WLAN) to dynamically cope with the changing channel condition and protect the throughput performance. For instance, 802.11a standard has 8 PHY "modes" that are each optimal for different ranges of channel SNR [2]. The procedure of selecting the best among the provided modes is called the *link adaptation*. One approach to link adaptation is to make the sender monitor the channel condition using hardware and determine the optimal mode [3], [4]. However, this approach is considered to involve extra implementation effort or modifications to current 802.11 standard [5].

A cheaper and yet efficient alternative approach is exploiting the 802.11 acknowledgements (or lack thereof). Automatic Rate Fallback (ARF) [6] is a representative example, and it is widely implemented in commercial products. It interprets the consecutive failures to receive 802.11 acknowledgements (ACKs) as a sign of bad channel condition. In practice, however, the lack of ACK could signal bad channel, collision, or even both. This is the biggest problem of the ACK-based approaches. The judgement whether the transmission failure was channel-induced or collision-induced can only be probabilistic, so the approaches inevitably run a gamble when interpreting the transmission failure. And waiting for consecutive transmission failures (*e.g.* two in [5], [6]) still leaves a significant chance of misjudging collisions with a channel error (Fig. 1).

The performance cost of misjudgement can be significant. If collision is interpreted as bad channel, the link adaptation can be unnecessarily triggered, leading to less efficient use of the network. Choi *et al*. [5] and Kim *et al.* [7] independently report that under some circumstances turning on the link adaptation leads to far inferior throughput than simply letting the system run at a high, fixed rate. For instance, certain 802.11g system only yields 11 Mbps with the default link adaptation scheme, while 30 Mbps or more could be achieved by simply fixing the transmission rate at 54 Mbps [5].

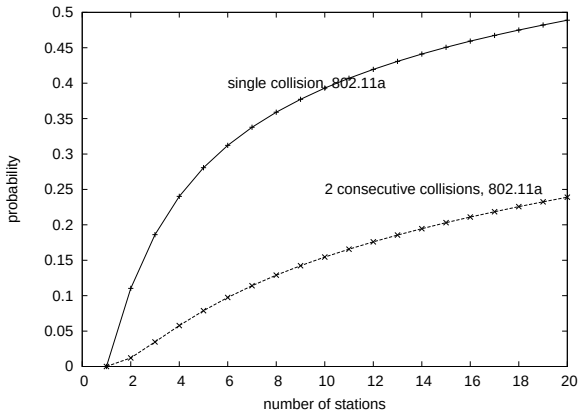

Fig. 1. (Consecutive) collision probability vs. number of stations in 802.11a.

The lack of capability to completely differentiate collision and channel-induced errors in existing ACK-based link adaptation schemes is noticed in [5] among others. However, there is dearth of prior work on this subject in the literature. Recently, Kim *et al*. [8] proposed an RTS/CTS-based scheme to differentiate the channel-induced error and collision. It exploits the short length of the RTS frame to reduce the error in differentiation. However, the RTS/CTS exchange is an optional feature of 802.11, so it is undesirable to rely on it. Moreover, the relative overhead of RTS/CTS exchange itself becomes non-trivial as the link speed increases [9], so this approach is likely to become increasingly costly and less effective.

In this paper, we propose a novel link adaptation algorithm that is robust against congestion. The contributions of the proposed idea can be summarized as follows:

- Instead of using a hardware, an algorithm with the 10-instruction critical path achieves close-to-optimal throughput performance over wide range of SNR values.
- It does not rely on any optional or extra-protocol mechanisms, in particular RTS/CTS, Clear Channel Assessment (CCA) or promiscuous channel monitoring.
- It is free of any heuristic parameters to be configured, which facilitates practical use.

We use the example of 802.11a in this paper, but the idea directly applies to faster 802.11 networks without any additional

This research is supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

overhead.

II. Congestion-resilient link adaptation

The whole idea of congestion-resilient link adaptation is simply ignoring the collision portion of the failure "signal". In order to take out the contribution from collision in a given failure event, we would need to know P_c, the collision probability involved in the failure event. Under perfect channel condition, measuring P_c would be trivial, *i.e.*, equate P_c with the transmission failure rate P_f. However, transmission failures under imperfect channel could be due to channel errors. Therefore, in general, $P_c \neq P_f$. In a *saturated* network, an alternative could be to reverse Bianchi's Kalman filter estimation of n from P_c [10] to get P_c out of n, where n is the number of observed nodes in activity. Unfortunately, this technique is impractical. First of all, the saturated system is a very strong assumption. Second, even if the system is saturated, monitoring n would require promiscuous reception of MAC frames, which could cost battery-operated units dearly.

In this paper, we develop an algorithm that does not require the exact knowledge of P_c. We will discuss later how it can be done, but for the moment let us assume that we somehow know P_c and develop a baseline link adaptation algorithm on it. For simplicity, we assume that P_c and the channel error probability P_h are disjoint, *i.e.*, $P_f = P_c + P_h - P_c \cdot P_h \approx P_c + P_h$. Given N_f transmission failures and N_t transmission attempts in the current mode, respectively, we estimate the average number of channel induced errors $\bar{H} = N_t \cdot P_h = N_f - N_t \cdot P_c$. As soon as we get $\bar{H} \geq k$, where k is a preset threshold, we drop the mode. In the baseline algorithm, we set $k = 1$, (the mode drop is cued by a single channel error) as all other existing link adaptation algorithms do. We call this algorithm COLA (COngestion-resilient Link Adaptation). In Fig. 2, the baseline COLA algorithm is presented, where we also use the following notations:

- m: current mode, *i.e.*, [1..8] in IEEE 802.11a
- N_s: number of consecutive successes in m
- u_m: number of required consecutive successes at m for mode increase

The COLA algorithm is executed for each transmission attempt. If the attempt fails, N_f is incremented. What may not be too intuitive is that the success count N_s is also increased by P_c. This is because statistically, we regard that the collision contributes P_c to each failure, and the channel contributes the rest. From the perspective of link adaptation, the collision portion P_c of a failure is still a "success," because it is not caused by the channel. The idea here is that we reflect only the channel-induced portion to the failure count that the link adaptation algorithm must heed.

Upon mode drop to m, we check if all transmission attempts failed in $m+1$. In that case, the next mode-up attempt to $m+1$ is made more conservatively. We use the exponential backoff scheme proposed in Choi *et al.* [5], so we double u_m. For a successful transmission, we check if the number of consecutive successes so far at m exceeds u_m. If so, we attempt to go up to $m + 1$. When the decision is made to go up one mode to $m + 1$, the u_{m-1} is reset to 1, since we succeeded at least once in mode m.

```
N_t = N_t + 1
if (transmission_failure)
    N_f = N_f + 1
    N_s = N_s + P_c /* collisions are counted as "successes" */
    H̄ = N_f − N_t · P_c /* how many channel-induced errors? */
    if (H̄ ≥ k) /* enough channel error(s), go down */
        N_s = 0 /* consecutive "successes" end here */
        if (m ≠ 1) /* can go down? */
            m = m − 1 /* drop mode */
            if (N_t == N_f) /* none succeeded at higher mode */
                u_m = 2 · u_m /* next time be more conservative */
            N_t = N_f = 0 /* start fresh in the new mode*/
else /* success */
    N_s = N_s + 1
    if (N_s ≥ u_m)/* enough consecutive successes */
        if (m ≠ 8) /* can go up? */
            u_{m−1} = 1 /* backoff unnecessary below */
            m = m + 1 /* up the mode */
            u_m = 1 /* start at basic value */
            N_s = N_t = N_f = 0 /* start fresh in the new mode */
    else if (m ≠ 1) /* not enough successes yet */
        u_{m−1} = 1 /* no mode "0" */
```

Fig. 2. Pseudo code of the COLA algorithm, executed for each transmission.

III. Algorithm refinement

In this section, we conduct a stepwise refinement of the baseline algorithm. We will call the refined algorithms COLA2 and COLA3, where the refinement culminates with COLA3. For experiments, we simulate a 802.11a BSS with n wireless stations equipped with link adaptation algorithms, all within mutual reception range. These n nodes transmit CBR traffic in UDP/IP encapsulation at their maximum allowed speed. The `ns-2` [11] simultor is used after enhancing the 802.11 module to support the 802.11a PHY. We use the empirical BER vs. SNR curves provided in Heiskala and Terry [12], to estimate the FER.

A. *Baseline COLA algorithm* vs. *ARF*

Fig. 3 shows the impact of COLA on throughput as a function of the channel quality in dB. We also plot the best ARF performance for comparison. As to ARF, there are two variants, which differ in the number of consecutive successes required for mode increase. One requires 3, and the other, 10. We call the former "ARF3", and the latter, "ARF10." For rate drop, two consecutive failures are prescribed for both variants. As expected, the ARF performance is severely affected by collisions. At $n = 10$, for instance, the throughput of the best performing ARF[1] is bounded by 5Mbps almost irrespective of the channel condition. And COLA consistently outperforms ARF. In one extreme, the throughput difference is over 10Mbps at $n = 10$, 20dB. With the maximum effective

[1]For $n = 1$, ARF10 is better, whereas for $n = 5$ and $n = 10$ ARF3 is. The reason is that ARF3 is quicker to try a higher mode and it helps the ARF3 to rebound to a suitable mode faster, if collisions are the more dominant cause for the rate drops. Even for $n = 1$ where there is no collision, performance difference arises between ARF mechanisms due to the difference in the mode-up attempt thresholds.

throughput of 30Mbps in 802.11a, the difference is significant. The throughput difference is more acutely pronounced as channel improves, since the dominant cause for transmission failures is collisions.

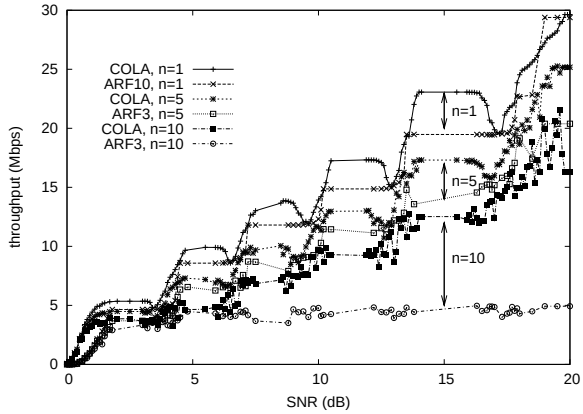

Fig. 3. Throughput performance of COLA vs. ARF.

We can measure how well COLA prevents wireless stations from incorrectly decreasing the mode on collisions, by plotting the mode (prescribed by COLA) against the channel quality. If COLA completely prevents it, the mode function should remain the same as n changes. In Fig. 4, `BESTFIX` is a hypothetical scheme that knows *a priori* the highest-performing mode. We obtain it through simulating the fixed modes from 1 to 8, and then choosing for each simulated SNR value the best-performing mode among them. Note that this hypothetical scheme is immune to the effect of n. For $n = 10$, we compare COLA with ARF3, which performs better than ARF10. In the figure, we notice that ARF is greatly affected by the increase of n. At $n = 3$, it already moves far down from BESTFIX. It is suppressed to mode 1 at $n = 10$, which bears out the throughput result (≈5Mbps) in Fig. 3. Note the throughput is smaller than the PHY rate of 6Mbps due to the encapsulation and channel access overheads. In contrast to ARF, COLA generally succeeds in maintaining the average mode close to the BESTFIX value, nearly independent of n. But we also notice that COLA visibly overshoots the mode in the transition region. It reveals that the "dips" in the throughput graph are caused by the excessive aggressiveness of COLA in the region.

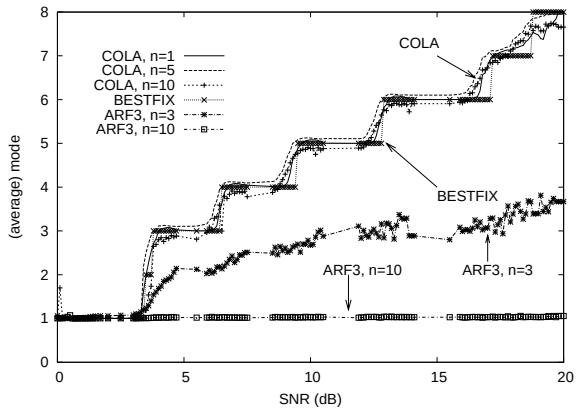

Fig. 4. The mode dynamics of COLA vs. ARF under varying channel quality.

B. Reflecting rate ratios in mode change

Although the baseline COLA significantly improves the throughput over ARF, the comparison with BESTFIX reveals that there is much room to improve (Fig. 5). Not only in the mode transition regions but also in the plateau does the throughput degradation persist. At this juncture, we remark

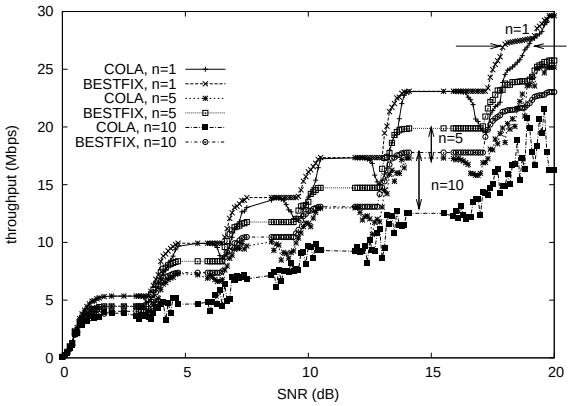

Fig. 5. Throughput comparison between COLA and BESTFIX.

that implicit in existing ACK-based link adaptation algorithms are the following assumptions:

- *A single channel-induced error* is a sufficient signal to drop the mode.
- *A single successful transmission* in the increased mode is enough to warrant the increase.

We argue that they are the source of performance degradation. One of the novelties of this paper is that we provide a technical basis for the exact number of channel-induced errors and successful transmissions to justify the up and down mode transitions, respectively. And we show below that it improves the COLA performance of Fig. 5 close to the optimum.

1) Mode drop: Due to the large rate difference between neighboring modes (*e.g.* in 802.11a it could be as large as 12Mbps), tolerating occasional losses at mode m may be much more profitable than transmitting errorless at mode $m-1$. Also considering the potentially harmful impact of the mode difference among stations that are reported in Heusse *et al.* [13], we should carefully decide the drop threshold k in terms of the rates. Let r_m be the nominal bitrate provided by the mode m and $p_h^{(m)}$ be the estimated channel-induced FER given by the current SNR under mode m. Then, it is more profitable to stay at m than to go down to $m-1$ if

$$(1 - p_h^{(m)}) \cdot r_m > r_{m-1} \quad (1)$$

$$\Rightarrow \quad 1 - p_h^{(m)} > r_{m-1}/r_m. \quad (2)$$

For IEEE 802.11a, the rate ratio r_{m-1}/r_m is either 0.66 or 0.75, with an exception of 0.89 for between mode 7 and 8. We remark that we have $1 - p_h^{(m-1)} \approx 1$ multiplied to r_{m-1} in the RHS of Eq. (1). Namely, under the channel condition that satisfies Eq. (2), we assume that the channel-induced FER at $m-1$ is close to 0. In the IEEE 802.11a, the modes usually satisfy the assumption (see Fig. 6). One notable exception is modes 2 and 3 whose BER curves almost concur. Other

than this, there is little overlap. It means that usually we do not see channel-induced errors at mode m even when we are experiencing significantly high error rates at mode $m+1$. Since r_m for every m is given, their ratios can be precomputed and quickly referenced to check for the condition in Eq. (2). Notice the 1% BER ceiling in the graph is extremely high – it is high enough to corrupt even a tiny packet, *e.g.*, of 100-bit length under uniform bit error distribution.

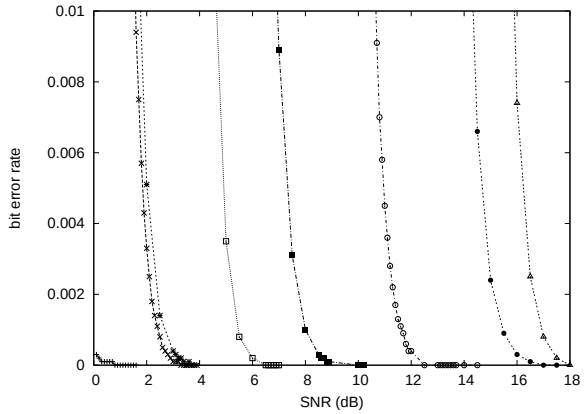

Fig. 6. BER characteristics of 802.11a modes.

As a consequence of enforcing Eq. (2) in the mode decrement decision, the COLA algorithm becomes more cautious in decrementing the mode. When there are non-zero channel-induced errors but the number is not enough to justify the drop in face of the rate differences, the modified COLA retains the current mode. In the pseudo code, the failure processing condition should be modified as follows:

```
    if (H̄ ≥ k) /* enough channel error(s), go down */
⇒ if (1 − H̄/N_t < r_{m−1}/r_m) /* go down */
```

And notice, we eliminated the heuristic parameter k.

2) Mode increase: Like in Eq. (2), it can be more profitable to stay at the current mode m if

$$(1 - p_h^{(m+1)}) < r_m / r_{m+1}. \quad (3)$$

Here, $p_h^{(m+1)}$ can be computed by tracking the successes and failures for the attempts to transmit at r_{m+1}. The difficulty is, however, that we need to measure it while staying at m. The whole point is not going up until we collect enough statistics about $m+1$. Staying at m, we perform $a_t \geq 1$ tests for $m+1$ recording the number of successes s_t. Then we calculate $p_h^{(m+1)} = 1 - s_t/a_t$ and check with Eq. (3). The a_t parameter can be any reasonably large number so that s_t/a_t has a comparable precision with r_m/r_{m+1}. It is set to 4 in this paper. A caveat in applying the tests above is that when a test transmission fails, it should not trigger the mode drop logic. So a flag that signals the test under way is added. This flag is checked before the transmission failure logic is executed. For convenience, we will call the algorithm COLA2. The mode-up test portion of the COLA2 logic is described in Fig. 7. This test is hardly executed when $p_h^{(m+1)}$ is very high. Namely, in the "plateau" region, the mode increase is not possible due to

```
test_processing:
    a_t = a_t + 1
    if (a_t ≥ T) /* tests completed */
        if (m ≠ 1) m = m − 1 /* restore mode */
        if (s_t/a_t ≤ r_m/r_{m+1}) /* cannot go up */
            u_m = 2 · u_m
            N_s = N_f = N_t = 0
        else /* high enough success rate, go up */
            u_{m−1} = 1; m = m + 1; u_m = 1; N_s = N_f = N_t = 1
        up_test = 0 /* out of the test */
    return
```

Fig. 7. Mode-up test of the COLA2 algorithm.

low success rate at $m+1$ anyway, and the modification does not cause any effect. It is only when $p_h^{(m+1)}$ is significantly small, namely, we are at the right edge of the SNR region for the mode m, that the modification suppresses a hasty increase decision.

Fig. 8 compares the throughput of the COLA2 and BESTFIX. The throughput curves are virtually on top of each other, except for the mode transition regions for 6→7 and 7→8. We can view the impact of the modification from the mode

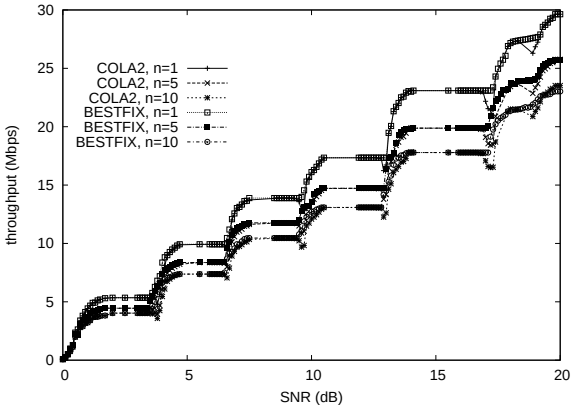

Fig. 8. Throughput comparison of COLA2 and BESTFIX.

dynamics angle in Fig. 9. The average mode under COLA2 is closer to the optimum than under COLA for both plateau and mode transition regions. But one might wonder why the

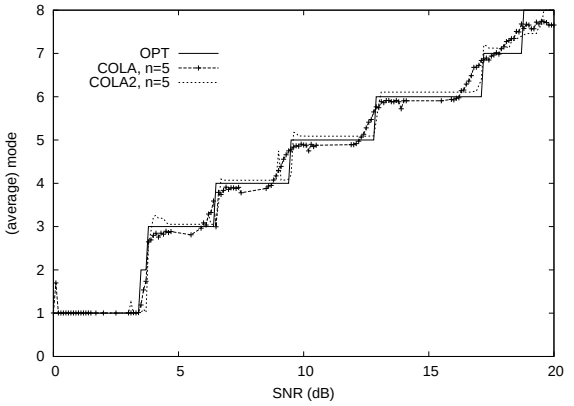

Fig. 9. Impact of the modification to the mode dynamics, $n = 5$.

throughput difference between the baseline COLA (Fig. 3)

and COLA2 (Fig. 8) is so large in the plateau region, since the average mode difference is not too large. Looking at the variation in the selected mode by the baseline COLA and COLA2 gives us the answer. Fig. 10 shows that the standard deviation of the baseline COLA is much higher. Even in a very stable region around 15dB, the deviation of the baseline COLA is close to 0.25, whereas COLA2 has almost 0. It implies that the baseline COLA mode selection is more likely to fluctuate off the BESTFIX. Namely, it stays more frequently in the sub-optimal mode than COLA2. Note the transmission in the mode higher than the optimal mode likely leads to failure, while in the lower suboptimal mode the transmission time is significantly elongated. So the throughput performance degrades significantly in the baseline COLA. In essence, the

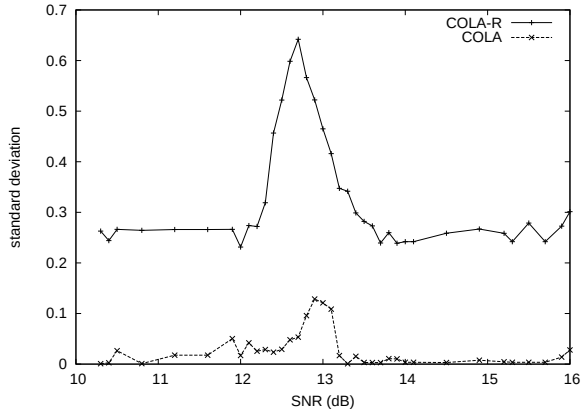

Fig. 10. Mode standard deviation of COLA2 and COLA, 10–16 dB.

rash decision to shift mode upon a single channel-induced failure or success is the source of the instability in the baseline COLA and other ACK-based link adaptation schemes. COLA2 minimizes the variability through Eq. (2) and (3).

C. Operation in the ignorance of P_c

In this section, we finally relax the impractical assumption that we know P_c. We modify COLA2 so that it works without knowing P_c. First, let us consider the plateau region for mode m, since it occupies most SNR ranges. Here, the 802.11a mode design has the property that the error rate at mode $m+1$ is extremely high. So any transmission at $m+1$ should fail. Looking back on Fig. 1, up to a large number of nodes we have $P_c < 0.5$. Then the two initial transmission attempts that fail at mode $m+1$ would render

$$\bar{H} \quad = \quad N_f - N_t \cdot P_c \geq 2 - 2 \cdot 0.5 = 1$$

and the baseline COLA would drop the mode back to m. More importantly, COLA2 will also drop the mode since

$$1 - \bar{H}/N_t \leq 1/2 < r_{m-1}/r_m$$

since the rate ratio is at least 0.66 for IEEE 802.11a. For a single transmission error the mode drop will not occur.

Even if we do not know the exact P_c value, we can exploit the above property. Namely, we can simply drop the mode upon two consecutive losses, *subject to* the rate ratio check. Surprisingly, this is the only modification required to give up the assumption about P_c in COLA2 since in the mode-up logic it does not use P_c anyway. We call this modified algorithm COLA3. Since we do not know P_c in COLA3, we use $P_f = N_f/N_t = P_c + P_h$ for P_h. One might think that it is reverting to ARF. Indeed, since P_f includes P_c, we can expect that as n grows P_f is erroneously inflated by P_c, adversely affecting the COLA3 performance. However, the rate ratio check curbs the mode from going down simply because 2 consecutive failures occur (Eq. (4)). We can see the impact of this additional check in the performance difference between COLA3 and COLA3 without the check ("`COLA3-ratiocheck`", Fig. 11). At $n = 30$ and at 20dB, collisions dominate. But the COLA3 throughput is 23Mbps with the check *vs*. 13Mbps without the check. It is not only comparable to COLA2 but also to the theoretical maximum given by Bianchi *et al.* [10].

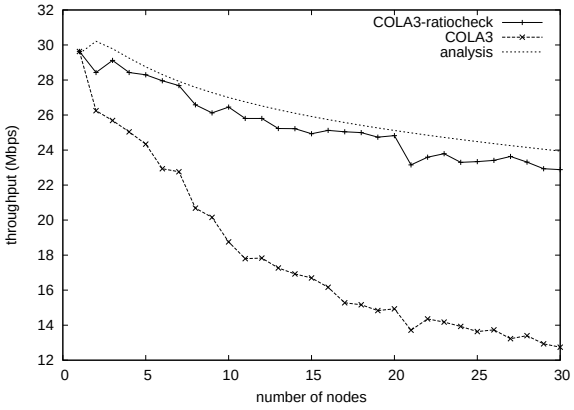

Fig. 11. The impact of the rate ratio check, 20dB channel.

In essence, the drop check in the failure processing is modified as follows:

$$\begin{aligned} & \text{if } (1 - \bar{H}/N_t < r_{m-1}/r_m) \text{ /* go down */} \\ \Rightarrow \quad & \text{if } (N_{cf} \geq 2 \;\; and \;\; 1 - N_f/N_t < r_{m-1}/r_m) \end{aligned} \tag{4}$$

where N_{cf} is the number of consecutive failures. It is reset to 0 upon a transmission success, and incremented upon a failure. Recollect that this is not executed when the mode-up test is under way. Fig. 12 shows the throughput of COLA 3 is quite close to that of COLA2, which in turn is close to that of BESTFIX (see Fig. 8).

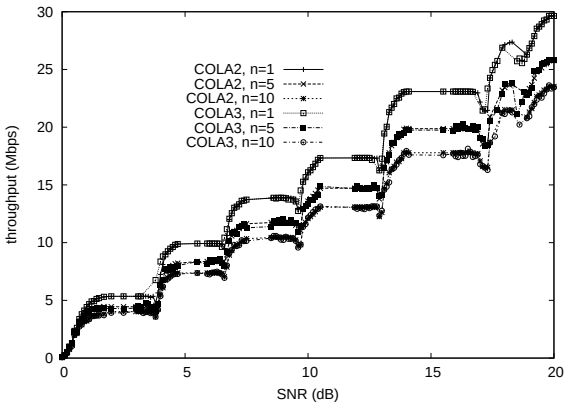

Fig. 12. Throughput of COLA2 and COLA3.

```
N_t = N_t + 1
if (transmission_failure)
    if (up_test)
        call test_processing
    else /* not testing */

    failure processing (box, Fig. 2) with modification of Eq. (4)

else /* success */
    if (up_test)
        s_t = s_t + 1
        call test_processing
    else /* not testing */
        N_s = N_s + 1
        if (N_s ≥ u_m and m < 8)
        /* enough successes at current mode: start tests */
            a_t = s_t = 0; up_test = 1;
            m = m + 1 /* temporarily move to the tested mode */
        else if (m ≠ 1)
            u_{m-1} = 1 /* backoff unnecessary below */
```

Fig. 13. Pseudo code of the COLA3, the finalized algorithm.

Visible deviations are found in the transition region $7 \rightarrow 8$, but in most SNR regimes the throughput performance is quite close to optimal. This is because for $7 \rightarrow 8$, the rate ratio (the last term of Eq. (4)) is quite high in 802.11a: $r_7/r_8 = 48/54 = 0.89$, so a very high success rate is required to stay at mode 8. Analyzing the dynamics in the transition region is general tricky. The channel-induced error probability for mode-up attempts improves, but the inclusion of P_c offsets the effect. Numerical analysis is under way, but the simulation results of this modification show that it leads to similar throughput performance as COLA2 for most SNR regimes.

Although the finalized algorithm looks complicated to be executed for each transmission attempt, it is light-weight. The critical path through the code is taken for the transmission failure immediately before the mode drop. It is composed of 10 statements. Other cases are much shorter. As to the algorithm configuration, only a_t parameter needs it. However, it just affects the precision of the mode-up test. As we mentioned earlier, it has only minimal impact on performance (experiments not shown due to space).

IV. Related work

There are several proposals for link adaptation in IEEE 802.11 networks. Many are built on accurate channel-estimation information [3]–[5], [14], [15] and there are also ACK-based schemes [6], [16]. In particular, the ARF scheme [6] is widely deployed in commercial products [7]. The issue of throughput degradation with the ACK-based approaches were only recently raised [5], [7], [8]. Kim *et al.* [8] proposed an RTS/CTS-based scheme to differentiate the channel-induced error and collision. It exploits the short length of the RTS frame to reduce the error in differentiation. In contrast to these prior works, COLA does not rely on hardware, power measurement, or any optional features such as RTS/CTS or CCA.

V. Concluding remarks

This paper proposes COLA, an IEEE 802.11 ACK-based link adaptation algorithm. It is light-weight, does not require RTS/CTS exchange or channel estimation, and is free of heuristic parameters that might complicate practical use. It operates on no explicit knowledge of the congestion level on the 802.11 link (as link adaptation should be), but it achieves close to ideal throughput for wide range of SNR values. We take the case of 802.11a in this paper, but COLA can be easily applied to other 802.11 links with minor adaptation. As it neither requires hardware support nor encroaches upon 802.11 standard, it can be easily implemented and deployed.

References

[1] IEEE, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Std 802.11-1999, Aug. 1999.

[2] IEEE 802.11a, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band*. Supplement to IEEE 802.11 Standard, Sept. 1999.

[3] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in proceedings of ACM Mobicom, 2001.

[4] D. Qiao, S. Choi, and K. G. Shin, "Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs," *IEEE Transactions on Mobile Computing*, 1(4), pp. 278–292, Oct.-Dec. 2002.

[5] S. Choi and D. Qiao,"Fast-responsive link adaptation for IEEE 802.11 WLANs," in proceedings of IEEE ICC, 2005.

[6] A. Kamerman and L. Monteban, "WAVELAN II: a high performance wireless LAN for unlicensed band," *Bell Labs Technical Journal*, pp. 118-133, Summer 1997.

[7] S. Choi, K. Park and C. Kim, "On the Performance Characteristics of WLANs: Revisited", in proceedings of the ACM SIGMETRICS, 2005.

[8] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs," to appear in proceedings of IEEE Infocom 2006.

[9] I. Tinnirello, S. Choi, and Y. Kim, "Revisit of RTS/CTS exchange in high-speed 802.11 networks," in proceedings of IEEE WoWMoM, 2005.

[10] 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.

[11] The ns-2 simulator. Available at http://www.isi.edu/nsnam/ns/.

[12] J. Heiskala and J. Terry, *FDM wireless LANs: a theoretical and practical guide*, SAMS Publishing, 2002.

[13] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in proceedings of IEEE Infocom, 2003.

[14] D. Qiao and S. Choi,"Goodput enhancement of IEEE 802.11a wireless LAN via link adaptation,"in proceedings of IEEE ICC, 2001.

[15] J. del P. Pavon and S. Choi, "Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement," in proceedings of IEEE ICC, 2003.

[16] P. Chevillat, J. Jelitto, A. N. Barreto, and H. Truong,"A dynamic link adaptation algorithm for IEEE 802.11a wireless LANs,"in proceedings of IEEE ICC, 2003.