IEEE 802.11 Rate Control Algorithms: Experimentation and Performance Evaluation in Infrastructure Mode

Sourav Pal, Sumantra R. Kundu, Kalyan Basu and Sajal K. Das

Center for Research in Wireless, Mobility and Networking (CReWMaN)
The University of Texas at Arlington, TX 76019-0015
Email:{spal,kundu,basu,das}@cse.uta.edu

Abstract. In this paper we evaluate the performance of practical Rate Control Algorithms (RCAs) operating at the Media Access Control (MAC) layer of IEEE 802.11 networks built on Atheros chipsets. The aim of this study is two-fold: (1) to explore the performance of the RCAs with varying wireless channel conditions at the link layer and (2) to observe the implicit effect the RCAs have on application level throughput and response times. By measuring the performance of heterogeneous traffic over specific rate adaptation algorithm, we are able to conclude that in addition to the state of the wireless channel, the performance of the RCA is closely tied to the nature and type of data stream currently being transmitted over the physical interface. For delay sensitive traffic like VoIP applications, aggressively trying to increase the bit-rate during fluctuating channel conditions might cause unacceptable packet loss and ultimately break session connectivity. Additionally, our study indicates that deriving channel information from raw RSSI values and packet probes might not reveal the complete picture about the dynamics of the underlying wireless channel.

1 Introduction

The IEEE 802.11 [4] physical layer (PHY) supports multi-rate transmission capabilities by dynamically choosing the most appropriate modulation technique for the received signal strength. This in turn, empowers the Wireless Network Interface Card (WNIC) to adapt the transmission rate to the conditions of the underlying radio channel. While the IEEE standard defines the specifications for 802.11 MAC protocol and RF-oriented PHY parameters [4], it does not define any particular instance of RCA and is open to the device manufacturer to improvise.

Currently there exists two different approaches for implementing the RCA in WNIC. They can be broadly categorized as: (i) the *software* approach and (ii) the *hardware* approach. In the former, the main functionalities of the RCA are implemented as a software driver in the operating system where as in the later, the algorithms are part of the wireless chipset. In this paper, we are interested in investigating the performance of *software RCAs* that interact with the PHY

layer of the WNIC. We use the 802.11 chipset (AR5212) from Atheros Communications [12] and the Multimode Atheros Driver (Madwifi) [1] for performance evaluation of the three available 802.11 multi-rate control algorithms: Onoe [1], Adaptive Multi Rate Retry (AMRR) [2] and SampleRate [3] bit-rate selection algorithm.

In particular, we are interested in investigating the following: (i) *How do the RCAs perform with variations of the Received Signal Strength Indicator (RSSI) values?* (ii) *How do the performance of the RCAs impact application level throughput for different traffic classes (heterogenous, streaming media, interactive and background)?*

The main focus and contributions of this paper are evaluation of the RCAs and development of performance metrics that can be used as a framework for comparing all of them. The rest of the paper is organized as follows. In Section 2 we briefly review the Madwifi driver and the available RCAs. This is followed by Section 3 where we describe the experimental testbed and the qualitative metric used for analyzing the RCAs. We validate and evaluate the effectiveness of the RCAs on application level traffic in Section 4. Finally, in Section 5 we conclude summarizing our observations.

2 Multiband Atheros Driver for WiFi (Madwifi)

Madwifi [1] is an Open Source driver for wireless chipsets from Atheros Communications [12]. It is a multi-core driver module that comprises of (i) a PCI hardware module (`ath_pci.ko`) for interfacing with PCI I/O bus (ii) Atheros chipset specific module (`ath_hal.ko`) for acting as the glue between the hardware registers and the driver software and (iii) the device independent module (`wlan.ko`) implementing the IEEE 802.11 state machine. Each of the RCAs are available as separate kernel modules inside the main driver. Information about the current rate and packet retransmission statistics are communicated by the hardware to the kernel registered RCA. The RCA module collects these parameters and develops its rate adaptation strategy. In Section 2.1, we have highlighted the principal parameters and default values for each of the RCAs as present in the Madwifi driver at the time of writing. It should be noted that optimization and performance tuning of the RCA parameters is possible but is beyond the scope of this work.

2.1 Rate Control Algorithms: Onoe, AMRR and SampleRate

Onoe [1] is a *credit* based RCA where the value of the credit is determined by the frequency of successful, erroneous and retransmissions accumulated during a fixed invocation period of 1000 ms. If less than 10% of the packets need to be retransmitted at a particular rate, Onoe keeps increasing its credit point till the threshold value of 10 is reached. At this point, the current transmission rate is increased to the next available higher rate and the process repeated with credit score of zero. Similar logic holds for deducting the credit score and moving to a

lower bit-rate for failed packet transmission/retransmission attempts. However, once a bit-rate has been marked as failure in the previous attempt, Onoe will not attempt to select that bit-rate until 10 seconds have elapsed since the last attempt. Due to the manner in which it operates, Onoe is *conservative* in rate selection and is less sensitive to *individual* packet failure. Further details of the algorithm are available in [1].

AMRR [2] uses *Binary Exponential Backoff (BEB)* technique to adapt the length (threshold) of the sampling period used to change the values of bit-rate and transmission count parameters. It uses probe packets and depending on their transmission status adaptively changes the threshold value. The adaptation mechanism ensures fewer failed transmission/retransmission and higher throughput by not switching to a higher rate as specified by the backoff mechanism. In addition to this, the AMRR employs heuristics to capture the short-term variations of the channel by judiciously setting the rate and transmission count parameters. For further details refer [2].

SampleRate [3] decides on the transmission bit-rate based on the past history of performance; it keeps a record of the number of successive failures, the number of successful transmits and the total transmission time along with the destination for that bit-rate. Stale samples are removed based on a EWMA windowing mechanism. If in the sampling process, no successful acknowledgment is received or the number of packets sent is multiple of 10 on a specific link, it transmits the packet with the highest rate which has not failed 4 successive times. Other than that it transmits packets at the rate which has the lowest average transmission time.

3 Experimentation and Performance Analysis

The experimental testbed used to measure the performance of the RCAs is shown in Figure 1. It comprises of two subnets: (i) a Gigabit Ethernet wired subnet (a.b.c.114) and (ii) an IEEE 802.11g wireless subnet (192.168.14.x). The content delivery server (a.b.c.114) is connected to the workstation (a.b.c.228) through a Layer 2 managed Gigabit Ethernet switch (Netgear GSM 7224). The AP is a Linux based workstation with 32bit PCI WNIC (Netgear WAG311). Using the tools `iwpriv` and `iwconfig`, we configured it to operate as a Master in IEEE 802.11g mode (2450 MHz radio spectrum). The AP creates a subnet (192.168.14.x) for the wireless clients and uses address translation (using `iptables`) for forwarding packets from the wireless network to the globally accessible wired domain. IP addresses are leased dynamically to the wireless clients using `dhcpd`.
A Sharp Actius (PC-MP30) laptop with 512MB RAM and an externally supplied 32bit cardbus from Netgear (WAG511v2) is used as the mobile client. We make sure that during the entire experiment the onboard wireless chipset is turned off. All of the platforms (including the mobile laptop) run the same configuration of SuSE Linux with unpatched vanilla 2.6.13.3 kernel.

The Madwifi driver used in our experiments has the following version of

RCAs: Onoe (v1.3.20), SampleRate (v1.2) and AMRR (v0.1). The corresponding version of `ath_hal.ko`, `wlan.ko` and `ath_pci.ko` modules are 0.8.2.0, 0.8.2.0 and 0.7.0.0 respectively. Since there are no additional clients in the wireless subnet, we disable the RTS/CTS (Request to Send/Clear To Send) feature of the IEEE 802.11 standard.

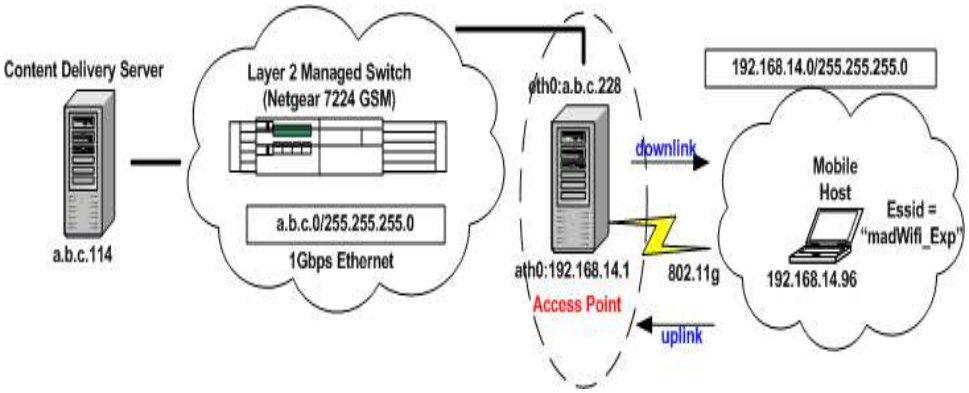

Fig. 1. IEEE 802.11 Experimental Testbed in Infrastructure Mode.

3.1 Experimental Setup

In order to investigate and compare the performance of the different RCAs to changing wireless conditions, we need to create an environment with identical variations of RSSI. However, broadly speaking this is a difficult task since the values of RSSI are influenced by several external factors beyond our control (interference, power leakage, fading, multipath propagation). Commonly used procedures like walking to and fro along a definite path or measuring the performance from a certain distance of the AP does not guarantee controlled wireless conditions. In order to circumvent such difficulties, we resorted to generating a closely monitored and controlled step function of RSSI variations that was almost identical for all the RCAs by taking advantage of the shielding influence of a microwave operating at 2450 MHz. Such an approach has also been used in [5][11].

For each of the experiments described below, we *uploaded* a 8 MB file from the laptop to the content delivery server and collected the RSSI values, corresponding transmission rates and timestamps by inserting probes inside the `ath_tx_start()` function in `ath_pci.ko` module. We used the Linux kernel function `do_gettimeofday()` to obtain microsecond granularity in our measurement data.

As the laptop is placed inside the microwave and the door of the oven closed,

the shielding materials of the microwave effectively block almost all radio waves in the 2450 MHz spectrum; causing the RSSI to vary as a step function. We kept the duration of the step length at approximately 5 seconds during each run of the experiment. This caused the RSSI to drop below 20dB. Subsequently, we left the door of the microwave open for another 1 second in order to generate an impulse function. In this case, the RSSI rose sharply above 30dB. Thereafter, we took the laptop out and collected the probe data *after* the entire 8 MB data file has successfully uploaded. We repeated the entire process till we observed minimal variance between identical datasets. This unique experimental procedure ensured that we are able uniformly expose all the RCAs to two step variations of RSSI values.

Low Link Performance Analysis In Figures [2]-[7] we show the manner in which the RCAs react to variations of RSSI values. As expected, when the RSSI falls, the corresponding transmission rate (governed by respective RCA), also decreases. However, all the RCAs react differently to *short term link fluctuations* and *low link conditions*. To have a better visualization of such transient dynamics, we have expanded certain regions (marked by ellipsis in Figures [2][4][6]) for each of the RCAs and presented in Figures [3][5][7] respectively.

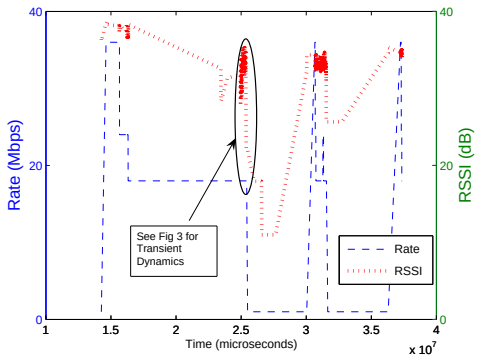

Fig. 2. Transmission Rate Adaptation and RSSI Variation for Onoe Rate Control Algorithm.

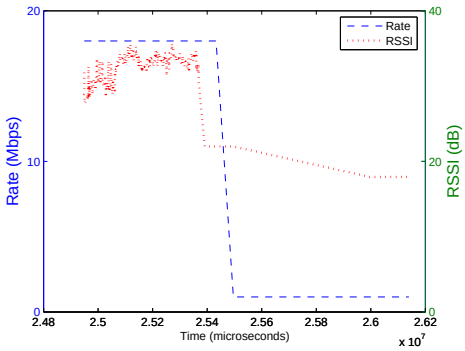

Fig. 3. Transient Dynamics for Onoe Rate Control Algorithm.

In Figures [2] and [3], we see that the credit based Onoe algorithm reacts conservatively to RSSI changes. It changes in *discrete steps* and does not closely follow changes to RSSI variation. The algorithm increases the transmission data rate only when the current bit-rate has at least 10 credits; this makes it practically insensitive to small scale variations of RSSI values (transient dynamics). Such behavior is also evident from Figure [3] where the RSSI varies a lot (when the microwave door is being closed) but the bit-rate remains fixed at 18 Mbps.

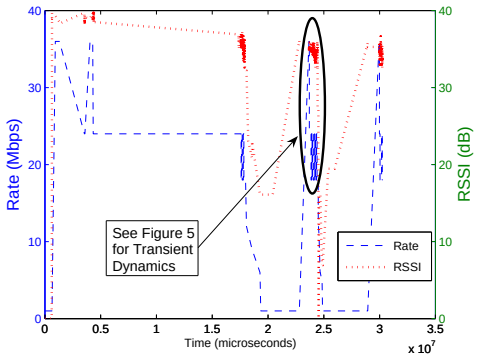

Fig. 4. Transmission Rate Adaptation and RSSI Variation for AMRR Rate Control Algorithm.

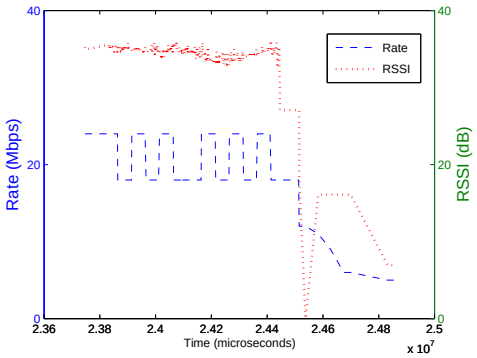

Fig. 5. Transient Dynamics for AMRR Rate Control Algorithm.

In Figures [4] and [5], we illustrate the performance of both the long term and short term response of the AMRR algorithm. Visual inspection from the transient dynamics confirm that AMRR is most sensitive to RSSI changes among all the three RCAs. This is explained by the fact that the AMRR algorithm resets all the values of retransmit parameters to one. Thus, for a *single retransmission failure*, the transmission rate is changed; consequently, the algorithm is able to track small changes of RSSI value. It should be noted that unlike Onoe which falls sharply after detecting that RSSI has changed significantly, the data rate in AMRR closely follows the RSSI trend and changes *smoothly* with a certain slope as it tries to gradually adapt the threshold backoff window. Similar to AMRR but on a more aggressive scale, SampleRate tries to match the transmission rate to the current wireless condition by concurrently sending probes at the next higher data rate. This is captured in Figures [6] and [7]. Based on the statistics of successful transmission, average throughput, round trip time, SampleRate decides on the value of the next transmission rate.

4 Application Layer Performance and Rate Control Algorithm

In this Section, we conduct experiments using four different traffic classes for exploring the impact of the RCAs on application level throughput. Application layer performance is what ultimately affects the session quality of the mobile user. Similar to the ones proposed in universal mobile telecommunications service (UMTS) [14], we use the following four traffic classes: (i) *Heterogeneous* (ii) *Streaming* (iii) *Interactive* and (iv) *Background* traffic. Examples of each traffic classes are *VoIP*, *video streaming*, *web browsing* and *file download* respectively. The corresponding applications used in our experiments for each traffic class

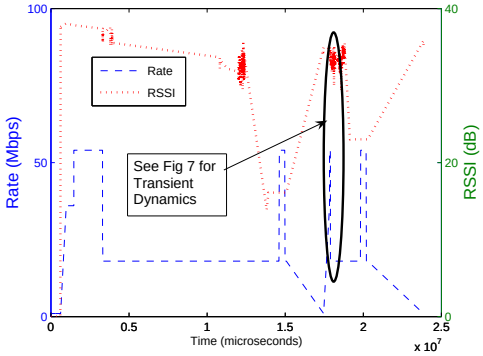

Fig. 6. Transmission Rate Adaptation and RSSI Variation for SampleRate Rate Control Algorithm.

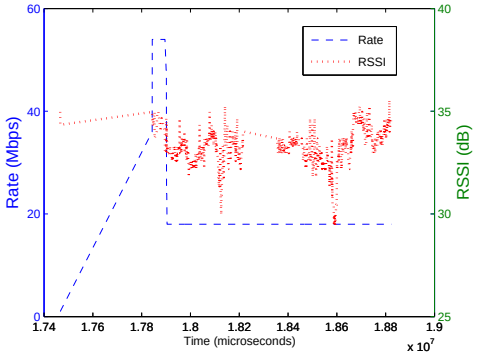

Fig. 7. Transient Dynamics for SampleRate Rate Control Algorithm.

are *Skype* [6], *Kaffeine* [7], *firefox* [8] and *sftp* [9] respectively. In each of the experiments, we vary the radio range from 12 feet to 80 feet from the AP (by walking along the corridor of our lab and carefully retracing the path along a predefined line), *after* the application software has been fired on the laptop. This ensures that noticeable variations of RSSI (from deep fade to strong signal) takes place. The packet level statistics collected during each run of the experiment provide valuable insights about the impact of the RCAs on application level performance; specifically with respect to average throughput, variation of throughput with RSSI and the jitter suffered by the applications. However due to lack of space, we present only the average throughput for each traffic class and also the inter-packet arrival for VoIP application for the three different RCAs. The rest of the information and detailed data sets for each experiment can be found at our project website [13].

In addition to the traffic characteristics observed at the client, we also measured the packet level statistics (by inserting probes in the Madwifi driver) at the wireless AP. Since our focus is on performance measurement of RCAs, we do not present end-to-end throughput but the throughput which is reflected at the AP and at the wireless client (a.k.a. laptop) where the RCAs are active. As a measurement tool, we used `Ethereal` [10] (v0.10.12) for capturing the packet level statistics at both the AP and the laptop. Since the traffic used in our experiments involved TCP flows, we used `tcptrace` (v6.6.7) to compute the average throughput statistics from the Ethereal dump files. We also modified `iptraf` (v2.7.0) in order to generate the variations of application throughput sampled at an average interval of 1 second. For VoIP experiments (based on `Skype`), we wrote our own analysis tool using the `libpcap` packet library for extracting the necessary statistics. The software is available at our website [13].

In Table 1 we provide data for the average throughput (in Kbps) observed for each application using all the three RCAs. Such statistics provide interest-

Table 1. Average Throughput (**in Kbps**) for Heterogeneous Traffic for different Rate Control Algorithms

Rate Control Algorithms	VoIP		Streaming	Interactive		Elastic
	Up	Down		Up	Down	
Onoe	73.89	69.076	936.48	20.05	91.37	1946.91
AMRR	36.47	42.24	1243.95	26.11	109.99	3295.87
SampleRate	50.49	66.92	1132.17	26.07	97.79	2617.32

ing insights regarding the performance of the RCAs. *First*, there is significant differences between upstream and downstream throughput even in the absence of any other wireless clients. *Second*, when it comes to non-real time traffic, AMRR seems to perform more effectively than its counterparts. The performance of SampleRate is very much comparable to AMRR when it comes to average throughput (barring VoIP applications). Though the measured average throughput for AMRR was observed to be higher than SampleRate, it exhibited poor performance with streaming multimedia over HTTP. Even at about distances exceeding 70 feet from the AP, we observed that SampleRate was able to buffer and play the video stream whereas for both AMRR and Onoe the media stream got stalled. Another interesting observation is that for voice and interactive traffic, the average throughput achieved is much less than the theoretical wireless channel of 1Mbps for all the three RCAs.

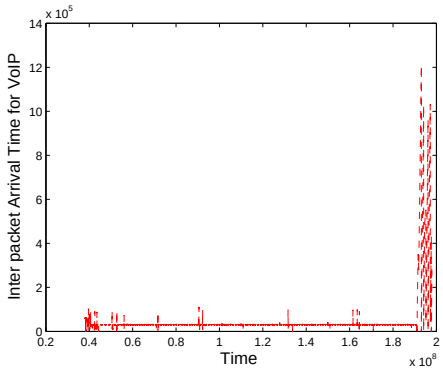

Fig. 8. Inter-Packet Arrival Time for VoIP Application for Onoe. Time in both the axes is in microseconds.

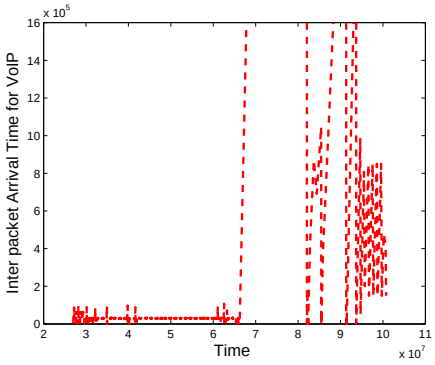

Fig. 9. Inter-Packet Arrival Time for VoIP Application for AMRR. Time in both the axes is in microseconds.

Next we measured the packet inter-packet time for VoIP applications using the three different RCAs. This is shown in Figures [8], [9] and [10]. It is observed

that the performance of Onoe when it comes to VoIP exceeds the performance of both AMRR and SampleRate. This is not surprising since Onoe is not aggressive like the other two and remains stable with variations of wireless conditions. On the other hand, both SampleRate and AMRR closely follow the RSSI and tries to move to a higher data rate as soon as wireless channel improves. This causes the transmission rate to fluctuate a lot and results in significant packet loss due to high channel bit error rate (BER). Thus, aggressively trying to increase the bit-rate during fluctuating channel conditions might not bode well for UDP based applications.

In case of AMRR, each time the laptop was taken from the AP to a distance greater than 60 feet, the VoIP call got dropped. This is very much evident from the delay shown in Figure [9]. For all the RCAs, we observed that the the average inter-packet delay exceeded the desired upper bound value of 150 ms required for VoIP applications. This is shown in Figures [8][9][10]. Consequently, we observed unacceptable and poor voice quality in the form of audible echo at the wireless client.

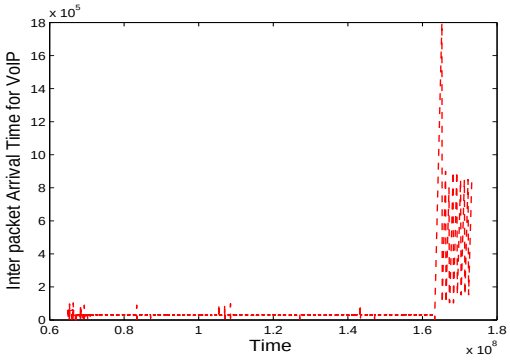

Fig. 10. Inter-Packet Arrival Time for VoIP Application for SampleRate. Time in both the axes is in microseconds.

5 Conclusions

In this paper, we have compared in a systematic manner the performance of the three RCAs available for Atheros based wireless chipsets using the Madwifi driver. We have investigated how each of them perform at the link layer by providing a controlled and closely monitored step function of RSSI variations. We have also exposed the effect of RCA on application layer traffic by conducting experiments involving four classes of heterogeneous traffic (voice, streaming, interactive, background) in controlled network subnets. The average application

level throughput and packet inter-arrival time observed with different RCAs has been reported in this study. Based on our observations, we conclude the following:

- The RCAs need to be aware of the *state* of the wireless channel. Deriving channel information from raw RSSI values and packet probes might not reveal the complete picture about the wireless channel dynamics.
- There is a possibility of improving the performance of the RCAs for low link conditions. The bit-rate adaptation technique might also depend on the stream type being transmitted over the wireless channel.

As part of our future work, we plan to optimize and performance tune the rate adaptation parameters for each of the RCAs and propose a new bit-rate selection algorithm that takes the above factors into consideration.

Acknowledgment: The material of this work is supported by NSF ITR grant IIS-0326505. We would also like to thank the anonymous reviewers for their helpful comments and suggestions which improved the quality of this study.

References

1. Madwifi Project, *"http://madwifi.sourceforge.net/"*.
2. M. Lacage, M. Hossein and T. Turletti, *"IEEE 802.11 Rate Adaptation: A Practical Approach"*, *IEEE MSWiM*, October 2004.
3. J. C. Bicket, *"Bit-rate Selection in Wireless Networks"*, *M.S Thesis, MIT*, February 2005.
4. IEEE Standard 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.
5. I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H Sips, *Automatic IEEE 802.11 rate control for streaming applications*, *Wireless Communications and Mobile Computing*, Vol 5, pp. 412-437, 2005.
6. Skype, *"http://www.skype.com"*.
7. Kaffeine Media Player, *"http://www.kaffeine.sourceforge.net"*.
8. Firefox Web Browser, *"http://www.mozilla.org"*.
9. Secure File Transfer Protocol, *"http://www.apps.ietf.org/rfc/rfc913.html"*.
10. Ethereal: A Network Protocol Analyzer, *"http://www.ethereal.com/"*.
11. I. Haratcherev, R. Lagendijk, K. Langendoen and H. Sips, *"Hybrid Rate control for IEEE 802.11"*, *IEEE MobiWac*, 2004.
12. Atheros Communication, *"http://www.atheros.com/"*.
13. Core Networking and Systems Lab (CoNS), *"http://crewman.uta.edu/cons/"*.
14. 3GPP TR 25.858 v5.0.0, "High Speed Downlink Packet Access: Physical Layer Aspects (Release5)," March 2002.
15. tcptrace, *"http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html"*.
16. IPTraf: IP Network Monitoring Software, *"http://iptraf.seul.org/"*.