

# Maximizing Throughput in a Simulated Wireless Environment

Brian Russell  
Department of Computer Science  
Rutgers University  
morbius@cs.rutgers.edu

## ABSTRACT

This work describes the results of experiments on the effects of noise and congestion in a simulated wireless environment. The basic mathematical model for environmental noise and congestion is presented along with descriptions of the experiments run and the results obtained. This paper also suggests directions for future research in this area.

## 1. Introduction

In a wired network, the transmission of frames is affected by router congestion [1]. In a wireless network, the transmission of frames is affected by congestion in a multihop environment and noise in the environment. The wireless node, acting as agent, cannot observe noise or congestion directly, it can only observe its own throughput. Worse, the responses to noise and congestion have conflicting effects on throughput. A node can respond to noise by reducing the rate at which bytes are transmitted, thus making the transmissions more robust to noise, but this same action would actually reduce throughput as a response to congestion [2]. The response to congestion would be to increase the frequency of frame transmission (i.e. more frames per unit time); such an increase would increase the vulnerability to environmental noise [2]. The goal of a wireless node would be to devise actions that would maximize throughput.

This paper describes the theoretical basis for the automated detection of changes in environmental conditions and their practicality in the face of noisy throughput observations. Experiments were run to consider learning rates, change detection and different algorithms to cope with noise in a simulated wireless environment.

Section 2 describes the theoretical basis of the simulated environment, including the probabilistic formulas for noise and congestion. Section 3 describes the experimental setup and the different algorithms used to manage throughput under changing noise and congestion conditions. Section 4 describes the experiments run and the results obtained. Section 5 describes the overall results of the experiments. Section 6 offers some suggestions for future work and finally, Section 7 offers the conclusions.

## 2. Theoretical Basis

In practice, the probability of a transmitted byte of a frame being affected by noise is a function of transmission speed and the error correction capability of the transmission protocol [1]. The faster the transmission speed, the greater the probability of error and vice versa. In the experimental setup, the probability of corruption of a byte was a probability  $p$  and the probability of a frame being successfully received was a function of the length of the frame  $(1-p)^n$ , where  $n$  is the number of bytes in the frame.

This assumption meant that experiments could be conducted by sending UDP packets

on a wired network without regard to transmission speed, which would have been impossible to control directly in code using a wired network. The experiments, although run without benefit of a wireless network, was still one in which there was a probabilistic loss of frames due to varying (externally controlled) conditions.

One limitation of these experiments was the treatment of congestion. In practice, frame loss is not a function of frame size, but a function of the rate of frame transmission, i.e. how many frames were transmitted during per time period. In these experiments, congestion was simply treated as a fixed probability  $q$  of frame loss, which is certainly an oversimplification.

Noise and congestion were combined into a single formula for the probability of not losing a frame in transmission:

$$1 - (1-q)(1-p)^{**n} \quad (1)$$

The decision of whether or not a frame was lost was made by using random numbers between 0 and 1.0. If the random number was greater than the value obtained from the above formula, the frame was considered "arrived", otherwise the frame was considered "lost".

Three distinct artificial noise and congestion conditions were created, each with a known optimal frame size and maximum throughput level. These conditions, called "scenarios", were used to evaluate the behavior of the mechanisms used in simulated wireless conditions. Scenario I set the noise probability of corrupting a byte in a frame  $p = 0.01$  and the probability of losing a frame due to congestion  $q = 0.1$ . The maximum throughput rate was 33 bytes/second and the optimal frame size was 100 bytes. Scenario II set  $p = 0.01$  and  $q = 0.45$ , with a maximum

throughput rate of 20 bytes/sec and optimal frame size of 100 bytes. Scenario III set  $p = 0.015$  and  $q = 0.1$  with optimal frame size 66 and maximum throughput of 20 bytes/second.

The last seven of the nine experiments run used the three noise/congestion scenarios and made probabilistic decisions about losing frames due to noise and congestion based on formula (1) above and Epsilon-greedy and Matcher algorithms that used different frame sizes as alternatives to select during execution. The available frame sizes were 30, 60, 100, 150 and 210 bytes. Table 1 shows the probability of a frame being "lost" for each available frame size in each of the noise and congestion scenarios.

Table 1: Probability of Frame Loss

Frame Scenario I    Scenario II    Scenario III  
Size

30	0.33427	0.59316	0.42809
60	0.50756	0.69906	0.63657
100	0.67057	0.79868	0.80145
150	0.80069	0.87820	0.90674
210	0.89095	0.93336	0.96234

### 3. Environmental Setup

There were two kinds of experiment architectures. The early architecture simulated the transmission of wireless frames at the link level by sending UDP segments from a sender process to a separate receiver process on the same machine. Due to the inherent variability of transmission time on a timeshared machine, throughput could not be measured accurately in terms of bytes per second, but only in a somewhat contrived bytes per frame. Later experiments ran as simulations of sending and receiving in a single process and calculated throughput in terms of bytes per virtual clock tick. Running the later experiments as single process

simulators without transmitting UDP segments resulted in much faster experiments -- a reduction from hours to seconds in some cases, as identified in the section describing the individual experiments.

In the early experiments, the stochastic decision to say that a frame was lost in transmission was made by a separate algorithm in the receiver software using random numbers. The later experiments had a separate "fabric" subroutine that made the decision.

The experiments used four different algorithms to control throughput, each implemented as a C++ class. The mechanisms were called Tracker, Matcher, EpsGreedy and Sessile. The Tracker class compared throughput for three adjacent frame sizes in a sliding window. If the throughput for the smallest frame size was greatest, the center of the window would be decremented by one. If the throughput for the largest frame size was greatest, the center of the window would be incremented by one. The Matcher class used the small number of alternative frame sizes where the probability assigned to each alternative was dynamically weighted by the throughput obtained for each frame size. The EpsGreedy class implemented an epsilon-greedy algorithm that randomly chooses the frame size with the highest throughput 95% of the time and the other alternatives 1.25% of the time. All of these classes explored to find better throughput alternatives in situations where the noise and congestion conditions can (and did) change. The Sessile class was used as a control. It simply set the frame size to be one fixed value (i.e. 100).

#### **4. Experiments**

This section describes the experiments that were run and the results obtained from each.

All of the experiments were intended to produce plottable data showing different relationships between noise, congestion, learning rates and throughput.

ex1: Generated throughput as a function of noise for fixed frame sizes. The sender transmitted frames as UDP segments of a size fixed at compile time. The receiver component received the UDP segments and treated them as frames, varying the noise level between 0.0 and 1.0. The output was throughput in terms of bytes-per-frame as a function of the noise level.

ex2: Generated throughput as a function of frame size  $n$ , where  $1 \leq n \leq 1500$ . The sender was modified to receive directives from the sender controlling frame size on a port different from the one used to transmit UDP segments to the receiver. Different variants of this experiment transmitted frames as UDP segments from the sender to the receiver in lengths varying from 1 to 1500, inclusive. The patterns encompassing every frame size between 1 and 1500 had to be repeated as many as 150,000 times to result in convergence over all frame size. This repetition took more than 22 hours to get converged data for all frame sizes.

Trying different patterns of frame size transmission, from simple increment from 1 to 1500, or random patterns over the 1500 frame sizes did not produce faster convergence.

ex3: Implemented automated mechanisms that attempted to control frame size to get maximum throughput, using the separate sender and receiver processes that communicated with UDP segments. One result was that the Matcher class proved that it could work well with observations of noisy throughput as well as observations of noiseless throughput.

Another result of this experiment demonstrated that the Tracker class worked well with observations of noiseless throughput, moving its sliding window until it centered on the frame size that gave it the highest throughput. However, the introduction of noise into the throughput rendered the Tracker unable to center on an optimal frame size, instead causing it to wander over frame sizes like a car careening out of control would be unable to stay on the road.

ex4: Generated throughput as a function of epsilon for an epsilon-greedy algorithm implemented as a C++ class, again using the separate sender-receiver architecture. This experiment demonstrated that noise prevented convergence to a smooth curve in the output data.

ex5: Generated throughput as a function of time using the separate sender-receiver architecture. This experiment demonstrated that system interference invalidates any accurate real-time measurement when transmitting frames as UDP segments.

ex6: The first true wireless simulator done entirely in software without sending or receiving UDP segments. This experiment had separate code to simulate transmission fabric that made the probabilistic decisions about frame loss. The experiment ran Matcher and Epsilon-greedy algorithms to compare performance for scenarios I, II and III. Produced throughput as a function of virtual clock time.

ex7: Generated cumulative total throughput as a function of virtual time for Matcher and Epsilon-greedy algorithms. This experiment demonstrated that it could produce output compatible with change detection algorithms. A different variant generated throughput as a

function of virtual time for different learning rates for Matcher and Epsilon-greedy algorithms. Again because of noise, the data generated would not fit a smooth curve.

ex8: Generated throughput as a function of learning rate for Matcher and Epsilon-greedy algorithms. Demonstrated that noise produced data that was not a smooth curve and that no noise led to unchanged output data, because the Matcher and Epsilon-greedy algorithms always made the same decisions.

ex9: Generated an error value as a function of time, where error was the square of the difference between predicted throughput and observed throughput. The expected result was reduction of error over time under stable noise conditions as the algorithm learned which frame size produced the highest throughput for the given (and not directly known) noise and congestion conditions, then a larger difference when the scenario was changed, followed by decreasing error over time as the algorithm learned which frame size produced the highest throughput for the a new set of noise and congestion conditions. The actual result showed no curve at all because the observations of noisy throughput did not follow a curve.

## 5. Results

The early experiments demonstrated that stable noise in the simulated wireless environment will lead to unstable variable instantaneous measures of throughput. A wireless node acting as an agent can measure throughput directly and cannot directly observe noise or congestion. A single increase or decrease in the instantaneous throughput perceived by a wireless node agent cannot therefore be used to determine if conditions have actually changed and cannot be taken as a cue for the agent to perform corrective or information gathering actions.

The latter experiments have demonstrated that over time smoothing of noisy throughput measurements does not prevent the selection of very different actions by the Matcher and Epsilon-greedy algorithms for slightly different learning rates even when the throughput information and random number generation are identical. For the epsilon-greedy algorithm, the choice of which frame size to use as the "greedy" alternative is determined by which frame size has the largest alternative. With different learning rates, the throughput calculations for adjacent alternatives will eventually approach similar values for different learning rates, but the largest throughput value for one learning rate may not be the same as the largest throughput value for an adjacent learning rate. When this occurs, the epsilon-greedy algorithm will choose different alternatives based on throughput information that differs only by the learning rate. Similarly, the Matcher algorithm bases its selections on weights that will differ slightly, again only because of the different learning rate. When a random number is generated that falls between the different weights, a different alternative will be selected. The result for either algorithm is completely different behavior even when the other conditions are otherwise identical. Significant and drastic long term changes of behavior from small changes in conditions (in this case the values used as part of random decisions) are entirely consistent with the basics of chaos theory [3].

## 6. Future Work

The mathematical model for the effects of noise and congestion on packet loss should be either verified or corrected by observing the effects of noise and congestion in an actual wireless environment to develop a model that accurately includes the effects of noise as a function of transmission speed rather than

frame size alone. The effects of congestion also need to be accurately modeled. The goal is to devise policies for wireless nodes to respond to varying noise conditions and congestion levels and to devise the means to quantify the performance of the policies in terms of responsiveness to varying conditions and maximizing throughput. Ultimately, wireless node agents should be able to devise their own policies in response to varying noise and congestion conditions.

## 7. Conclusions

Wireless communication in the real world is inherently complicated by transmission noise and congestion. The experiments performed here were based on an unverified model of noise and congestion, but they have shown that noise and congestion completely undermine the effectiveness of methods and algorithms that otherwise work successfully in noiseless conditions. These same experiments have also shown that an Epsilon greedy and Matcher algorithm can work with observations of noisy throughput, but the noise makes it difficult (if not impossible) to accurately quantify the responsiveness of the algorithms.

## REFERENCES

- [1] James F. Kurose and Keith W. Ross. Computer Networking: A Top-Down Approach Featuring The Internet, Third Edition. Addison-Wesley Publishing, 2005.
- [2] Kishore Ramachandran, Haris Kremo, Marco Greutser, Predag Spasojevic and Ivan Seskar. Practical Scalability Enhancements in Dense Rate-Adaptive IEEE 802.11 Networks, 2005.
- [3] Chaos Theory: A Brief Introduction. <http://www.imho.com/grae/chaos/chaos.html>.