AGILE Rate Control for IEEE 802.11 Networks

Lochan Verma[1], Seongkwan Kim[2], Sunghyun Choi[3], and Sung-Ju Lee[4]

[1] Wireless Lab., DMC Division, Samsung Electronics, Korea
[2] System Design Lab. 1, DMC Division, Samsung Electronics, Korea
[3] School of Electrical Engineering & INMC, Seoul National University, Korea
[4] Multimedia Communications & Networking Lab., HP Labs, Palo Alto, CA 94304
{lochan.verma,seong.kim}@samsung.com, schoi@snu.ac.kr, sjlee@hp.com

Abstract. We present a transmission rate adaptation algorithm called *AGILE* (ACK-Guided Immediate Link rate Estimation) for IEEE 802.11 networks. The key idea of AGILE is that the transmitter adjusts the transmission rate by means of measuring the SNR (Signal-to-Noise Ratio) during any frame reception including the ACK (Acknowledgment) frame, and estimating the corresponding maximum achievable throughput using a *profile*, which is materialized by extensive off-line measurement. AGILE is equipped with an advanced RTS (Request-To-Send)/CTS (Clear-To-Send) activation algorithm, *eRTS filter* that intelligently switches on/off RTS frame transmission to enhance the achievable throughput depending upon the existence of multiple contending (or even hidden) stations. The effectiveness of AGILE is evaluated in our MadWifi-based testbed implementation and we compare its performance with different rate adaptation schemes in various scenarios.

1 Introduction

The transmission rate optimization over the medium quality variation is a well-known algorithmic issue. Ideally, a rate adaptation algorithm should downgrade or upgrade to a suitable rate with wireless channel dynamics in a timely manner. Most of today's algorithms suffer in agility as they monitor the channel quality over predefined time period and/or threshold number of frame transmissions. It is also important that rate adaptation decisions are not influenced by increased wireless medium contention or the presence of a hidden station.

Rate adaptation algorithms can be classified as: (i) *Close Loop*, which requires feedback from the receiver to make a rate selection, and (ii) *Open Loop*, which is not dependent on receiver feedback. Current 802.11 standard does not support any feedback from the receiver to the transmitter, which makes algorithms like RBAR (Receiver-Based AutoRate) [4] and OAR (Opportunistic Auto-Rate) [11] unemployable. On the other hand, open-loop algorithms depend on local information available at the transmitter to decide rate. The criterion for rate selection in these algorithms can be (i) either SNR (Signal-to-Noise Ratio) based or leveraging frame loss ratio measurement and (ii) using either of them with predefined threshold tables. ARF (Automatic Rate Fallback) [7], ONOE [1], AARF (Adaptive Auto Rate Fallback) [9], AMRR (Adaptive Multi Rate Retry) [9], SAMPLE (SampleRate) [2], CARA (Collision-Aware Rate Adaptation) [8], and RRAA

(Robust Rate Adaptation Algorithm) [14] algorithms are based on frame loss ratio estimation, which is typically performed by keeping track of unacknowledged transmissions. Frame loss estimation-based rate selection policies often suffer from poor responsiveness to varying channel quality. QCS [10], JC [3], CHARM (Channel-Aware Rate selection algorithM) [6], and SGRA (SNR-Guided Rate Adaptation) [15] algorithms have SNR measurement-based rate selection policy supported by a threshold table.

We present a transmission rate adaptation algorithm called *AGILE* (ACK-Guided Immediate Link rate Estimation) for IEEE 802.11 networks. The strength of AGILE lies in precise and timely rate adaptation, which is driven by the SNR measurement of any frame reception including ACK frames, and a predetermined look-up table. This table named as *profile* represents the relation between delivery ratio and SNR for a given transmission rate, and has been materialized by extensive field-testing experiments. AGILE also utilizes an advanced collision resolution technique, *eRTS filter* that enhances achievable throughput when there exist multiple contending (or even hidden) stations.

The rest of the paper is organized as follows. In Section 2, the design rules and implementation procedure of AGILE are described. In Section 3, the solution for collision awareness is presented. Section 4 presents the comparative performance evaluation based on testbed measurements, and the paper concludes with Section 5.

2 AGILE

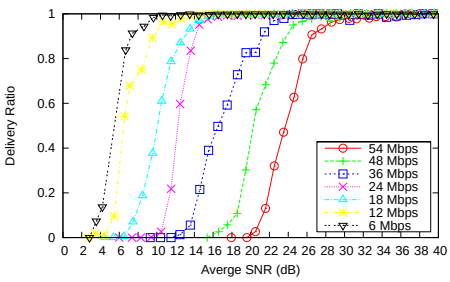


Fig. 1. Profile using 802.11a broadcast frames with 1 dB bucket along x-axis for different transmission rates.

Design Overview. In order to promptly respond to the channel dynamics, AGILE utilizes SNR of any frame received. Initially, the first frame on a link is sent at the lowest transmission rate and after receiving an ACK or any other frame, the most suitable rate is selected from the profile look-up. Our idea is that the SNR of a frame represents the quality of the wireless medium at that time. This is in line with the reciprocity theorem that states *the characteristics of the wireless medium remain unchanged if the roles of transmitter and receiver are interchanged at a particular instant* [12].

Profile shown in Fig. 1 has been generated through extensive off-line experiment using two NICs (Network Interface Cards) for different 802.11a rates. Note that profile is dependent on vendor and chipset. We use Atheros/MadWifi NIC/driver pair as MadWifi is an open source driver for Atheros chipsets. We develop a profile using 1000-byte broadcast frames and modify the driver to register the attributes related to each transmission/reception event of frames, which

generates an enormous database listing the sequence number, time-stamp, and RSSI (Received Signal Strength Indication).[5] Detailed information on profile generation is presented in [13].

An integral part of a rate control algorithm is the retransmission policy. AGILE does not follow the conventional retransmission policy adopted by SAMPLE, ONOE, and AMRR. Our retransmission policy is based on extensive experiment testing that generated the highest throughput. Table 1 shows the rate for each retransmission.

Table 1. AGILE rate adjustment for retransmissions with 802.11a

Attempt	Initial	Retry1	Retry2	Retry3	Retry4	Retry5	Retry6	Retry7
Rate	Current	Current	18 (6)*	18 (6)*	6	6	6	6

* (6) denotes that the lowest transmission rate is used when 'Current' is less than or equal to 18 Mbps.

AGILE Algorithm. Assume an L-byte frame is to be transmitted using a transmission rate index i out of the 802.11a data rate set. The probability of a successful frame transmission is given by $P_s^i(L) = (1 - P_{e\_data}^i(L)) \cdot (1 - P_{e\_ack}^i(14))$, where $P_{e\_data}^i(L)$ and $P_{e\_ack}^i(14)$ represent the error probabilities for an L-byte data frame and the 14-byte ACK frame, respectively. $P_s^i(L)$ is determined through a profile look-up according to the SNR of a received frame.

Let $s^i(k)$ be the probability that a data frame is successfully transmitted at rate index i after k transmission attempts. $s^i(k) = (1 - P_s^i(L))^{k-1} \cdot P_s^i(L)$. The transmission time for a frame with rate index i can be determined by

$$T_{frame}^i(L) = \sum_{k=1}^{Max\_Try} s^i(k) \cdot \sum_{j=0}^{k-1} \left[DIFS + \overline{T}_{Backoff}(j) + T_{data}^i(L) + SIFS + T_{ack}^i \right], \tag{1}$$

where Max_Try is the maximum frame transmission attempts, $DIFS$, $SIFS$, $CWmin$, $CWmax$, and $SlotTime$ hold the same meaning as in IEEE 802.11 standard [5], and $\overline{T}_{Backoff}(j)$ is the average backoff interval in μsec, after j consecutive unsuccessful attempts:

$$\overline{T}_{Backoff}(j) = \begin{cases} \frac{(CWmin+1)^j - 1}{2} \cdot SlotTime & 0 \le j < 6, \\ \frac{CWmax}{2} \cdot SlotTime & j \ge 6, \end{cases} \tag{2}$$

$$T_{data}^i(L) = t_{PLCPoverhead} + \left(\frac{L \cdot 8}{Tx\_rate(i)} \right), \tag{3}$$

$$T_{ack}^i = t_{PLCPoverhead} + \left(\frac{14 \cdot 8}{Tx\_rate(i^*)} \right), \tag{4}$$

where $t_{PLCPoverhead} = t_{PLCPpreamble} + t_{PLCPheader} = 20$ μsec. The transmission rate index i^* used for an ACK transmission is determined by the highest

[5] In MadWifi, RSSI is reported as gain over noise in dB. This is equivalent to SNR. Thus, we refer to RSSI as SNR in our work.

rate in the basic rate set that is smaller than or equal to the corresponding data transmission rate. The typical basic rate set in the 802.11a includes 6, 12, and 24 Mbps, and in this case, i^* is determined as 1, 1, 3, 3, 5, 5, 5, 5 for $i = 1, 2, 3, 4, 5, 6, 7, 8$, respectively.

A station, say $\ddot{A}$, deciding the transmission rate for a frame runs the following algorithm:

ALGORITHM 1. *AGILE rate control algorithm*
1. $\forall$ *i,* $\ddot{A}$ *looks up the profile to calculate* $P_s^i(L)$ *using SNR of a received frame.*
2. $\forall$ *i,* $\ddot{A}$ *executes Eq. (1) to calculate* $T_{frame}^i(L)$ *using* $P_s^i(L)$ *in Step 1.*
3. $\ddot{A}$ *selects rate index i with the minimum* $T_{frame}^i(L)$ *as the rate index.*

3 Collision Awareness

High wireless medium contention or the presence of hidden stations results in many transmission failures due to frame collisions; triggering the algorithms based on frame loss estimation to unnecessarily downgrade the transmission rate. The usage of RTS/CTS is an effective solution to tackle the problem of increased frame collisions. However, it costs precious bandwidth consumption.

AGILE and other SNR-based rate control algorithms have intrinsic collision awareness since rate adaptation decisions are driven by SNR measurements and not by the currently-experienced frame loss levels. Nevertheless, they do require intelligence to protect transmissions from suffering collisions to further boost the performance. In the absence of such a protection mechanism, the achievable throughput of a station can be reduced due to increased retransmissions.

eRTS filter. Inspired by A-RTS (Adaptive RTS) filter proposed in [14], we develop eRTS (enhanced RTS) filter to protect the transmissions from collisions. eRTS design makes it implementation friendly into the current open source drivers. The key parameter in A-RTS is *RTSWnd*, which specifies the window size, in terms of the number of frames, within which all data frames are sent with a preceding RTS frame. Another parameter *RTSCounter* depends upon RTSWnd and represents the actual number of data frames sent with a preceding RTS frame. Initially, RTSWnd and RTSCounter are set as zero, meaning that the RTS/CTS exchange is disabled. A data frame is preceded by an RTS frame as long as RTSCounter is larger than zero. RTSWnd is incremented by one if the last data frame transmission was unsuccessful and RTS was not used, and RTSCounter is set as RTSWnd. When the last data frame transmission failed with a preceding RTS or succeeded without RTS, RTSWnd is halved as the frame did not experience collision, and RTSCounter is set as RTSWnd. RTSWnd is kept unchanged if the last data frame transmission succeeded with a preceding RTS while RTSCounter is decremented by one in this case.

However, A-RTS filter requires per-frame RTS on/off, which leverages the freedom of precise control over each (re)transmission attempt of a frame. MadWifi does not provide control over frames to such a granularity and RTS/CTS usage can only be enabled/disabled for all (re)transmission attempts of a frame.

With this limitation, we make the changes and enhancement to the A-RTS filter to be implementable in the driver.

A data frame which requires greater than or equal to *RThresh* (Retry Threshold) retries for a successful transmission triggers RTSWnd control. We also modify the A-RTS filter to perform a linear decrease of RTSWnd if RTSWnd exceeds the *LThresh* (Linear Threshold); otherwise, it follows multiplicative decrease. RThresh and LThresh values are experimentally determined and fixed as 1 and 3, respectively. The A-RTS filter incorporating the above changes is called *eRTS* filter. Algorithm 2 depicts its complete operation. A station, say *Ä*, which has a data frame to transmit runs the following algorithm:

ALGORITHM 2. *eRTS filter*
1. Ä increases retry_count for each frame retransmission.
2. If RTS is disabled and retry_count ≥ RThresh then Ä increments RTSWnd by 1, and sets RTSCounter equal to RTSWnd.
3. If step 2 conditions are not satisfied, Ä checks if RTS is enabled XOR retry_count ≡ 0, if it is true step 4 is executed otherwise it jumps to step 5.
4. If RTSWnd > LThresh, decrement RTSWnd by 1, otherwise reduce RTSWnd by a half. Set RTSCounter equal to RTSWnd.
5. Finally Ä checks if RTSCounter > 0, and if true, Ä enables RTS transmission to precede the current data frame and decrements RTSCounter by 1. Otherwise RTS transmission is disabled for the current data frame.

4 Testbed Results

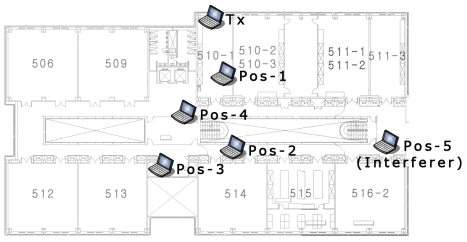


Fig. 2. Testbed.

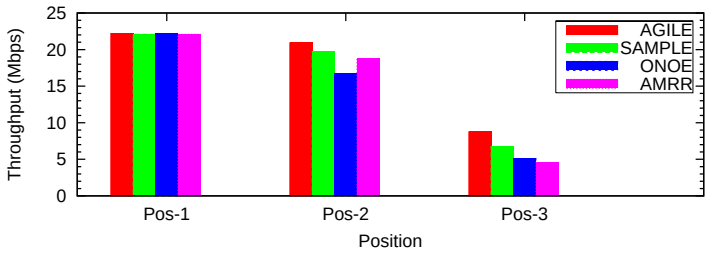
Fig. 2 shows the floor plan of our indoor Linux based testbed. Each laptop is equipped with a Cisco aironet 350 802.11a/b/g PCMCIA card. All experiments are performed using the 802.11a at channel 157 with the center frequency of 5.785 GHz. Using Iperf traffic generator tool the transmitter generates UDP packets of size 1003 bytes as fast as it can so that it always has a packet to transmit. For Sections 4.1 and 4.2, we disable the usage of eRTS filter to show AGILE's throughput performance gain over other compared algorithms. Impact of enabling eRTS algorithm is presented in Section 4.3.

4.1 Experiments with Stationary Nodes

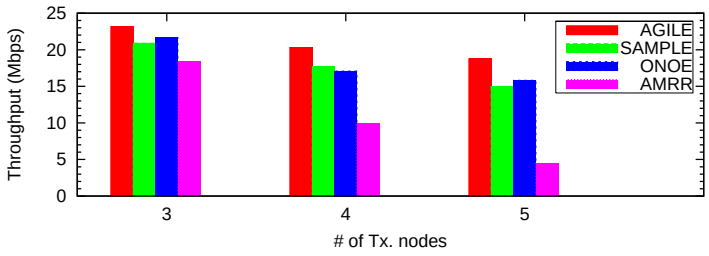
As shown in Fig. 3(a), the throughput gain of AGILE over other algorithms becomes larger as the path loss increases. At Pos-1, the transmitter and receiver pair is fairly close and all the algorithms mostly use the highest transmission rate. AGILE has throughput gain of 20.2% over ONOE at Pos-2 and 48.45%

over AMRR at Pos-3. ONOE, AMRR, and SAMPLE perform rate adaptation decisions either after predefined time period or threshold number of frame transmission. This criterion makes them less responsive to channel variations. The experiment to study the effect of increasing wireless medium contention is performed with the receiver at Pos-1. The transmitters are placed next to each other at Tx in Fig. 2. As seen in Fig. 3(b), AGILE achieves a throughput gain of 20.8%, 50.4%, and 76% over the worst performer (AMRR) with 3, 4, and 5 contending nodes, respectively. With increasing contention levels, each station experiences an increase in the frame losses, thus triggering rate decrement decisions in SAMPLE, ONOE, and AMRR.

Through the experiment to verify the responsiveness to varying link quality, we explore the agility of different rate adaptation algorithms. In order to create variation in channel quality, we block the transmitter NIC card every 5 seconds for 5 seconds with a thick book forming a canopy around the card. This reduces the SNR of the transmitted and the received frames. The receiver is located at Pos-2 in this experiment. AGILE provides 20.5%, 9.6%, and 20.7% throughput gain over SAMPLE, ONOE, and AMRR as seen in Fig. 4.



(a)



(b)

Fig. 3. Performance comparison of AGILE with other algorithms: (a) Throughput with increasing path loss; and (b) Throughput with increasing contention level.

4.2 Transmitter Mobility

We perform this experiment by placing the receiver at Pos-2 and transmitter moves back and forth between Tx and Pos-1 at roughly 1 m/sec, which is normal

human walking speed. As shown in Fig. 4, AMRR has a throughput gain of 9.3% over SAMPLE, which is consistent with the results reported in [14].

AGILE has throughput gain of 13.6%, 14%, and 4.7% over SAMPLE, ONOE, and AMRR, respectively. AGILE shows the performance superior to other algorithms by quickly reacting to the channel dynamics and selecting accurate rates.

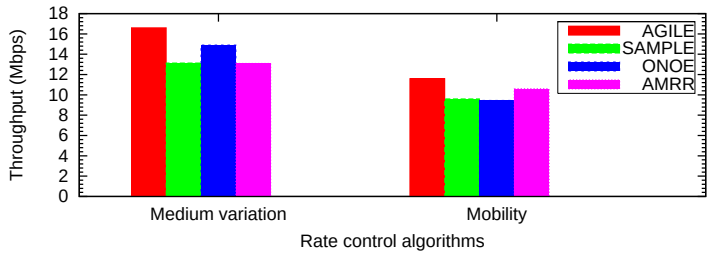


Fig. 4. Throughput performance comparison of AGILE with other algorithms considering the varying medium quality scenario and the transmitter mobility scenario.

4.3 Effect of Hidden Stations

To generate a hidden station topology, we place a receiver at Pos-4 while two transmitters are located at Tx and Pos-5. The transmitter at Pos-5 is named as the hidden interferer, and generates broadcast UDP packets of 1003 bytes at various traffic generation rates to expose the transmissions from Tx to Pos-4 to mild, moderate, and intense interference from the hidden interferer. Transmitter at Tx follows an identical setting as that in other previously mentioned experiments.

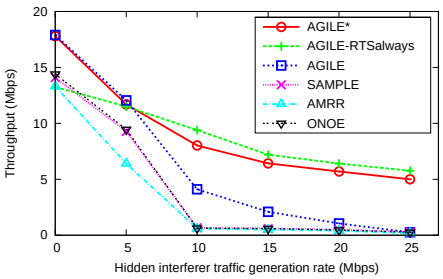


Fig. 5. Throughput performance under the effect of hidden station.

Fig. 5 shows the maximum achievable throughput at Tx with varying interference from the hidden interferer. To quantify the performance gains achieved by enabling eRTS algorithm we experiment both with AGILE, with eRTS filter disabled and AGILE*, which has eRTS filter enabled. SAMPLE, ONOE, and AMRR perform poorly with significant throughput degradation as the hidden interference intensity shifts from mild to intense. With increasing interference levels the achievable throughput with AGILE also degrades significantly. AGILE-RTSalways includes an RTS/CTS handshake between the Tx and Pos-4 before each data frame transmission to reserve the medium. For hidden interference between 0–5 Mbps always using RTS/CTS acts as an overhead degrading the throughput. However, AGILE-RTSalways outperforms all other schemes for 5–25 Mbps rate of hidden interference.

AGILE* represents the most suitable algorithm across mild, moderate, and intense interference conditions. Overall, AGILE is a very responsive rate adaptation algorithm and the supplementary eRTS filter further boosts its performance.

5 Conclusion

In this paper, we propose an SNR-based rate adaptation algorithm called AGILE and provide collision awareness with eRTS filter. We have implemented both AGILE and eRTS into MadWifi driver and performed extensive experiments to compare the performance with existing algorithms. In all scenarios, AGILE provides the best performance. As a future work, we plan to introduce on-line compensation into the profile, which is obtained through off-line experimentation. This will enable each station to have a profile for each individual wireless link enabling AGILE to perform even better.

References

1. Onoe Rate Control Algorithm. `http://madwifi.org/browser/madwifi/trunk/ath_rate/onoe/`.
2. J. Bicket. Bit-rate selection in wireless networks. Master's thesis, MIT, 2005.
3. J. del Prado Pavon and S. Choi. Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement. In *Proc. IEEE ICC'03*, pages 1108–1113, Anchorage, AK, USA, May 2003.
4. G. Holland et al. A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks. In *Proc. ACM MobiCom'01*, 2001.
5. IEEE Std. IEEE 802.11-1999, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, Aug. 1999.
6. G. Judd et al. Efficient Channel-aware Rate Adaptation in Dynamic Enviroments. In *Proc. ACM MobiSys'08*, Breckenridge, Colorado, USA, June 2008.
7. A. Kamerman and L. Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, 2(3):118–133, Aug. 1997.
8. J. Kim et al. CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs. In *Proc. IEEE INFOCOM'06*, 2006.
9. M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proc. ACM MSWiM'04*, Venezia, Italy, Oct. 2004.
10. D. Qiao et al. Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs. *IEEE Trans. Mobile Comput.*, 1(4), 2002.
11. B. Sadeghi et al. Opportunistic Media Access for Multirate Ad Hoc Networks. In *Proc. ACM MobiCom'02*, 2002.
12. C. T. Tai. Complementary reciprocity theorems in electromagnetic theory. *IEEE Trans. Antennas Propagat.*, 40(6):675–681, 1992.
13. L. Verma, S. Kim, S. Choi, and S.-J. Lee. Reliable, Low Overhead Link Quality Estimation for 802.11 Wireless Mesh Networks. In *Proc. IEEE WiMesh'08*, San Francisco, Calafornia, USA, June 2008.
14. S. Wong et al. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proc. ACM MobiCom'06*, 2006.
15. J. Zhang et al. A Practical SNR-Guided Rate Adaptation. In *Proc. IEEE INFOCOM'08*, Las Vegas, Nevada, USA, Apr. 2008.