Throughput Enabled Rate Adaptation in Wireless Networks

Duy Nguyen[†], J.J. Garcia-Luna-Aceves[†]
[†]Computer Engineering Department
University of California, Santa Cruz, CA 95064, USA
Email: duy, jj, cedric @soe.ucsc.edu

Cedric Westphal[†,*]
[*]Huawei Innovation Center
Santa Clara, CA 95050, USA
Email: cedric.westphal@huawei.com

***Abstract*—Designing a rate adaptation algorithm that performs well in diverse settings is challenging due to the complex physical-layer effects of wireless links, including interference, attenuation, and multi-path fading. We present Throughput Enabled Rate Adaptation (TERA), a new approach to rate adaptation that accounts for interference and congestion effects implicitly but is based solely on measurements of the throughput attained at different data rates. TERA is surprisingly simple and compatible with existing 802.11 implementations. Extensive simulations and real-world experiments are used to show that TERA consistently outperforms all prior rate adaptation schemes used to date.**

I. Introduction

The objective for rate adaptation in a wireless network is to assign the largest possible transmission rates to nodes in a way that multiple access interference (MAI) is minimized while the receiving nodes are still able to decode the transmitted packets under the current channel state. Rate adaptation constitutes a key aspect of the functionality of the IEEE 802.11 physical layer (PHY).

Section II provides a survey of prior work on rate adaptation. While many solutions exist addressing the rate adaptation problem, the design of an efficient solution applicable to multiple diverse scenarios has proven to be elusive. This is due in part to the complex nature of a wireless channel and its interaction with the channel contention caused by users as they access the shared resource, plus the fact that network-level steps taken by nodes (e.g., attempting to use alternate routes around congestion hotspots) may induce additional interference by making more nodes relay packets.

Section III presents TERA (Throughput Enabled Rate Adaptation). The design of TERA is based on the simple observation that providing effective rate adaptation in a wireless network is necessarily related to the throughput experienced by nodes at different data rates. Furthermore, end-user applications care about the throughput attained, rather than the transmission rate used or the loss ratios. MAI and natural phenomena associated to radio wave propagation are the key reasons for throughput reduction in wireless networks. Adapting to them is complicated by the unpredictability of interference. A network may be subject to little or a lot of interference, depending on the characteristics of the environment, the network density, and node movement, and environmental mobility. A major concern with MAI is that it increases very rapidly with node density and impacts the network layer, which causes MAI to spread over multiple hops as nodes attempt to route around congestion. Given that MAI depends on the time that each transmission occupies the channel, nodes must attempt to transmit at the highest data rates that render the largest throughput. Because the algorithm used for rate adaptation must operate without any a-priori information on the state of the system, nodes must increase or decrease their rates quickly if they perceive significant degrees of successful or failed transmissions, and must change their rates smoothly otherwise. The algorithm used in TERA for rate adaptation is based on the 802.11 rate index, which makes it compatible with 802.11 networks. The use of throughput as the single parameter for rate adaptation in TERA leads to a solution that is surprisingly simple, robust, and efficient.

Section IV describes its prototype implementation in Linux and its testing over multiple platforms that demonstrate interoperability with a variety of legacy access in a variety of settings. Extensive simulation and real-world experiments demonstrate that TERA outperforms other rate adaptation mechanism across a wide range of realistic scenarios and environments. Section V offers our concluding remarks.

II. Related Work

Rate adaptation schemes can be classified based on whether explicit or implicit feedback to the transmitters is used. Explicit feedback requires the receiver to explicitly communicate the channel condition on the receiver's side back to the sender. Implicit feedback looks at acknowledgment (ACK) packets or other channel information (i.e., received signal strength indicator (RSSI)) to infer the channel conditions on the receiver's side. We use the term rate control and rate adaptation interchangeably.

A. Explicit Feedback Approaches

Explicit feedback approaches can be viewed as receiver-driven rate adaptation, because the receiver dictates the rate that should be used. The receiver obtains its current channel condition and relays this information back to the sender.

Receiver Based Auto-Rate (RBAR) [6] selects the bit rate based on the S/N measurements. Upon processing a request to send (RTS) packet, the receiver calculates the highest bitrate and piggybacks this selected bit rate on the clear to send (CTS)

packet. However, RBAR needs an accurate mapping between S/N values rates for different hardware.

Collision-Aware Rate Adaptation (CARA) [10] combines the RTS/CTS packets for Clear Channel Assessment (CCA) functionality to differentiate frame collisions and frame failures. CARA requires too many control packets.

Effective SNR [5] presents a delivery model by taking RF channel state as input and predicts packet delivery for the links based on the configuration of the Network Interface Controller (NIC). It takes advantage of the channel state information (CSI) either from feedback or estimated from the reverse path and computes its effective SNR by averaging the subcarrier BERs in order to find the corresponding SNR, where BER is a function of the symbol SNR and OFDM modulations. The drawback of using CSI is that SNR needs to be measured instantaneously, and feedback delay may not allow mode adaptation on an instantaneous basis [4]. Because CSI itself is an approximation of the wireless channel, it may need to incorporate other information, such as higher-order statistics of SNR and Packet/Bit Error Rate or both for improving its accuracy and robustness [4].

In addition to incurring overhead by requiring the receiver to relay its channel state information back to the sender, an explicit approach may encounter the possibility of stale feedback due to the dynamic channel conditions during data transmissions. If the channel coherence time is very short, the receiver may not be able to relay accurate information to the sender. In the worst-case scenario, the receiver ends up sending feedback information to the sender continuously, which occupies the channel with feedback packets and prevents the sender from transmitting data.

B. Implicit Feedback Approaches

Implicit feedback approaches can be viewed as a sender-driven rate adaptation, given that the sender adapts its rate by inferring the channel conditions on the receiver side.

The Automatic Rate Fallback (ARF) scheme is one of the earliest rate control algorithms designed for WaveLAN-II [9]. Upon encountering a second missed acknowledgement of data packets, then it falls back to a lower rate. A counter is used to track the number of good and bad acknowledgement packets for upgrading rates accordingly. However, the limitation of ARF is that it was designed for a few rates and does not work well with current IEEE 802.11 implementation.

ONOE is a credit-based rate control algorithms originally developed by Atheros [1]. It extends ARF [9] to current IEEE 802.11. However, its limitation is that the credit-based system tends to be too conservative and often gets "stuck" using lower rates.

The Adaptive Multi Rate Retry (AMRR) scheme [11] introduces a Binary Exponential Back-off and adaptive threshold value depends on the feedback obtained from the number of attempted packets. The limitation of this approach is that binary exponential back-off tends to be too conservative in adapting rates.

The Sample rate control algorithm [8] begins by sending the data at the highest bit rate. Upon encountering four successive failures, the scheme decreases the bitrate until it finds a usable bitrate. At every tenth data packet, the algorithm picks a random bitrate that may do better than the current one. MINSTREL [12], a widely deployed and popular Linux rate control, is an improved version of Sample, which takes into account the exponential weighted moving average statistics for sorting throughput rates. Unfortunately, MINSTREL still spends 10 percent of transmitted frames in trying random rates when its current rate is working perfectly.

Robust Rate Adaptation Algorithm (RRAA) [15] uses short-term loss ratios to opportunistically adapt the rates. Like CARA [10], it employs an RTS filter to prevent collision losses from rate decreases. However, enabling RTS filtering upon encountering failed transmissions might not work as well as simply transmitting the data at lower rates. Besides, this adds an additional control overhead. Due to the nature of air interface, it is complex and difficult to predict the cause of the packet collisions.

Woo and Culler [16] proposed an adaptive rate control that uses loss as collision signal to adjust the transmission rate for sensor networks. The limitation of this scheme is that it assumes that nodes have a notion of descendants or parents as in sensor networks.

MAICA [3] adapts the data rates used for transmission based on packet loss and a credit-based system. MAICA is inspired by the use of additive increase, multiplicative decrease policies AIMD congestion control mechanism and adopts it for the rate adaptation in wireless environment. The only limitation in MAICA is the need to tune several parameters needed for its credit-based system, which reduces its ease of use and deployment.

Rate Index	IEEE 802.11a Data Rates (Mbps)	IEEE 802.11b Data Rates (Mbps)	IEEE 802.11g Data Rates (Mbps)
0	6	1	1
1	9	2	2
2	12	5.5	6
3	18	11	9
4	24	n/a	12
5	36	n/a	18
6	48	n/a	24
7	54	n/a	36
8	n/a	n/a	48
9	n/a	n/a	54

TABLE I: Rate Index and Data Rates Conversion Table [7]

III. THROUGHPUT-ENABLED RATE ADAPTATION (TERA)

Rate adaptation that operates solely on throughput monitoring at the transmitting nodes is desirable, because it would require no configuration of performance parameters, and would incorporate MAI and channel effects implicitly. This is precisely the motivation behind TERA. The design challenge for TERA is that the transmitter must: (a) increase its transmission rate quickly as long as the perceived throughput increases, (b) avoid drastic fluctuations in data rates resulting

from instantaneous measurements of throughput, (c) reduce its transmission rate quickly when the channel conditions are poor, and (d) must operate without any a-priori knowledge of the state of the channel. Algorithm 1 specifies the way in which TERA attains rate adaptation based on throughput measurements taking into account the four constraints just mentioned.

Algorithm 1 TERA Algorithm

```
1: ω = time window
2: idx = the rate index as shown in Table I.
3: check(ω) = check whether time window ω is expiring
4: Multiplicative = successive successful probes
5: Oscillate = rates oscillate in a see-saw fashion state
6: resetTimeInterval = for adjusting the frequency of probing
7: Γ_prev = previous throughput
8: prev_idx = previous rate index
9: while check(ω) do
10:    if Probing then
11:       if Γ < Γ' and idx! = prev_idx then
12:          idx = prev_idx
13:          if Multiplicative then
14:             resetTimeInterval = 100ms
15:          else
16:             resetTimeInterval = 900ms
17:          end if
18:       else if Γ > Γ_prev and idx! = prev_idx then
19:          resetTimeInterval = 100ms
20:       end if
21:       return
22:    end if
23:    if Δ ≥ 1 then
24:       // Multiplicative Increase
25:       if Multiplicative and !Oscillate then
26:          if idx + idx < max_idx then
27:             idx = idx + idx
28:          else
29:             idx = max_idx − 1
30:          end if
31:          // Additive Increase
32:       else if !Oscillate then
33:          if idx + 1 < max_idx then
34:             idx + +
35:          end if
36:       end if
37:       Probing = true
38:    else if Δ ≤ 0.90 and Δ ≥ 0.75 then
39:       // Additive Decrease
40:       if idx > 0 then
41:          idx − −
42:       end if
43:    else if Δ < 0.75 then
44:       // First occurrence: Additive Decrease
45:       if first and idx > 0 then
46:          idx − −
47:          // Successive occurrences: Multiplicative Decrease
48:       else if second and idx > 0 then
49:          idx ← idx ∗ M_D
50:       end if
51:    end if
52: end while
```

Given that the available rates are constrained to the deterministic values of the transmission rate vector of Table I, this requires selecting the index corresponding to the adequate rate value. This paper addresses rate adaptation for Single Input Single Output (SISO) systems only, but the approach can be easily adapted to IEEE 802.11n or other Multiple Input Multiple Output (MIMO) systems by including the new MIMO enhancements as described in [2].

TERA adjusts the transmission rate based on a periodic review of the performance of transmissions over a time window ω. The rationale for using a time window is to ensure that transmission rates do not fluctuate too much and that transition between rates can proceed as seamlessly as possible. This is particularly important for such bandwidth sensitive applications as audio and video. We use $\omega = 100$ millisecond as an implementation guideline, because it provides sufficient information about the performance attained at given rates, and is also sufficient as a reactive time for adjusting rates.

Let $x_i(t)$ be the rate index of the ith user during time slot t. The throughput attained by a node (Γ) is calculated as in MINSTREL [12] by taking into account the probability of success and the packet transmission time, i.e., $\Gamma = P * M/\tau$, where *P = (number of successful packets)/attempts* denotes the probability of successful transmission using rate index $x_i(t)$, M denotes the Megabits of data transmitted, and τ denotes the time for one try of one packet over the air using rate index $x_i(t)$.

To adjust rates, the ratio of the current throughput is monitored over the reference throughput, which is obtained by applying an exponential weighted moving average (EWMA) filter. This filter can be easily adjusted depending on how sensitive the reaction should be to the surrounding environment. Let $\Delta = \Gamma_t/\Gamma'_t$ be the ratio of the current throughput over the reference throughput with exponential weighted moving average (EWMA) at time t, where Γ'_t is defined as $\Gamma'_t = \alpha * \Gamma_t + (1 - \alpha) * \Gamma'_{t-1}$ for $t > 1$ and $\Gamma'_t = \Gamma_t$ for $t = 1$. The coefficient α represents a constant smoothing factor between 0 and 1. A higher α value discounts older observations faster. We recommend a value of α between 0.75 and 0.95 to ensure sufficient reactiveness to changes in throughput and the surrounding environment.

During the transmission window, TERA keeps track of the packet successes as well as the number of attempts. These are used for calculating the probability of success needed to compute the attained throughput at time t (Γ_t). The EWMA filter is used on the current throughput to prevent TERA from being too sensitive to throughput fluctuations.

The initial step in Algorithm 1 consists of determining if the node is in the probing state. The *Probing* flag is used to signal that the protocol is in the probing state. The objective of having such a state is that, in some cases, increasing the transmission rate does not necessarily translate into a higher throughput. The transmission rate is increased only when the node is in the probing state, where the node searches to find the highest rate for the best throughput as quickly as possible with a combination of two probing frequencies.

To determine whether a higher rate produces a better throughput, the node keeps track of the last throughput it attained using the last rate index, so that the node can fall back to the best throughput. While the node is in probing state, the node sets the frequency of its probing rate to 100ms if the throughput at the higher rate is larger than or equal to the reference throughput. This allows the node to increase its rate quickly as long as the resulting throughput does not deteriorate. Conversely, if the probing of a higher data rate produces a deterioration in throughput, the frequency with which a new rate is probed is set at 900ms. The rational for a 900ms delay is

to keep the probing window under 1s, so that TERA can react to improving channel conditions resulting in better throughput without undue delays. When the current throughput is better than the reference throughput, TERA attempts to increase the rate either additively or multiplicatively. If the increased rate provides a better throughput, it is adopted and the algorithm exits the probing state.

To decrease the data rate, the node attempts to find a lower rate that provides better throughput and rely on its probing rate increase to bring it back to the highest transmission rate possible for the best throughput. After ensuring that TERA is not in a probing state for increasing rate, TERA checks the value of Δ, which is the ratio of the current throughput over the reference EWMA throughput. If Δ is at least 1, this is an indication that the transmission rate is of good quality. Accordingly, if there is no oscillation in rate selections, then the rate is increased multiplicatively with successive successful probing increase: $x_i(t+1) = x_i(t)+x_i(t)$; otherwise, the rate is increased incrementally (i.e., $x_i(t+1) = x_i(t)+1$).

If Δ is between 0.90 and 0.75, then the quality of transmission rate is marginal, and the rate is decreased incrementally (i.e., $x_i(t+1) = x_i(t)-1$), so that the node may be able to find a better throughput by transmitting at lower rates.

If Δ condition is less than 0.75, then the transmission rate is of poor quality, and the rate is decreased incrementally with the first occurrence (i.e., $x_i(t+1) = x_i(t)-1$), and is decreased multiplicatively with each successive occurrence (i.e., $x_i(t+1) = M_D x_i(t)$, with $0 < M_D < 1$ a multiplicative factor). The rationale for the initial incremental decrease of transmission rate is to probe the receiver at a slightly lower data rate. If that fails to improve throughput, it is a clear indication that the channel conditions are poor and that the data rate must be lowered drastically.

If Δ value is between 1 and 0.90, the algorithm does nothing in order to account for the fact that the maximum throughput of the next lower rate is still lower than the current throughput.

The *Multiplicative* flag in Algorithm 1 denotes that the node has experienced successive successful probes (e.g., two successive additively increases of the throughput). In this case, the algorithm enters the *Multiplicative* state, which allows the rate to be increased multiplicatively. The *Oscillate* flag in Algorithm 1 is used to detect scenarios in which rate selection is alternating back and forth. The typical scenario is one in which the transmitter keeps trying to increase its rate, but fails to provide a better throughput doing so every time. When this happens, the frequency of probing is adjusted by resetting *resetTimeInterval* appropriately to every 100ms or every 900ms.

Note that multiplicative increases and decreases in data rate are needed for TERA to respond quickly to drastic changes in channel conditions. Using only incremental changes to transmission rates would make the transmitter too slow in responding, given that there are 10 possible rates in IEEE 802.11g (Table I) and rate adaptation must take place over a transmission window.

IV. Performance Evaluation

A. Simulation Results

We compared the performance of TERA against several relevant rate adaptation schemes using NS-3 [14]. Unless otherwise specified, the following parameters are used in the simulations: a packet size of 512 bytes, a drop tail queue with a maximum length of 100, the IEEE 802.11a MAC model, a constant speed propagation model, a log distance propagation loss model ($L = L_0 + 10n \log_{10} \frac{d}{d_0}$), a transmission range of 150m, interference range of 250m, and UDP throughput. Each simulation was performed for 60 seconds; simulations lasting longer than 60 seconds produced similar results in benchmarking runs.

First, we considered the case of two nodes moving during data transmission. The source node moves at a speed of 1 m/s away from the target with no pause. The objective is to see how decreasing signal strength and fading affect performance. Similarly, we have the source node moves at a speed of 1 m/s towards the target with no pause. The objective is to study the reactiveness of protocols with increasing signal strength.

Fig. 1 reports the results for this scenario. The throughput results are at the lowest when nodes are farthest apart at 140m and greatest when they are in close proximity. ONOE, being conservative in raising rates, takes some time before transmitting at the optimal rate. AMRR has many sporadic dips throughout the experiment, this is probably due to the exponential back-off mechanism. RRAA does not perform well in this fading scenario because it lowers its rate quickly due to employing short-term loss. MINSTREL performs well but it takes dips during transition such as 30m, 50m, 90m, and 115m due to its probing and trials and errors nature before achieving the optimal rate. CARA's control packets probing for collision detection suffers a slight performance decrease. The fading pattern works well for ARF due to gradual increasing and decreasing signal strength; however it still does not perform well as TERA. During the fading transition, TERA lowers its rate accordingly to adapt to it. The same applies to MAICA. However, due to the credit threshold, MAICA is slow at the initial start because it does not employ multiplicative increase like TERA. Fig. 2 provides the opposite scenario where a node is moving toward the target node. Ideally, protocol should be able to recognize the increasing signal strength and increase their rates accordingly. Again, TERA has a smooth transition between rates due to its reactiveness.

Second, we select nine sources placed in a 10x10 grid topology and assign them 25 target nodes with the flows being exponentially distributed with mean of 3 seconds, for a total of 225 (25x9) distinct flows (see Fig 4(a)). We choose the grid topology and set it up this way because we did not want to employ any specific routing protocol, which could have influenced the results on rate adaptation. Flows are exponentially distributed to ensure that this scenario does not favor any approach. Finally, we have 50 flows in a grid 10x10 topology where each node transmits to its immediate neighbor.

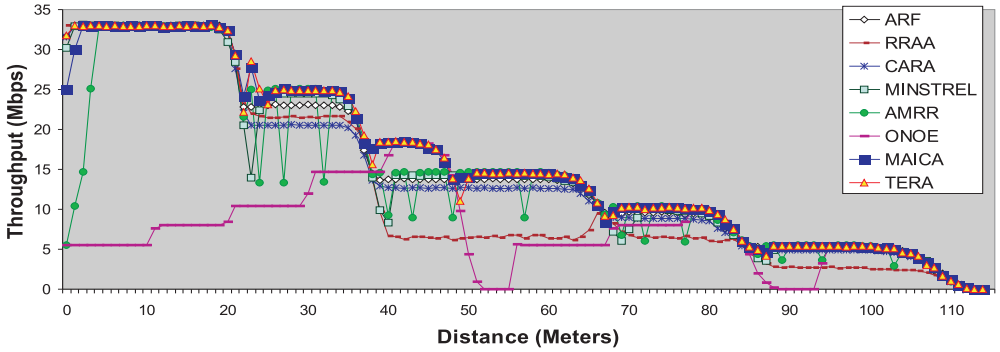


Fig. 1: Node Moves Away From Target

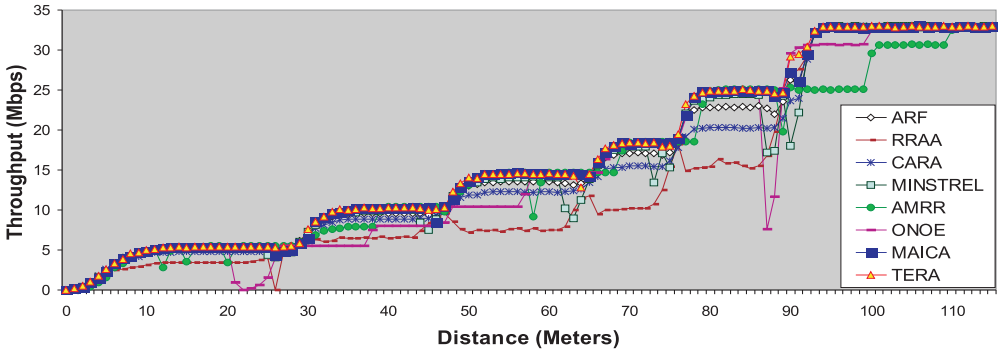


Fig. 2: Node Moves Towards Target

Fig. 3 shows the results for MAICA and TERA in a 100-node,10x10 grid topology with each node transmitting to each of its 8 immediate neighbors with exponentially distributed flows lasting an average of 3 seconds. In these scenarios, we show that TERA can perform as well as, if not better than, MAICA without the complexities incurred in MAICA for the careful tuning of parameters. Given that TERA is a better approach than MAICA, the rest of the subsequent scenarios and experiments do not include MAICA.

Fig. 4(b) reports the results for the 10x10 grid scenario described in Fig. 4(a) with exponentially distributed flows. AMRR, ARF, and ONOE do not perform well due to conservative approach with which they increase transmission rates. RRAA does not perform well because short-term loss ratio is unpredictable in this experiment.

Fig. 5 shows the results for the 50-flow scenario in which nodes are paired up with its immediate neighbors. In this scenario, progressive increase seems to outperform other aggressive probing protocols, as in ARF, CARA, and MINSTREL.

B. Experimental Results

We implemented TERA in the Linux Kernel Wireless Stack [13] to demonstrate that it can work correctly and independently of the Atheros chipset. We compare our approach against the default ATH5K Atheros chipset and a popular Linux rate control MINSTREL, given that these are

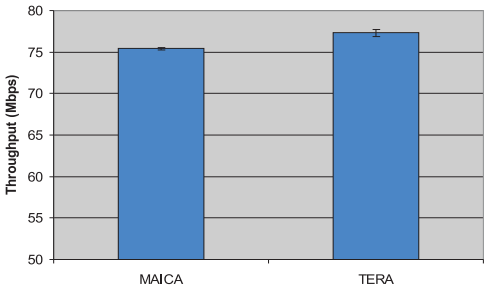
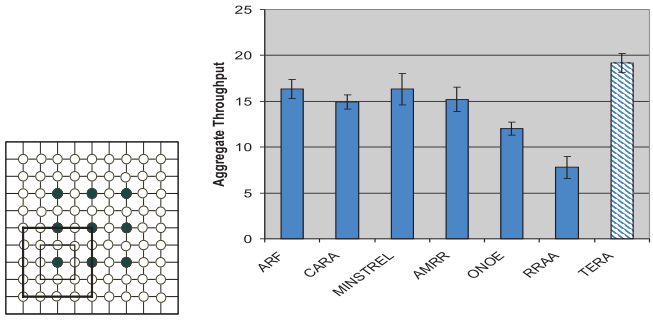


Fig. 3: MAICA and TERA



(a) Setup (b) Aggregate Throughput

Fig. 4: Exponentially Distributed Flows (mean=3s)

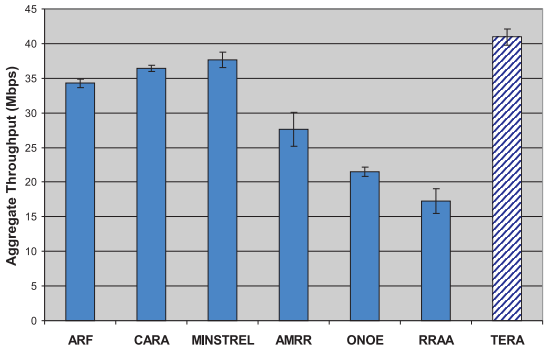
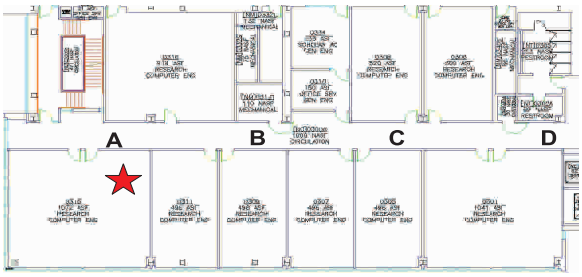
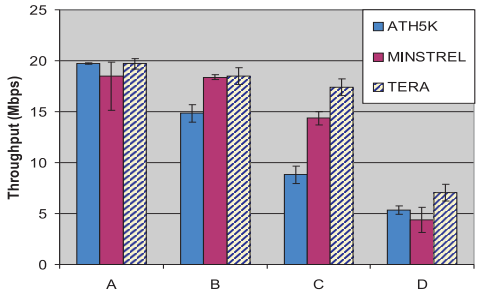


Fig. 5: 50 Flows



(a) Engineering Building (30m x 75m)



(b) Throughput vs Location

Fig. 6: Experiment Results: Throughput at Different Locations

available and widely deployed in many real-world settings. Many schemes in our network simulations are either not publicly made available or only exist as simulation code.

We set up our wireless access point in our laboratory as shown as a star symbol in Figure 6(a). We measure throughput at different locations around the building.

Figure 6(b) reports the experiment results of our protocol compared at two other widely popular and deployed Linux rate adaptations. At location A, nodes are about 10m from the access point (AP) router and they are all able to obtain high throughput approximately 20 Mbps. We repeat the experiment on average of seven times and find that MINSTREL has a large standard deviation, perhaps due to its spending 15 percent of the time trying random rates even though its current is working perfectly fine. At location B, 20m away from the AP, protocols suffer only slight degradation in throughput. ATH5K, however, gets stuck in sub-optimal rates. At location C and D, which are 40m and 65m away from the AP router respectively, TERA was able to perform much better than ATH5K and MINSTREL. At location C, TERA is 50 percent better than ATH5K and 15 percent better than MINSTREL. At location D, TERA is 20 percent better than both ATH5K and MINSTREL.

V. Conclusions

We proposed a new throughput-centric TERA protocol, based on implicit feedback. The key insight of our work is that throughput can provide a great measurement tool for adapting rates robustly. We evaluated TERA extensively via network simulations and real world settings with many different scenarios for fading, interference collisions. The results show that TERA performs consistently better than many multi-rate adaptation schemes that are widely used and deployed today, especially in dense networks. Furthermore, TERA is surprisingly simple, practical, and is compatible with today's WiFi networks.

References

[1] ATHEROS Communications. http://www.atheros.com.
[2] D. Nguyen and J.J. Garcia-Luna-Aceves. *A Practical Approach to Rate Adaptation for Multi-Antenna Systems*. Proc. IEEE ICNP, 2011.
[3] D. Nguyen and J.J. Garcia-Luna-Aceves and C. Westphal. *Multi-Rate Adaptation with Interference and Congestion Awareness*. Proc. IPCCC, 2011.
[4] S. Gatreux, V. Erceg, D. Gesbert, and R. W. Heath. *Adaptive Modulation and MIMO Coding for Broadband Wireless Data Networks*. IEEE Communications, 2002.
[5] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 Packet Delivery from Wireless Channel Measurements. In *In Proc. of SIGCOMM*, New Delhi, India, 2010.
[6] G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks. In *Proc. ACM MOBICOM*, Rome, Itaty, 2001.
[7] IEEE 802.11 Working Group. *Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) specifications*, 2007.
[8] J.C Bicket. *M.S thesis: Bit-rate selection in wireless networks*. MIT Press, 2005.
[9] A. Kamerman and L. Monteban. *WaveLAN-II: A High-performance wireless LAN for the unlicensed band*. Bell Lab Technical Journal, 1990.
[10] J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: Collision-aware Rate Adaptation for IEEE 802.11 WLANs. In *Proc. INFOCOM*, 2006.
[11] M. Lacage, M. H. Manshaei, and T. Turletti. *IEEE 802.11 rate adaptation: A practical approach*. INRIA Research Report.
[12] Minstrel Linux Wireless. http://linuxwireless.org/en/developers/Documentation/mac80211/RateControl/minstrel.
[13] The Linux Kernel. http://www.kernel.org/.
[14] The ns-3 project. http://www.nsnam.org/.
[15] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proc. MOBICOM*, 2006.
[16] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *Proc. Mobicom '01*.