

# USING FUZZY INFERENCE TO IMPROVE TCP CONGESTION CONTROL OVER WIRELESS NETWORKS

Matt Wozniak and Hala ElAarag  
Department of Mathematics and Computer Science  
Stetson University  
DeLand, FL  
{mwozniak, helaarag}@stetson.edu

While modern wireless networks have been in development for a couple of decades, the Transmission Control Protocol (TCP) which runs over those networks has existed since the mid 1970s. As it was developed before wireless networks were even conceived, TCP was not optimized to consider the physical characteristics of wireless network links. Specifically, TCP responds to packet loss due to link errors in the same way it responds to packet loss due to congestion: it cuts back the rate at which traffic is sent. A means of improving performance of TCP over wireless links is to classify packet losses, and react only to those losses perceived as being caused by network congestion. This paper seeks to use environmental variables available to TCP implementations to feed a fuzzy inference system that will classify packet loss as due to congestion or random collision without sacrificing the end-to-end reliability of TCP.

<b>Introduction</b>	<b>3</b>
<b>Related Work</b>	<b>3</b>
<b>Background Information</b>	<b>4</b>
<b>TCP Congestion Control and Wireless Networks</b>	<b>4</b>
<i>Figure 1: A simplified diagram of TCP slow start.</i>	5
<b>Fuzzy Inference</b>	<b>6</b>
<i>Figure 2: Membership functions for the linguistic values “Slow”, “Medium”, and “Fast”</i>	6
<i>Figure 3: Graphical Mamdani max-min inference (only the first three rules are displayed).</i>	8
<b>TCP+Fuzzy</b>	<b>9</b>
<b>Packet Loss Events and Classifier Inputs</b>	<b>9</b>
<i>Figure 4: A simplified control flow diagram depicting our proposal for a TCP+Fuzzy classifier.</i>	10
<i>Figure 5: Classifier inputs recorded after a TCP loss event.</i>	10
<b>Loss Classification with Fuzzy Inference</b>	<b>11</b>
<i>Figure 6: ns-3 simulation topology</i>	11
<b>The Fuzzy Inference Rule Base</b>	<b>11</b>
<i>Figure 7: Inter-arrival and delay calculation.</i>	12
<i>Table 1: Fuzzy rule set for the TCP packet loss classifier.</i>	13
<b>Simulation</b>	<b>13</b>
<b>Conclusion</b>	<b>16</b>
<b>References</b>	<b>16</b>

# 1. Introduction

TCP traffic makes up about 90% of all Internet traffic [SHI09] making it the most widely utilized reliable transport protocol. One aspect of TCP that could use improvement is its performance over wireless network links. When TCP detects a lost segment it reacts by resetting the congestion window (*cwnd*)\* and cutting the slow start threshold (*ssthresh*)† [ALL99, MOR03]. Over wired links packet losses are often caused by network congestion, but wireless links are subject to noisy and intermittent packet losses due to fading radio signals or interference. The effect on TCP is that it is prone to reducing *cwnd* when there is no congestion. This *cwnd* reduction necessarily implies throughput will be less than optimal [PAR00] thus it is of interest to prevent TCP from reacting to wireless link losses in the same way that it reacts to congestion losses.

One possible way for TCP to react more appropriately to link losses is to endow TCP with a segment loss classifier that can infer the reason for a particular loss. Based on this inferred reason TCP can efficiently react to segments lost due to link noise by choosing not to reset *cwnd* or reduce *ssthresh*. The classifier should only make use of information available to the TCP sender and receiver. Practically this constraint limits us to information that can be obtained by a sending or receiving TCP socket. Because the goal of congestion avoidance is goodput‡ maximization it is most effective for the TCP sender implement to implement a loss classifier. This paper proposes the creation of a fuzzy-inference-based loss classifier that can be easily appended to most TCP implementations.

To develop the fuzzy inference classifier we use ns-3 to model the network and simulate traffic. Our proposed TCP+FUZZY implementation was created by attaching a classifier to a TCP NEW Reno socket. However the classifier can be attached to most other implementations (like TCP Tahoe) as well. We simulate TCP+FUZZY, and a number of other TCP implementations, on a dumbbell network topology that subjects traffic to congestion and intermittent segment losses (i.e. noisy link losses).

The rest of this paper is organized as follows: in Section 2 we discuss past attempts to improve TCP performance on wireless links. In Section 3 we give an overview of TCP congestion control and fuzzy inference. Section 4 presents our proposal for a TCP+FUZZY classifier in detail. Section 5 discusses our simulation environment and configuration and Section 6 presents a summary of the conclusions drawn from this proposal.

## 2. Related Work

Many different solutions have been proposed to improve TCP performance on wireless links [MAX01, XYLO2, CHAO2, GER99, BAK97]. These solutions have varied in design and scope from TCP modifications to changes in link layer behavior. Balkrishnan et al. outline many of the solutions which include link-layer protocols endowed with forward error correction (FEC) or retransmissions (ARQ), split-connection protocols, and selective acknowledgments [BAL97, BAL98]. Split-connection protocols require a base station to break a TCP stream into two pieces such that the client talks to the base station, which in turn maintains a

---

\* The *cwnd* is the number of unacknowledged packets that can be in transmission at any given time. A higher *cwnd* raises the maximum possible throughput.

† TCP exponentially increases its *cwnd* until the size of *cwnd* is greater than *ssthresh*. *cwnd* is then linearly incremented.

‡ The ratio between the amount of useful information delivered and the total sending duration. Goodput is always less than the throughput which takes into account the total amount of data transmitted (including TCP/IP/Ethernet headers, interframe gaps, retransmitted bytes and protocol specific overhead (e.g. TCP acknowledgments)).

separate TCP connection with the server [BAL97, BAL98]. The disadvantages of this approach are covered in [PAR00]. Selective acknowledgements (SACK) are essentially a backwards compatible addition to the TCP protocol defining a way for a receiver to send back a dropped packet. Some suggest that SACK can help improve TCP performance over wireless LANs [BAL98], but others claim that SACK becomes inefficient in the presence of link-layer retransmission technologies [PAR00] and [SIK01] demonstrates SACK's poor performance in the event of correlated segment losses. TCP SACK also suffers from the same issue as non-SACK TCP in that losses are always interpreted as congestion. TCP Westwood, proposed by [CLA02], is a sender-side extension to TCP that improves congestion control by continually tracking the available bandwidth and setting the `cwnd` and `ssthresh` appropriately based on the bandwidth available at the time of the loss event.

With the exception of TCP SACK and TCP Westwood, the solutions outlined above depend on non-standard modifications to link layer protocols or the presence of packet-delivery middle-men (as is the case for split-connection protocols). Ideally no changes should be made outside of TCP as this introduces transport layer dependencies on the data link layer breaking the network model abstraction. Our proposed classifier has the advantage of operating independently of other network layers.

El Khayat et al. [Elko8] worked with applying machine learning towards improving TCP congestion control over wireless links. Using information attainable at the transport layer, they generated a database of packet loss characteristics used to train various machine learning algorithms. After experiencing a segment loss, the classifier maps a set of inputs to a single output (the concluded cause of loss event). The classifier's output indicates either congestion or link interference as the cause of a loss. Our proposed classifier uses fuzzy inference, rather than machine learning algorithms, to derive conclusions, removing the need to pre-populate the inference engine with training data.

Fuzzy control systems have been utilized in many networking-related areas. Zhang et al. [ZHAO5] used fuzzy control systems to improve TCP packet loss retransmission (RTO) times. Several improvements have been proposed [SAB06, CAL03, KHA07] to improve cache replacement algorithms by incorporating fuzzy inference. A proposed streaming media protocol [PAV05] makes use of fuzzy controllers for congestion control. Chrysostomou et al. [CHR07] proposed a new active queue management scheme that utilized fuzzy logic. Fuzzy logic has previously been used in TCP congestion control: Shi et al. [SHI09] developed a fuzzy controller that reacts in changes to the bandwidth utilization by continually adjusting `cwnd`. Their approach is fundamentally different than the classifier-based approach used in TCP+FUZZY. There is a demonstrated applicability of fuzzy inference in solving problems that are difficult to stochastically model or analyze. Fuzzy inference systems also allow problems to be defined intuitively using a propositional IF-THEN rule base (which is discussed in Section 3).

## 3. Background Information

### 3.1. TCP Congestion Control and Wireless Networks

To understand issues caused by TCP over wireless networks, one must first understand the internal mechanisms TCP uses for congestion control. For a full understanding of the relevant aspects of TCP readers should refer to [RFC2581] and [RFC793]. In this section we briefly summarize TCP's congestion control behaviors.

As a TCP connection persists, packets are sent from a sender to a receiver. During slow-start, with every packet that is successfully acknowledged by the receiver, the sender's `cwnd` is increased by one full segment size. A consequence of this can be seen in Figure 1. As `cwnd` increases the number of packets in transmission grows exponentially. At the point at which a loss is realized by the sender (perhaps by three

duplicate ACKs or a transmission timeout), `ssthresh` is halved, `cwnd` is reset to one full segment size and TCP enters a slow-start state. Note that TCP Reno is capable of entering into a fast recovery mode which avoids cutting `cwnd` and `ssthresh` by retransmitting only the triply-acknowledged packet, but if more than a single packet has been lost TCP Reno will timeout, cut `cwnd` and `ssthresh`, and enter slow-start.

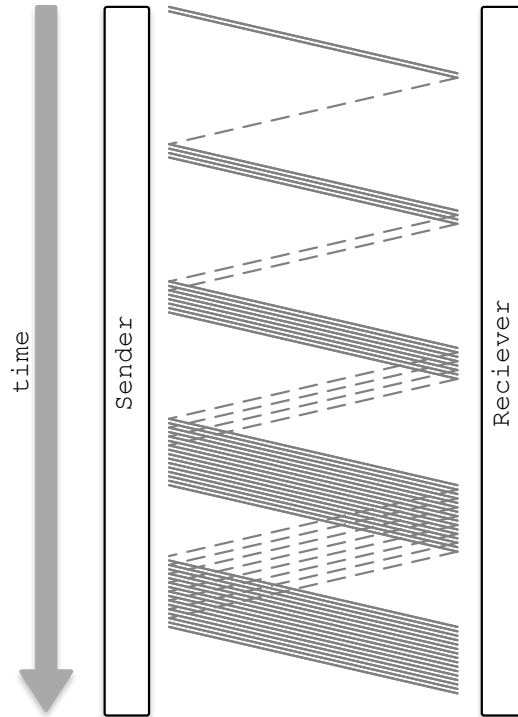


Figure 1: A simplified diagram of TCP slow start.

TCP's approach works well for packet losses that are a result of network congestion. In this situation the intuitive response is to reduce that rate of outgoing traffic in an attempt to resolve the congestion issue. But consider the scenario where a packet is dropped because of a link-error over a wireless connection. It is not optimal for TCP to reduce the rate of outgoing traffic when congestion is not a concern. Every time a non-consecutive packet loss occurs, `cwnd` is reset and `ssthresh` is recalculated based on the following formula:

$$\text{ssthresh} = \max (\text{FlightSize} / 2, 2 * \text{SMSS}) \quad (1)$$

Where `FlightSize` is the amount of outstanding data in the network and `SMSS` is the sender's maximum segment size. `FlightSize` can often be approximated by `cwnd`. One can think of `ssthresh` as a gauge of the amount of traffic that a connection can sustain at a point in time. But that conceptual model is only valid under the assumption that packet losses are mostly caused by congestion related issues. As soon as link-errors must be considered this conceptual model should be abandoned.

This summary of TCP's congestion control mechanisms and wireless networks has not considered the possible presence of link-level bit-error reduction mechanisms such as interleaving or FEC which are often found on cellular networks [GURO3]. These mechanisms are present because of the high cost of bit errors due to high latency, but they are limited in scope. For example, FEC makes only three attempts at a retransmission [GURO3]. Poor radio conditions and mobility are some of the various causes for corruption

errors on wireless links. Link-layer retransmissions are not ideal because the network conditions are obfuscated from the TCP implementation. If, for example, a packet is retransmitted at the link layer several times, its RTT will deviate significantly from the average. It also may be possible that TCP retransmits the packet by the time the link layer has correctly delivered the original.

### 3.2. Fuzzy Inference

In this section we present a brief overview of fuzzy logic and fuzzy inference which we use to classify packet losses as either due to congestion (C) or (LE). For theoretical discussion of multi-valued logic and fuzzy set theory we recommend the reader to [BERO8]. Background in fuzzy control systems and fuzzy inference can be found in [CHE00] and [ROS10]. While traditional logic contains only two truth values (true and false or 1 and 0), fuzzy logic may contain an infinite number of truth values on the continuous range  $[0, 1]$ . A fuzzy set is a set whose memberships are defined by a function that maps the elements of the set to  $[0, 1]$ . These mapping functions are called membership functions, and there may be an arbitrary number of them on any given domain. Domains of discourse, usually of time-varying quantities, (e.g. inter-arrival time, age, cost) are known as linguistic variables, and the membership functions defined on them are called linguistic values.

A fuzzy inference engine is composed of a set of input variables and a set of output variables. A set of linguistic rules defined inside of the inference engine maps the inputs to the outputs. These rules follow the modus ponens (IF-THEN) form and an inference engine may have an arbitrary number of them defined. Figure 2 depicts a linguistic variable “Speed” and its associated membership functions (“Slow”, “Medium”, and “Fast”).

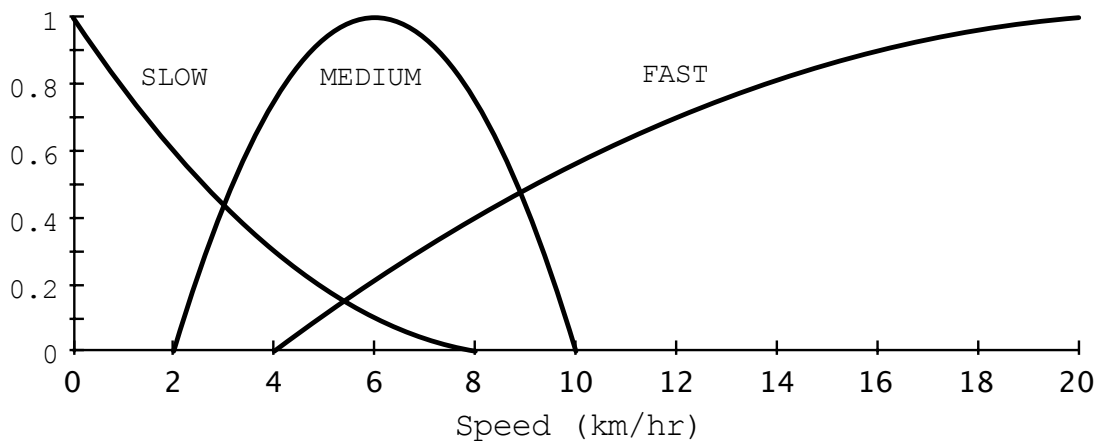


Figure 2: Membership functions for the linguistic values “Slow”, “Medium”, and “Fast”

The membership functions in Figure 2 represents the extent that a given value on the domain belongs to that linguistic descriptor. For example, a speed of 6 kilometers per hour is roughly 0.2 Fast, 1.0 Medium and 0.1 Slow (i.e. it has only 0.2 membership on the fuzzy set represented by the linguistic value Fast). It is not necessary for the membership functions to add to one at a particular point on the domain. These linguistic values are completely subjective in definition and tailored to the problem with which they are being applied. Hence, fuzzy inference is a type of expert system. Figure 2, for example, might be appropriately defined for a problem dealing with marathon runners. A world class marathon runner may run around 20 kilometers per hour while a beginner may run a marathon at only one or two kilometers per

hour. Thus the membership functions defined for the three linguistic values are intuitive. Ross [ROSS10] discusses the handful of ways in which these membership functions can be defined: intuition, inference, rank ordering, neural networks, genetic algorithms, and inductive reasoning. This paper uses an intuitive approach to define and optimize the membership functions.

Once membership functions have been defined, a fuzzy rule base must then be defined. A fuzzy rule-base is simply a series of implications consisting of a variable number of antecedents and consequents. These rules are defined in terms of linguistic variables and values. Consider the problem of modeling the average speed of a marathon runner as a function of the input variables “Temperature” and “Humidity”. Inference rules must be generated that satisfactorily model the problem. An example of such rules might be:

1. IF Temp is Hot and Humidity is High THEN Speed is Low
2. IF Temp is Mild and Humidity is High THEN Speed is Medium
3. IF Temp is Mild and Humidity is Mild THEN Speed is Fast
- ...
- n. IF Temp is Mild and Humidity is Low THEN Speed is Fast

These implications can be generalized in the Mamdani form for any number of antecedents (inputs):

$$\text{IF } X_1 \text{ is } A_k^1 \text{ and } X_2 \text{ is } A_k^2 \text{ and } \dots \text{ and } X_n \text{ is } A_k^n \text{ THEN } Y_i \text{ IS } y_k^i \text{ for } k = 1, 2, \dots, r \quad (2)$$

The rule sets may be conjunctive or disjunctive. In this paper rules have been aggregated in a disjunctive manner. Rules are aggregated using Mamdani max-min inference [ROSS10] [CHE00], a commonly used and computationally cheap inference method traditionally utilized in fuzzy control systems. Mamdani max-min inference is depicted graphically in Figure 3, where the average speed of a marathon runner is modeled in terms of the temperature and humidity during a race.

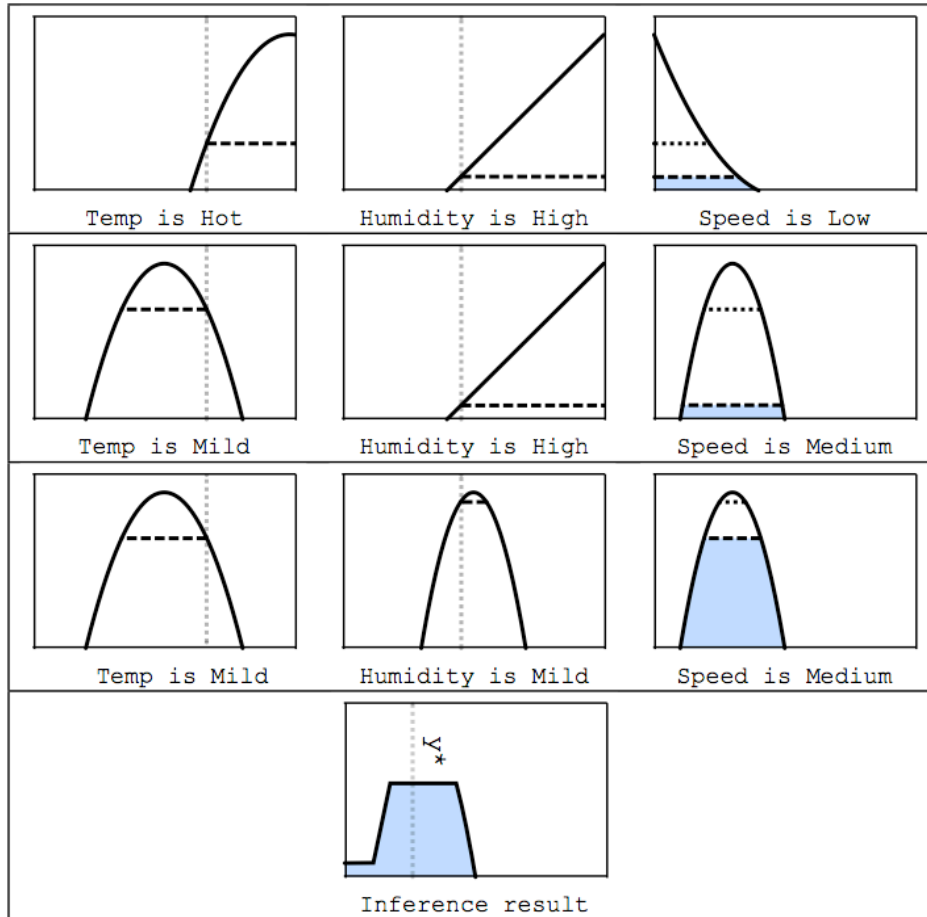


Figure 3: Graphical Mamdani max-min inference (only the first three rules are displayed).

Figure 3 depicts a Mamdani inference system, where each horizontal series of graphs corresponds to a rule in the rule set (membership functions for Temperature and Humidity have been created, while the membership functions for Speed are equivalent to those presented in Figure 2). Inputs to the system can be crisp values that take the form of delta functions after fuzzification\*. The delta function is represented by the intersection of the dotted vertical line on the x-axis with the membership function value at that point. The minimum value of the inputs is intersected with the membership function for (producing the highlighted fuzzy sets to the right of each inference rule). For every rule a derived fuzzy set produced and these sets are aggregated using either conjunction or disjunction (for max-min inference conjunction is used). This process produces the final inference result presented at the bottom of Figure 3. This result is a fuzzy set with low membership values for small values of  $x$  (speed), with larger membership values for medium values of  $x$ , and finally no membership for large values of  $x$ . Because the result is a fuzzy set, we must now look at techniques for finding representative crisp values that can be used in a classifier. The act of transforming a fuzzy set into a crisp value is known as “defuzzification”.

\* The process of creating fuzzy sets from crisp inputs. There are other means of fuzzification of interest in applications where there may be uncertainty about the inputs. We do not consider those methods here.

There are various methods for defuzzification and like other aspects of fuzzy control systems, selecting the right one is a subjective task. This paper makes use of an approximate centroid method for turning a fuzzy set or function into a scalar. The centroid method is defined as follows:

$$y^* = \frac{\int \mu_c(y) \cdot y \, dy}{\int \mu_c(y) \, dy} \quad (3)$$

Where  $y^*$  is a scalar representing the crisp inference,  $y$  is the domain of the output and  $\mu_c(y)$  is the implied membership function of the rule base. Equation 3 is mathematically equivalent to calculating the center of mass on the  $x$  axis for a given function  $f(x)$ . Thus the centroid method can conceptually be thought of as the “center of gravity”. The purpose of defuzzification is to arrive at a single, representative value for a fuzzy set. In Figure 3,  $y^*$  is indicated by a dotted vertical line passing through the inferred fuzzy set.

Reading the rule set in Figure 3 it is easy to draw intuitive conclusions about the relationships between the inputs and outputs. For example, if the temperature and humidity are high, runners can be expected to average slightly less than normal speed. It is necessary, of course, that the writer of the fuzzy rule base have some knowledge of the processes that affect human athletic performance, e.g. humidity and heat can have a detrimental effect on a runner’s speed. Fuzzy inference is only as good as the rules provided.

## 4. TCP+Fuzzy

In this section we present our proposal for a fuzzy-inference-based segment loss classifier. The classifier can be built onto any TCP implementation given that it meets certain constraints that are discussed in section 4.3.

### 4.1. Packet Loss Events and Classifier Inputs

Figure 4 shows an overview of the major TCP behavior change introduced in TCP+FUZZY. In the event of a loss `CWND` is reduced only if the loss experienced was caused by congestion. The classifier is embedded into a TCP implementation and is called at the `CongestionLoss?` stage of the flow diagram in Figure 4. Performance-affecting congestion avoidance measures are avoided if the loss experienced was due to link interference. Important implementation details that must be considered for any loss classifier include the choice of classifier inputs and how to deal with the possibility of loss misclassifications.

In Figure 5 it is observed that TCP has suffered a loss during the slow start phase. Before the sender can realize a loss has occurred, the retransmission timer for the lost segment must expire. The lost segment is denoted with an incomplete line and two crossing lines marking the drop. Later in the timeline the retransmission timer expires; it is at this point that a loss event is triggered internally and the lost segment is retransmitted. What information available to TCP can be used to infer the loss cause? The inputs must obviously correlated in some way to the output. They must also be available to TCP receivers and (most importantly) senders. Two good indicators that fit such criteria are the one-way delays and inter-arrival times for a TCP socket in communication.

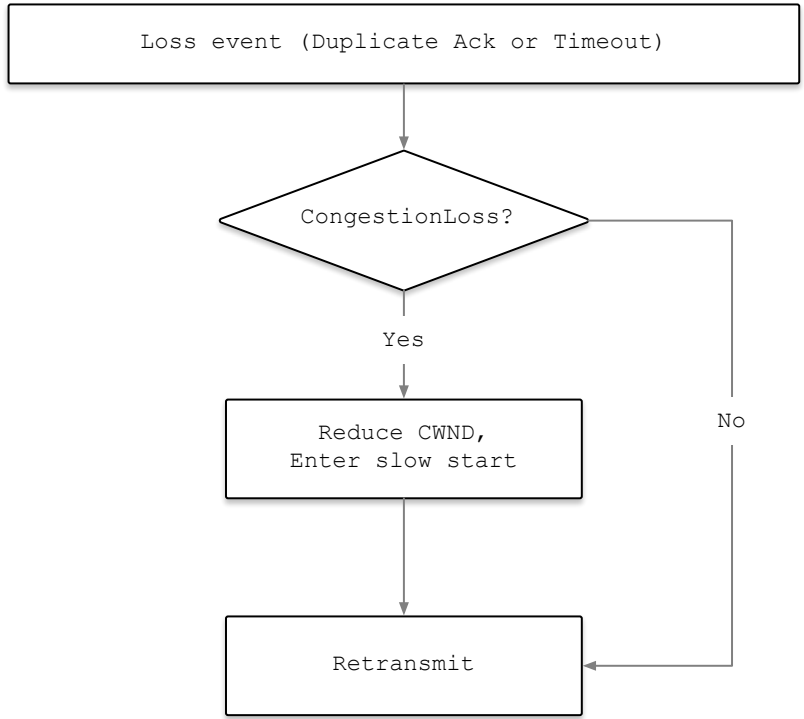


Figure 4: A simplified control flow diagram depicting our proposal for a TCP+Fuzzy classifier.

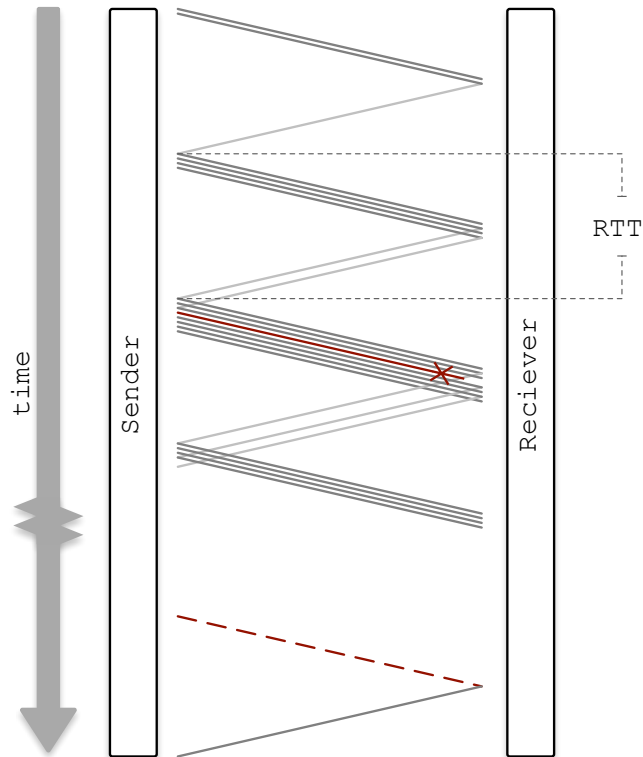


Figure 5: Classifier inputs recorded after a TCP loss event.

## 4.2. Loss Classification with Fuzzy Inference

Once the inference system is correctly classifying packet losses to an acceptable degree, it must be ensured that TCP+FUZZY does not hog more than its fair share of network bandwidth. TCP congestion control constantly adjusts individual throughput such that throughput gets distributed equally across all TCP connections on a network [KURO9]. It is important to ensure that TCP+FUZZY does not deviate from the behavior exhibited by normal TCP. Thus we seek to test our TCP+FUZZY against TCP by running multiple connections over a bottleneck and measuring the throughput and bandwidth for both types of TCP implementation. Ideally TCP+FUZZY will outperform other TCPs in terms of goodput, but both implementations should grab an equal share of the bottleneck's bandwidth. If TCP+FUZZY shares throughput fairly then there should be very little difference, in terms of individual TCP flow throughput, of a TCP/TCP pairing and a TCP/TCP+FUZZY pairing.

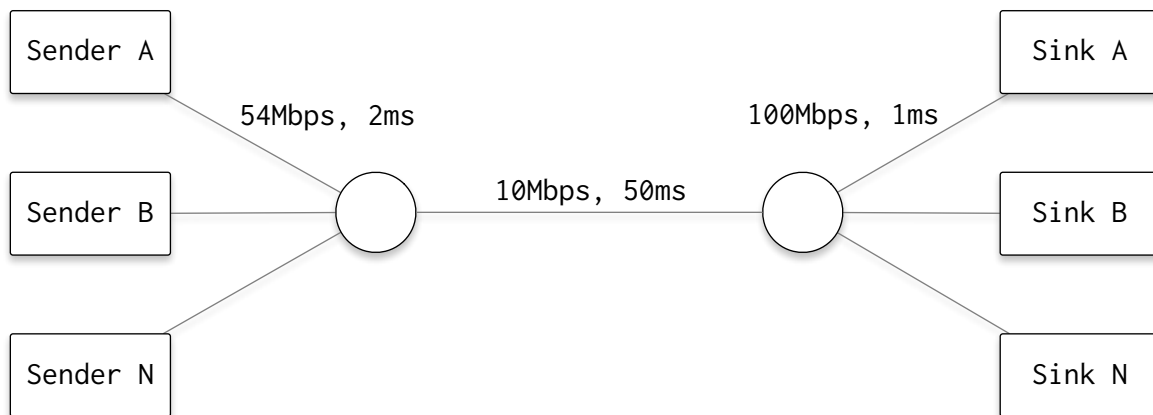


Figure 6: ns-3 simulation topology

The proposed fuzzy inference system classifies packet losses as either caused by congestion (C) or a link error (LE). Multiple inputs are received and generates an output, a member of {C, LE}. There are two types of possible errors in this classification scheme to be concerned with: a link error misclassified as a congestion error ( $Err_{LE}$ ), and a congestion error misclassified as a link error ( $Err_C$ ). Padhye et al. developed a stochastic model that models TCP throughput as a function of the send rate and the round trip time (RTT). Extending this model in their work on developing a machine learning classifier for TCP, [ELK08] analytically determined the maximum misclassification of congestion losses that can be incurred while still remaining TCP-friendly. In order to maintain TCP friendliness across both wired and wireless networks,  $Err_C$  must be kept below 18% [ELK08]. Because there is an inverse tradeoff between  $Err_C$  and  $Err_{LE}$  (meaning that  $Err_C$  can be decreased at the expense of an increase in  $Err_{LE}$ , and vice versa), the goal for this inference system is to minimize  $Err_{LE}$  while maintaining  $Err_C$  below the TCP-friendliness threshold.

## 4.3. The Fuzzy Inference Rule Base

The two fuzzy variable inputs into the TCP+FUZZY inference engine are the inter-arrival times and the one-way delays for outgoing packets. Before we discuss how the TCP+FUZZY makes use of these variables, it is worthwhile to discuss how they are initially collected. The TCP timestamp option [RFC1323] defines a means of sending packet departure and arrival information between sockets. It adds two new fields in the TCP options block: a timestamp and an echo reply timestamp. The timestamp field is stamped with the current clock time of the sending socket on each outgoing packet. The echo reply field is used to echo

timestamps back to their originating socket. Using this information we can calculate inter-arrival times and one-way delays.

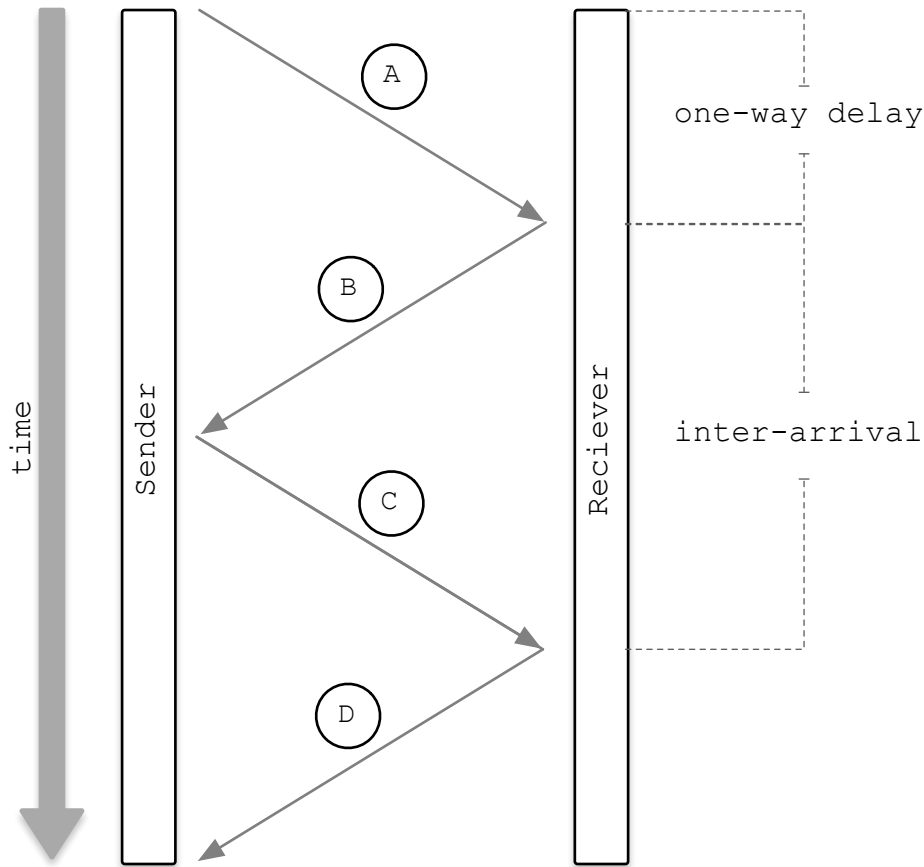


Figure 7: Inter-arrival and delay calculation.

Figure 7 depicts two sockets communicating. For convenience, each packet has been named with a letter. To calculate the delay we use the following equation:

$$\text{delay} = \text{B.echo} - \text{B.timestamp} \quad (3)$$

Notice that if the clock cycles are not synchronized between the sockets this will yield a non-accurate measurement. This can be ignored because it is only the relative measurements that we will be concerned with. To calculate the inter-arrival time:

$$\text{inter\_arrival} = \text{D.timestamp} - \text{Sender.timeToEcho} \quad (4)$$

where `Sender.timeToEcho` is the previous timestamp received from Receiver (equivalent in this example to `B.timestamp`).

We have defined two linguistic terms (*increasing*, and *stable*) for describing the variances of the inter-arrival times. The variances are good indicators that there is some atypical congestion condition on the network. It stands to reason that a drop due to a link error should have little correlation to the variances experienced prior to that drop. However a drop due to congestion may have indeed been foreshadowed by

fluctuating inter-arrival variances. To calculate the variance, we store the last two round-trip time's worth of inter-arrival times. When a loss occurs, the ratios for the variances of the previous two round-trip times are compared against each other. A ratio less than one suggests that the variance has increased in the last couple  $RTT$ s. And intuitively an increase in variance suggests congestion on the network. The inter-arrival input is then defined as the variance of the second-to-last  $RTT$  over the last  $RTT$ . If the variance is decreasing we can only assume that the network is stabilizing, perhaps indicating that congestion is not the root cause of a lost packet.

The one-way network delays (from the sender to the receiver) we expect to be roughly on par with the estimated  $RTT$  ( $eRTT$ ) divided by two. In some situations where congestion is experienced, the delays experienced on one side of the network will not be equal to the delays going the other way. We use this intuition as a metric for gauging loss causes. The second input is then defined as the ratio of the measured delay to the estimated  $RTT$ . The closer that ratio is to 1, the more stable the network is, which equates to a lower probability of congestion. The ratio may be larger or smaller than one so we define the input as

$$\text{Abs}(1 - 2 * \text{delay} / eRTT) \quad (5)$$

one or more fuzzy membership functions (high, medium, and low).

The following table represents the entire fuzzy rule base. Keeping the rule base as small as possible keeps it easier to comprehend and reduces the processing power needed to run the inference.

*Table 1: Fuzzy rule set for the TCP packet loss classifier.*

Inter-arrival Variance	One-way delay	Loss Cause
Incrementing	High	C
Incrementing	Medium	C
Stable	High	C
Stable	Medium	LE
Incrementing	Low	C
Stable	Low	LE

When the two metrics conflict TCP+Fuzzy plays conservatively and always falls back on determining congestion as the packet-loss cause.

## 5. Simulation

To simulate TCP+FUZZY, a dumbbell simulation was written to expose a set of sending nodes to a variety of network conditions. A set of  $n$  nodes are each connected to an single node from another  $n$ -sized set, and they are separated by a bottleneck that is susceptible to network congestion. Each of the sending nodes is tasked with transmitting data in a randomly on-off pattern to its receiver node. The rate at which they attempt to send is based on the bandwidth of the bottleneck divided by the number of node pairs.

A variety of tests were performed under these circumstances. The trials were run where all TCP sockets were of the same type (e.g. all TCP NEW RENO or TCP+FUZZY) and the average throughput and goodput for the nodes was recorded. Then the trials were performed where a variety of TCP implementations were used, and the average throughput and goodput were recorded for each TCP type. We contrast the results in the next section.

Finally, one very important metric is TCP-friendliness, which for our TCP+FUZZY classifier can be measured by our classification (or misclassification) rate. As discussed earlier, we must misclassify congestion losses at no greater than 18% or we risk compromising TCP-friendliness. Throughout each trial our ns3 network simulation omnipotently determines the cause of each loss and compares it to what the fuzzy inference engine in TCP+FUZZY predicted it to be.

Figure 8: Throughput & goodput ( $t=100s$ )

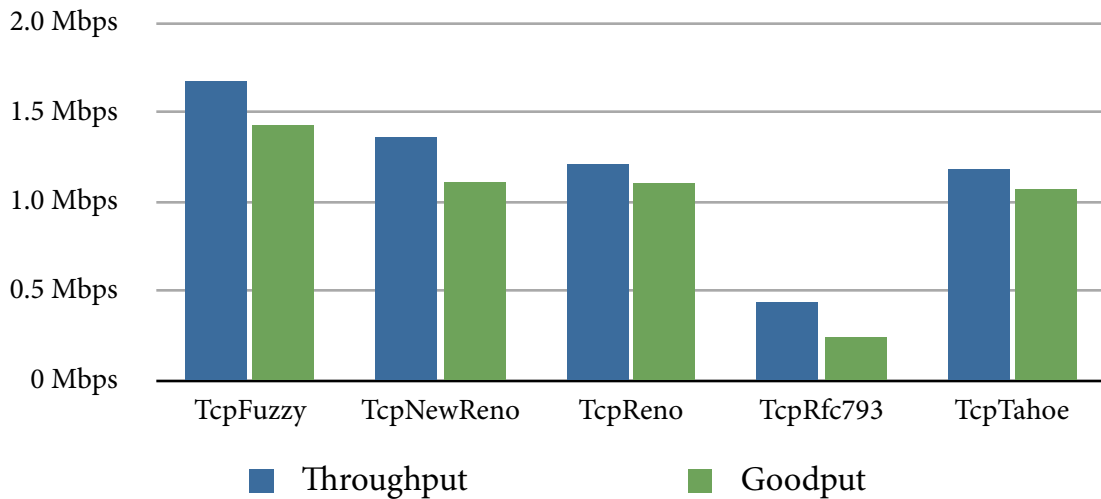


Figure 9: Throughput & goodput ( $t=250s$ )

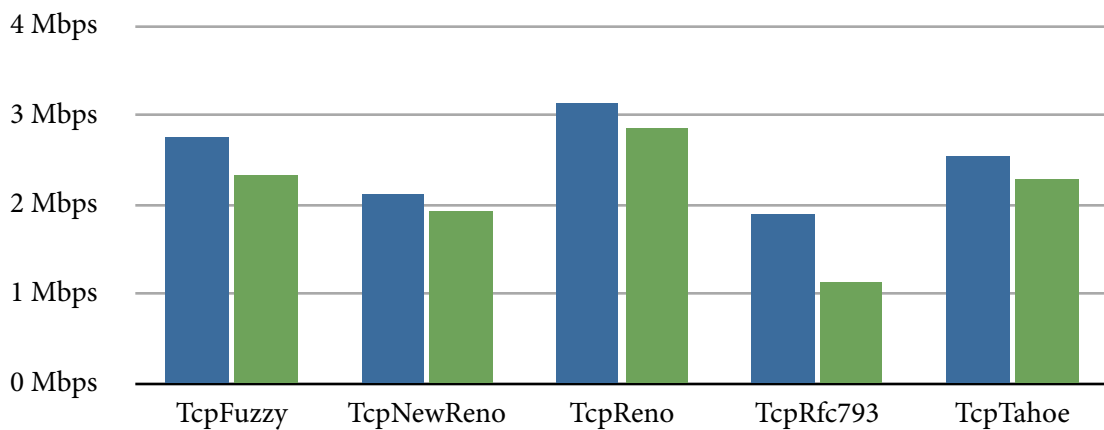
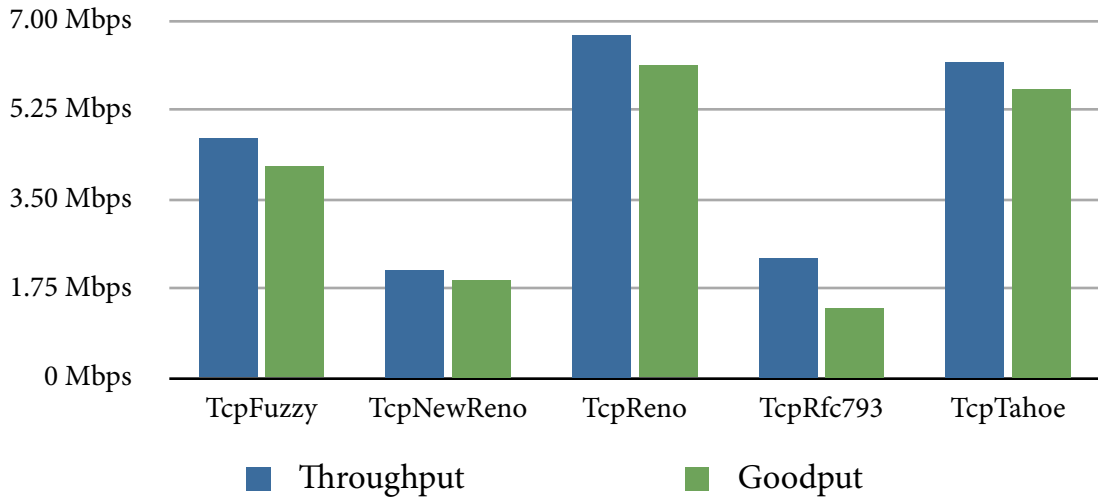


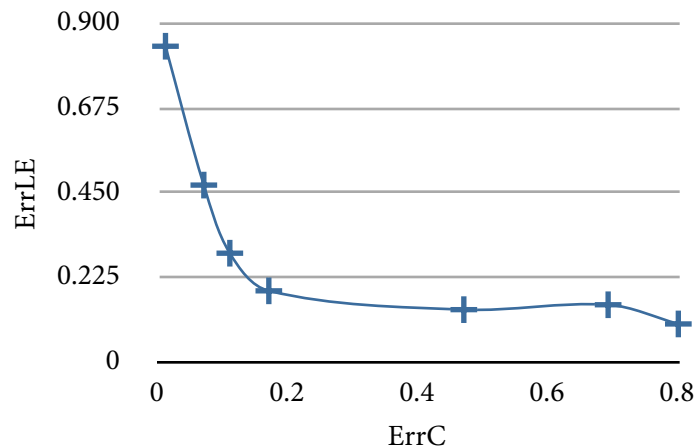
Figure 10: Throughput & goodput ( $t=500s$ )



These graphs were generated by running the simulation and recording the average goodput and throughput for the TCP socket types. Results indicate that our TCP+FUZZY inference system succeeds in increasing goodput over other TCP implementations for smaller transmission times. As the transmission times increase (moving upwards of 500 seconds) both Tahoe and Reno become more effective than TCP+FUZZY, TCP new reno and rfc793.

We now demonstrate that TCP+FUZZY maintains TCP friendliness. As the inference engine was tweaked for force misclassifications that raised  $Err_{LE}$  it is observed that TCP+FUZZY maintains TCP-friendly values of  $Err_C$  for reasonable values of  $Err_{LE}$ . In all of the trials  $Err_C$  was held at or below 19%. In the vast majority of the cases, though,  $Err_C$  did not climb higher than 18%.

Figure 11:  $Err_C$  vs  $Err_{LE}$



## 6. Conclusion

Using a fuzzy inference engine to calculate TCP packet losses can lead to an increase in good-put over busy or congestion-prone wireless networks. The inference relies on the fact that the inter-arrival times and the one-way delays of the sender are good indicators of network congestion. By carefully choosing the right metrics the problem can be described using intuitive if then statements which can be trivially fed into a fuzzy inference engine. The advantage of the TCP-FUZZY is that it can be appended to any existing TCP protocol without modification to any of TCP's other behaviors.

## 7. References

[APS99] "Allman M., Paxson V. & Stevens, W. TCP Congestion Control Request for Comments 2581, Network Working Group <<http://tools.ietf.org/rfc/rfc2581.txt>> April, 1999. (Accessed 8 Oct. 2010).

[BAK97] "B.S. Bakshi, P. Krishna, D.K. Pradhan and N.H. Vaidya, Improving performance of TCP over wireless networks, in: International Conference on Distributed Computing Systems (May 1997).

[BAL97] "H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. IEEE/ ACM Transactions on Networking, 1997.

[BAL98] "Balakrishnan, Hari, and Randy H. Katz. Proc. IEEE Globecom Internet Mini-Conference. Australia, Sydney. Computer Science Division, Department of EECS, University of California at Berkeley, Nov. 1998. Web. 10 Oct. 2010.

[BER08] "Bergmann, Merrie. An Introduction to Many-valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems. Cambridge: Cambridge UP, 2008. Print.

[CAL03] "Calzarossa, V.G.: A Fuzzy Algorithm for Web Caching. Simulation Series Journal 35(4), 630–636 (2003)

[CHAO1] "K. Chandran, S. Raghunathan, S.R. Prakash, A feedback-based scheme for improving TCP performance in ad hoc wireless networks, IEEE Personal Communications 8 (1) (2001) 34–39.

[CHE00] "Chen, G., and Trung Tat. Pham. Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems. Boca Raton, FL: CRC, 2001. Print.[Chr07] "C. Chrysostomou, A. Pitsillides, G. Hadjipollas, A. Sekercioglu, M. Polycarpou, "Fuzzy Logic Congestion Control in TCP/IP Best-Effort Networks", 2003 Australian Telecommunications Networks and Applications Conference (ATNAC 2003), Melbourne, Australia, 8 - 10 December 2003 (CD ROM - ISBN: 0-646-42229-4).

[GER99] "M. Gerla, K. Tang and R. Bagrodia, TCP performance in wireless multi-hop networks, in: Proceedings of IEEE WMCSA'99, New Orleans, L A (February"1999). [Goe03] "Goebel, Greg. "An Introduction To Fuzzy Control Systems." Internet FAQ Archives - Online Education - Faqs.org. June 2003. Web. 10 Oct. 2010. <<http://www.faqs.org/docs/fuzzy/>>. [ElK08] "El Khayat, I., Geurts, P., and Leduc, G. 2010. Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning. Wirel. Netw. 16, 2 (Feb. 2010), 273-290. DOI=<http://dx.doi.org/10.1007/s11276-008-0129-y>

[KHA06] "S. Khajouejinejad, M. Sabeghi, and A. Sadeghzadeh, "A Fuzzy Cache Replacement Policy and Its Experimental Performance Assessment," in Innovations in Information Technology, Dubai, May 2006, pp. 1-5.

[MAS01] "Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, and Ren Wang. 2001. TCP Westwood: Bandwidth estimation for enhanced transport over wireless links. In Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01). ACM, New York, NY, USA, 287-297.

[MOR03] "B. Moraru, F. Copaciu, G. Lazar and V. Dobrota. "Practical Analysis of TCP Implementations: Tahoe, Reno, NewReno". Proceedings of RoEduNet International Conference: Networking in Education and Research. 2003. pp. 125-130.

[NS3] The Ns-3 Network Simulator. 17 Apr. 2011. Web. 17 Apr. 2011. <<http://www.nsnam.org/>>.

[PAR00] Parsa, Christina, and J.J. Garcia-Luna-Aceves. "Improving TCP Performance over Wireless Networks at the Link Layer." *Mobile Networks and Applications* 5 (2000): 57-71. Print.

[RFC793] "RFC 793 - Transmission Control Protocol (RFC793)." Faqs.org. University of Southern California, Information Sciences Institute, Sept. 1981. Web. 22 Nov. 2010.  
<<http://www.faqs.org/rfcs/rfc793.html>>.

[RFC2581] "Allman, M., V. Paxson, and W. "RFC 2581 - TCP Congestion Control (RFC2581)." Faqs.org. Apr. 1999. Web. 22 Nov. 2010. <<http://www.faqs.org/rfcs/rfc2581.html>>.

[ROS10] "Ross, Timothy J. *Fuzzy Logic with Engineering Applications*. Chichester, U.K.: John Wiley, 2010. Print.

[SAB06] "Sabeghil, M. and Yaghmaee, M. H. 2006. Using fuzzy logic to improve cache replacement decisions. *IJCSNS International Journal of Computer Science and Network Security*, Seoul, v.6, n.3A, 182-188.

[SIK01] B. Sikdar, S. Kalyanaraman, and K. Vastola. Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK. In *Proc. of the IEEE GLOBECOM*, pages 25–29, 2001.

[SHI09] Shi, Kai, Yantai Shu, and Qingfeng Song. "Receiver Centric Fuzzy Logic Congestion Control for TCP Throughput Improvement over Wireless Networks." *Proc. of International Conference on Fuzzy Systems and Knowledge Discovery*, Tianjin, China. Web. 16 Apr. 2011.  
<<http://www.computer.org/portal/web/csd1/doi/10.1109/FSKD.2009.835>>.

[WAX88] "Waxman, B. M. "Routing of Multipoint Connections." *IEEE Journal on Selected Areas in Communications* 6.9 (1988): 1617-622.

[XYLO1] "G. Xylomenos, G.C. Polyzos, P. Mahonen, M. Saaranen, TCP performance issues over wireless links, *IEEE Communications Magazine* 39 (4) (2001) 52–58.

[ZHAO5] “Zhang, Zhongwei and Li, Zhi and Suthaharan, Shan (2005) Fuzzy logic strategy of prognosticating TCP's timeout and retransmission. In: Halgamuge, Saman K. and Wang, Lipo, (eds.) Computational intelligence for modeling and prediction. Studies in Computational Intelligence (2). Springer-Verlag, Berlin. ISBN 3540260714