

USING FUZZY INFERENCE TO IMPROVE TCP
CONGESTION CONTROL OVER WIRELESS NETWORKS

By

MATTHEW WOZNIAK

Advisor

DR. HALA ELAARAG

A senior research proposal submitted in partial
fulfillment of the requirements for the degree of Bachelor
of Science in the Department of Mathematics and Computer
Science in the College of Arts and Science at
Stetson University DeLand, Florida

FALL TERM

2010

ABSTRACT

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.

1. Introduction	4
2. Related Work	5
3. Background Information	8
3.1. TCP Congestion Control and Wireless Networks	9
3.2. Fuzzy Inference	10
4. TCP+Classifier	15
4.1. Packet Loss Events and Classifier Inputs	15
4.2. Loss Classification Utilizing Fuzzy Inference	18
4.3. The Fuzzy Inference Rule Base	19
5. Simulation	21
6. Conclusion	22
7. References	24

1. INTRODUCTION

TCP traffic makes up about 90% of all Internet [Shu09] making it the most widely utilized reliable transport protocol. TCP performs poorly over wireless links because its congestion control algorithm treats all packet losses equally. On wireless links, packet losses are likely to be caused by link errors—due to physical characteristics that may involve noise or spotty radio connections—as opposed to congestion. This negatively impacts TCP throughput over that channel because TCP invariably reduces its bandwidth in the event of a packet loss in an attempt to curb the perceived congestion. Specifically, in the most commonly used TCP implementations, a dropped packet will cause the sending TCP server to cut the slow start threshold ($ssthresh$) value by at least two full segment sizes (or more commonly $ssthresh$ may be reduced to half of the congestion window ($cwnd$)) [APS99, Mor03]. Because maximum throughput occurs when this congestion window is large [Par00], it is of interest to prevent TCP from reacting to link errors in the same way that it reacts to congestion.

One possible means of enabling TCP to react selectively to link-error losses is to endow TCP with a packet loss classifier that can infer the cause of an individual loss. Ideally (for reasons discussed below), the classifier should only make use of information directly available to the TCP sender and receiver. This paper proposes the creation of a fuzzy inference system that can classify packet losses. The fuzzy classifier can then be attached to either the sending or receiving TCP implementation.

To develop the fuzzy inference system, network conditions are simulated using ns-3. This approach is ideal because of the difficulty in characterizing packet losses on actual networks [EIK08]. Our simulation generates random network topologies so as to reduce

bias for or against a specific type of topology. From those topologies a database of packet losses is recorded for use in testing the classifier. With the database generated, a set of fuzzy inference rules is created and modified so as to maximize the accuracy and minimize error. Padhye et al. have demonstrated [Pad00] analytically that TCP throughput as a function of error probability can be accurately modeled. El Khayat et al. have extended this model to be applicable to TCP implementations endowed with classifiers [EIK08].

The rest of this paper is organized as follows: in Section 2 we cover past attempts at solving the problem of inefficient TCP utilization of wireless networks. In Section 3 we discuss concepts and background needed to understand this paper's proposal. Section 4 presents our proposal for a TCP+classifier in detail, including specific information on the simulation and fuzzy inference rules used in our classifier. Finally the Section 5 reviews this proposal and its potential impact on TCP wireless performance.

2. RELATED WORK

There have been many attempts to rectify TCP's suboptimal performance over wireless networks [Mas01] [Xyl02] [Cha02] [Ger99] [Bak97]. To cope with this issue, many solutions—varied in design, scope and application domain—have been conceived. Balkrishnan et al. outline many of the solutions, including “reliable” link-layer protocols (link-layer protocols endowed with forward error correction (FEC) or retransmissions (ARQ)), split-connection protocols, and selective acknowledgments [Bal97, Bal98]. This approach, along with breaking the OSI network model abstraction, is unproven to play nicely with TCP implementations that expect the underlying link layer to be of the best effort variety. Split-connection protocols involve a base station breaking the TCP stream

into two pieces such that the client talks to the base station, which in turn maintains a separate TCP connection with the server [Bal97, Bal98]. [Par00] outlines the drawbacks to this approach. 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]. TCP SACK also suffers from the same issue as normal TCP in that losses are always interpreted as congestion. [Cla02] proposes TCP Westwood, an 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.

An ideal solution to the TCP/wireless problem would be one that could be implemented entirely inside a TCP state machine. This would ensure that only the sender would need to be endowed with the packet-loss classification logic; every other node on the link may simply speak normal TCP. 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).

El Khayat et al. [EIK08] worked with applying machine learning (ML) towards improving TCP over wireless connections. The idea is to use data available inside of the state machine as variables to feed into a ML algorithm that powers a packet loss classifier. This classifier distinguishes between packet loss events caused by link errors (`LE`) and those caused by congestion (`C`). By lowering the number of registered `LE` on wireless

networks (and simultaneously maintaining the correct c for TCP-friendliness on wired networks), TCP performance can be improved over wireless links without impact on non-wireless channels.

Fuzzy control systems have been utilized in many networking-related areas. [Zha05] used fuzzy control systems to improve TCP packet loss retransmission (RTO) times. [Sab06], [Cal03] and [Kha07] worked to improve cache replacement algorithms by incorporation fuzzy logic. [Pav05] proposes a new streaming media protocol that makes use of fuzzy controllers for congestion control. [Chr07] uses fuzzy logic on the IP layer to implement an intuitive, customizable approach for active queue management. These approaches demonstrate the applicability of fuzzy logic for problems that may be difficult to statistically analyze. Fuzzy inference systems also allow problems to be defined in an intuitive way using a propositional IF-THEN rule base (which we will discuss later in this paper).

[Shu09] have developed a fuzzy controller that reacts to changes in bandwidth utilization to continually adjust the congestion window. Their models indicate a performance increase in TCP endowed with the fuzzy controller over TCP NewReno and Westwood. Their proposal requires a modification of TCP's standard packet loss response. Our proposal differs in that we define a classifier that filters packet losses from TCP, but does not alter any other aspect of the TCP loss event behavior. See Figure 1 below for a visualization:

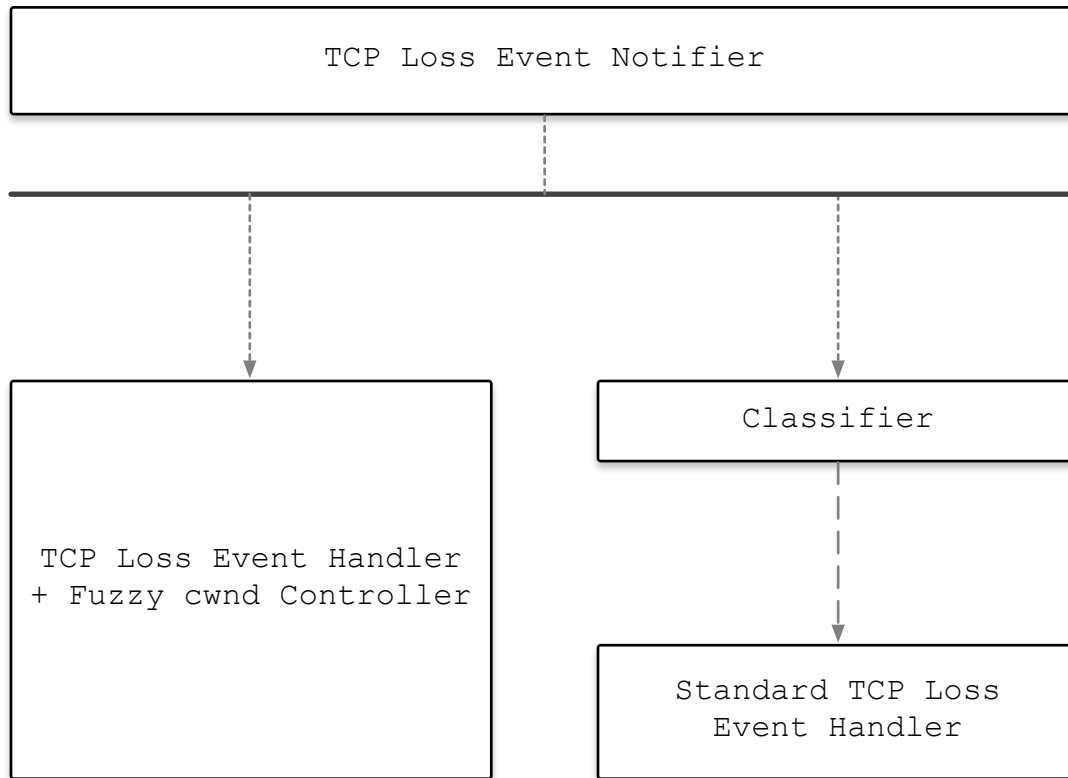


Figure 1: Different approaches to handling TCP loss events. On the left, [Shu09]’s method, which requires modification to the TCP loss event handler (dynamically modifying the `cwnd`). On the right, our proposed method which filters loss events from the standard event handler.

The dotted lines represent loss events generated from either timeouts or duplicate ACKs. Our classifier filters a subset of these incoming events from the TCP loss event handlers (specifically the loss events it deems to be caused by link-errors). An advantage of using a filtering classifier is that as long as classification errors are kept below certain values, we can maintain TCP friendliness over non-wireless links (a statement which we verify later). In essence, our proposal is to take the speed, customizability and understandability of fuzzy controllers and fuzzy inference systems and implement the TCP+classifier proposed by [EIK08] (which made use ML algorithms).

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. TCP is a large and complex protocol, and this paper assumes familiarity with TCP's sliding window and congestion control mechanisms and refers readers to [RFC2581] and [RFC793] for more information. As a TCP connection persists, packets are sent from a sender (S) to a receiver (R). With every packet that is successfully acknowledged by R, S's congestion window ($cwnd$) is increased (in principle) by one full segment size. At the point at which a loss is realized by S (perhaps by three duplicate ACKs or a transmission timeout), the slow start threshold ($ssthresh$) is halved and $cwnd$ is reset to one full segment size. Note that TCP Reno is capable of entering into Fast Recovery mode, but if more than a single packet has been lost, the end result for $cwnd$ and $ssthresh$ will be the same.

This conservative approach works well for packet losses that are a result of congestion on the network (perhaps a router or other node experienced a buffer overflow). 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 case where a packet is dropped because of a link-error over a wireless connection: in this situation it is most certainly not intuitive to react by reducing the rate of outgoing traffic as there is no congestion that needs considering. Worse, the reduction in bandwidth will only serve to reduce throughput, slowing speeds down for the user. Every time a non-consecutive packet loss occurs, $cwnd$ is reset and $ssthresh$ is recalculated based on the following formula:

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

Where FlightSize is the amount of outstanding data in the network and SMSS is the sender maximum segment size. FlightSize can often be approximated by $cwnd$. One can think of $sssthresh$ 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 are introduced, the probability of which is independent of congestion, this conceptual model must 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 like interleaving or forward error correction (FEC) which are often found on cellular networks [Gur03]. 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 [Gur03] after which it simply gives up. Poor radio conditions and mobility are some of the various causes for corruption errors on wireless links. Note that link-layer retransmission are not ideal because the network conditions are obfuscated from the TCP implementation. For example, if a packet is retransmitted at the link layer several times, its RTT will stray significantly from the average. It also may be possible that TCP retransmits the packet by the time the link layer correctly delivers the original.

3.2. Fuzzy Inference

Presented now is a brief overview of fuzzy logic and fuzzy inference systems, which will be used to classify packet losses as either C or LE . For theoretical coverage of multi-valued logic and fuzzy set theory we recommend the reader to [Ber08]. For background in fuzzy control systems and fuzzy inference, please see [Che00] and [Ros10].

While traditional logic contains only two truth values (true and false), fuzzy logic may contain an infinite number of truth values on the continuous range $[0,1]$. This allows for the definition of fuzzy membership functions with which values in a universe may be partial members.

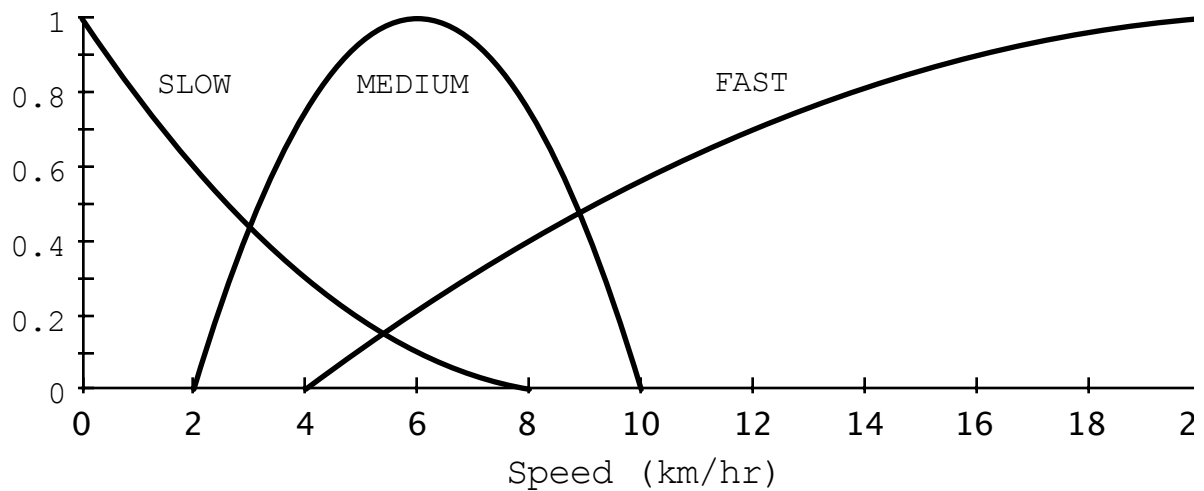


Figure 2: Fuzzy membership functions slow, medium, and fast for the domain speed

In Figure 2 you can see three fuzzy membership functions, *slow*, *medium*, and *fast* that are mapped on the domain of a range of possible speeds (in kilometers per hour). These membership functions represent the extent that a given value on the domain “belongs” to that linguistic descriptor. For example, a speed of 15 km/hr might be 0.65 fast, and 0.25 medium. Note that it is not necessary for each values for an x on the domain to add up to one. The membership values can sum to any number, greater than or less than one. These linguistic descriptors are completely subjective in definition and tailored to the problem with which they are being applied. Thus if these functions were applied to a problem dealing with race cars, the Fast membership function might have membership values on the order of hundreds of kilometers an hour. A problem centered around

marathon runners might have definitions mapping to speeds on the order of tens of kilometers an hour. Ross [Ros10] 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 with manual fine-tuning to define and optimize the membership functions.

Once membership functions have been defined, a fuzzy rule-based system must also be defined. A fuzzy rule-based system is nothing more than a series of implications consisting of a variable number of antecedents and consequents. These rules are defined in terms of natural language descriptors so that human knowledge and intuition can be applied towards the problem. Consider the problem of modeling the average speed of a marathon runner as a function of the environmental variables “temperature” and “humidity”. First, inference rules must be generated that accurately 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
- ...
- r. 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):

$$\text{IF } X_1 \text{ is } A_1^k \text{ and } X_2 \text{ is } A_2^k \text{ and } \dots \text{ and } X_n \text{ is } A_n^k \text{ THEN } y^k \text{ for } k = 1, 2, \dots, r$$

equation (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 [Ros10] [Che00], a commonly used and computationally cheap inference method

traditionally utilized in fuzzy control systems. The x_1 through x_n are inputs, while y^k are outputs. Graphically, Mamdani max-min inference can be depicted as follows:

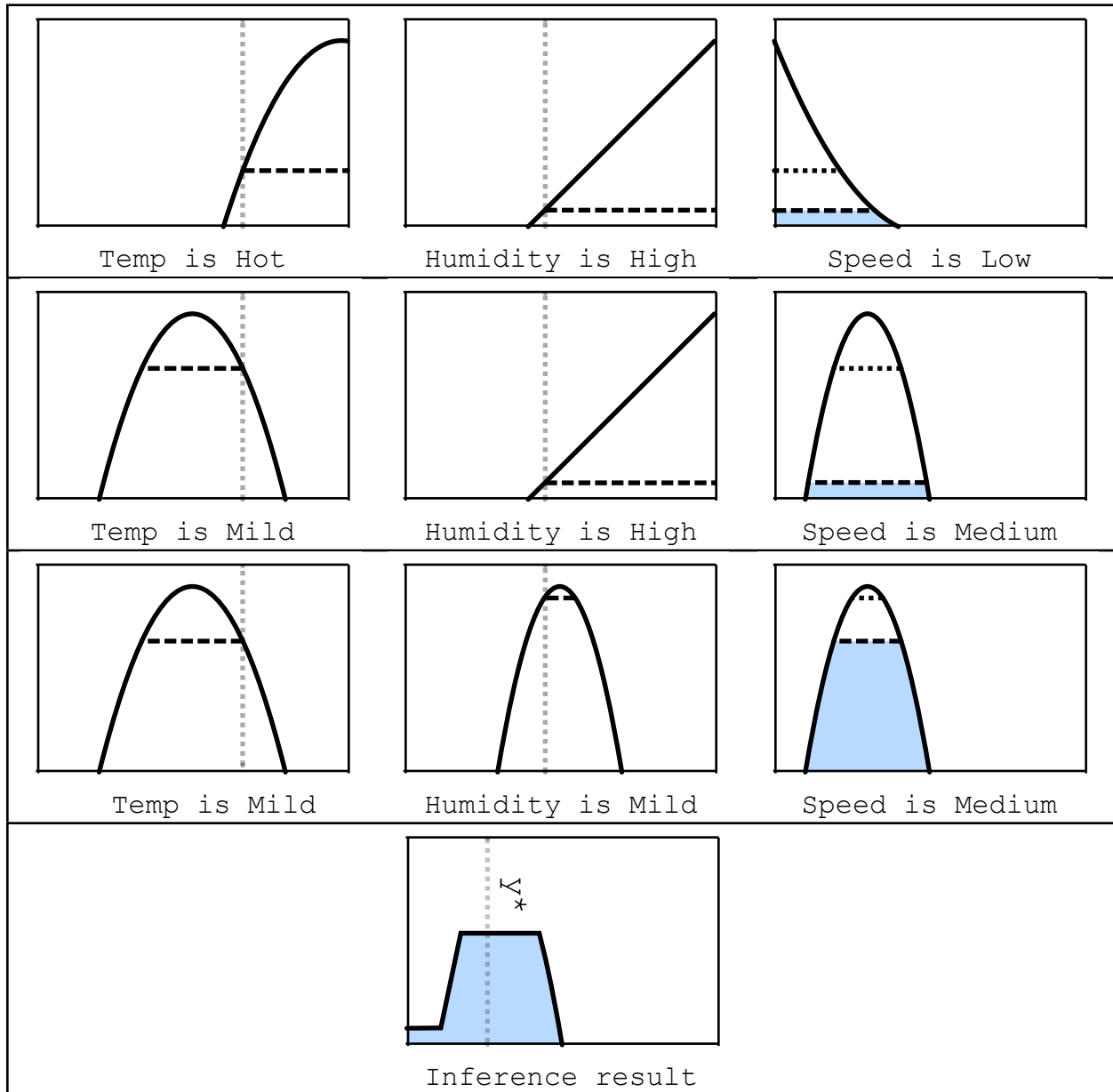


Figure 3: Graphical Mamdani max-min inference (only the first three rules are displayed)

Figure 3 depicts a Mamdani inference system, with each horizontal series of graphs corresponding to the rule sets listed above. Inputs to the system can be crisp values that take the form of delta functions. You can see here that the delta function is repre-

sented 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 left of each inference rule). Finally, each fuzzy set produced is aggregated using either conjunction or disjunction (for max-min inference conjunction is used), producing the final inference result 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”.

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:

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

The centroid method can conceptually be thought of as the “center of gravity”. Defuzzification will yield the single most representative value of the fuzzy set (dependent, of course, on the defuzzification method used). In Figure 3, y^* is indicated by a dotted vertical line passing through the inferred fuzzy set. The whole inference rule set might be read as “based on the temperature and humidity present, we can expect runners to

average slightly less than normal speed.” This result is intuitive if you agree that humidity and heat can have a detrimental effect on a runner’s performance. Note that fuzzy inference is only as good as the rules provided. If the first rule was redefined as `IF Temp is High and Humidity is High THEN Speed is High`, then the fuzzy inference system would return a mathematically correct result, but one that might not reflect real world outcomes. It is of great importance that the inference rules accurately model the problem domain or the inference results will not be of any use.

4. TCP+CLASSIFIER

Presented here is our proposal for a packet loss classifier built on fuzzy inference technology. The classifier can be built into any TCP implementation given that it meets certain constraints that are discussed in section 4.3.

4.1. Packet Loss Events and Classifier Inputs

The proposed classifier is a multi-input, single-output function. The inputs are all available inside of the sender’s TCP state machine. Before we discuss the choice of inputs, it is important to note TCP’s slow start and congestion avoidance behavior.

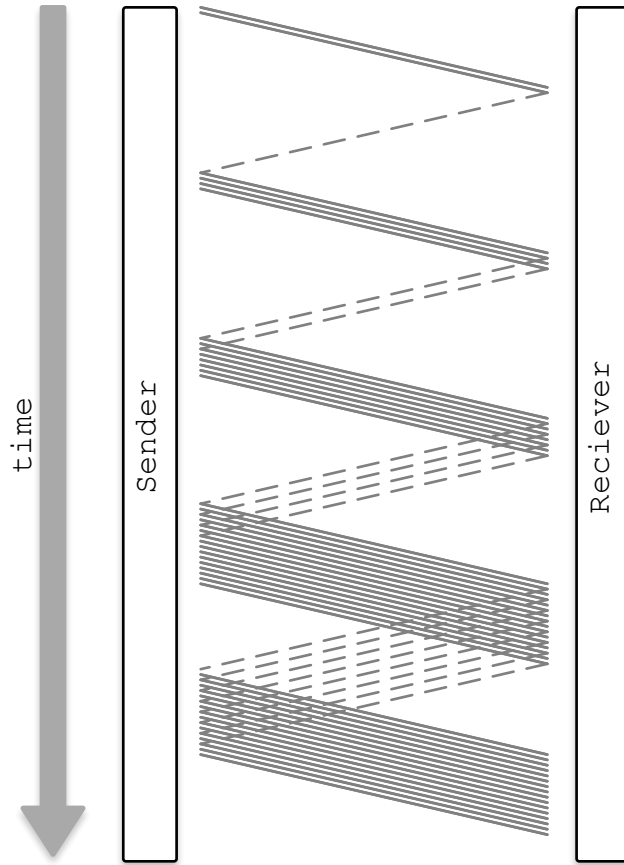


Figure 4: A simplified diagram of TCP slow start

As Figure 4 shows, TCP grows exponentially during its slow start phase. This diagram is greatly simplified, but by the 5th round trip time is reached TCP has exited slow start and entered congestion avoidance. When a packet loss occurs in most TCP implementations (excluding Westwood), the slow start threshold is reduced (see section 3.1). In Figure 4 no packet losses have occurred and TCP transmits data without error.

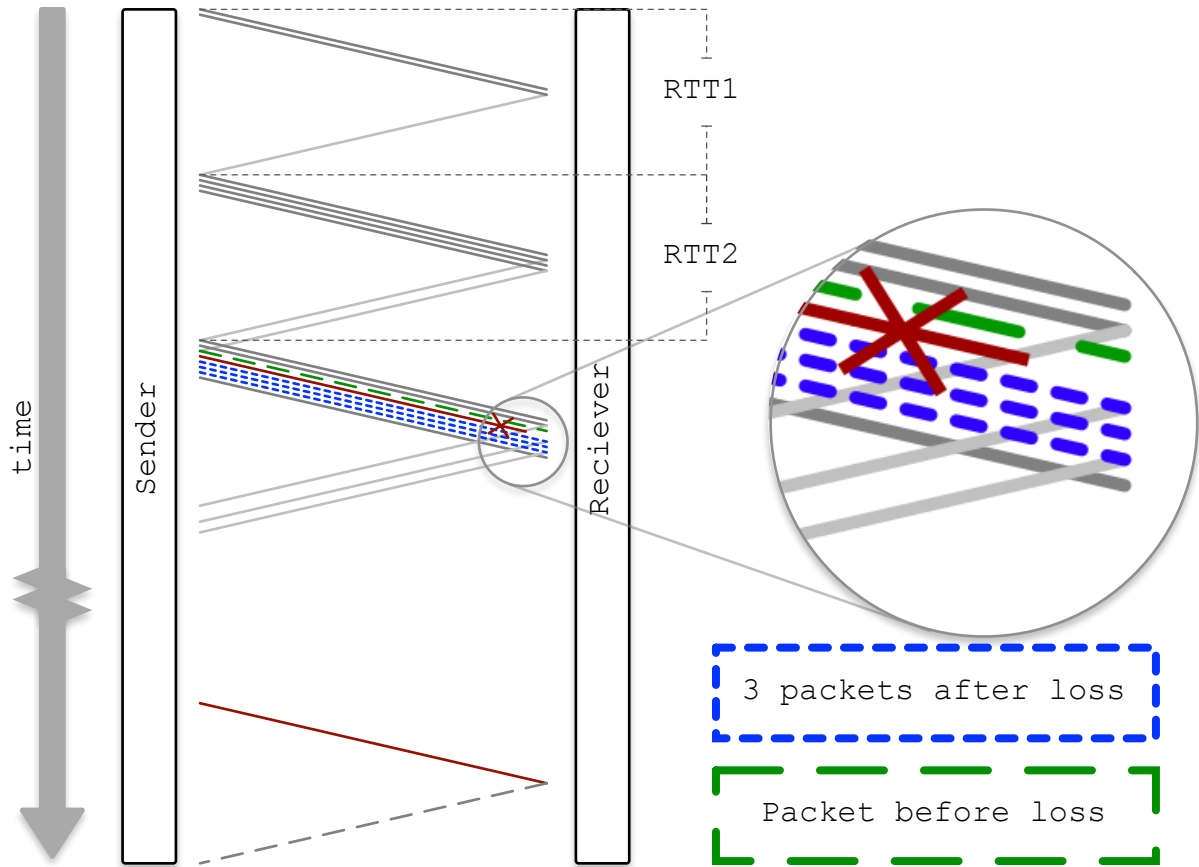


Figure 3: Classifier inputs recorded after a TCP loss event

In Figure 5 we see that TCP has suffered a loss on its way through a slow start. We note that before the sender can realize a loss has occurred, the retransmission timer for the lost segment must fire. The lost segment is denoted in red. 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. As inputs for the packet loss classifier, the average, standard deviation, minimum and maximum values are all computed for both the one-way delay and the inter-arrival times for packets sent during each of the last two round trip times. These eight values (four normalized values for the last RTT and four for the second to last RTT) are then related to the packet sent immediately before the loss (the green cut line in Figure 3) and the three packets sent after the loss (the dotted blue line

in Figure 3). Note that the values for the last two RTTs are calculated continuously and kept on hand in the event of a loss. Also note that the values calculated are *only* for packets sent out, not ACKs received. We can discern nothing about the condition of the network by looking at incoming ACKs [EIK08].

4.2. Loss Classification Utilizing Fuzzy Inference

Once the inference system is correctly classifying packet losses to an acceptable degree, it must be ensured that, in addition to outperforming TCP in terms of bandwidth, TCP+classifier does not hog more than it's fair share of network throughput. TCP congestion control constantly adjusts individual throughput such that throughput gets distributed equally across all TCP connections on a network [Kur09]. It is important to ensure that TCP+classifier does not deviate from the behavior exhibited by normal TCP. Thus we seek to test our TCP+classifier against TCP by running multiple connections over a bottleneck (see Figure 3) and measuring the throughput and bandwidth for both types of TCP implementation. Ideally TCP+classifier will outperform TCP in terms of bandwidth, but both implementations will grab an equal share of the bottleneck's throughput. If TCP+classifier share's throughput fairly, then there should be very little difference, in terms of per-connection throughput, of a TCP/TCP pairing and a TCP/TCP+classifier pairing.

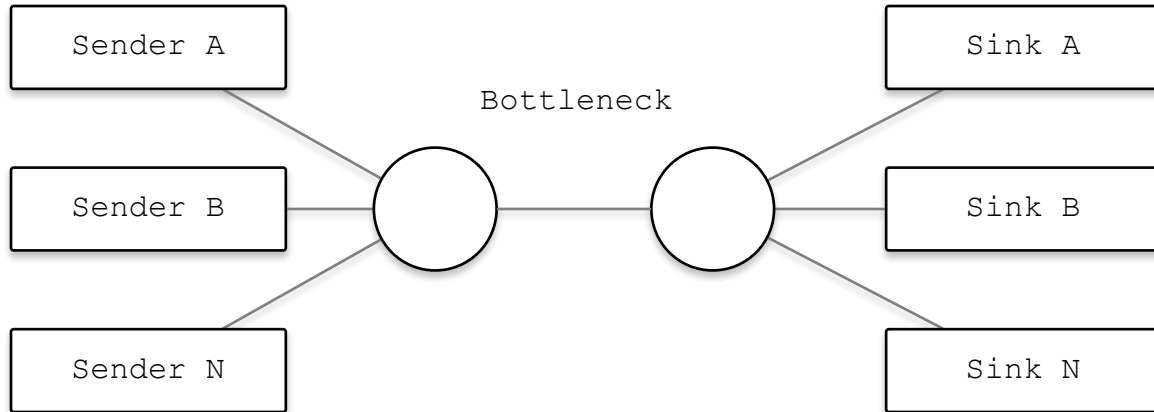


Figure 6: Comparing TCP+classifier to TCP.

The proposed fuzzy inference system will classify packet losses as either being 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). This model has been extended to cover situations involving TCP+classifier [EIK08]. It has been shown that in order to maintain TCP friendliness across both wired and wireless networks, Err_C must be kept below 18%. 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

From the inputs described in Section 4.1 we derive several rules based on intuition for our classifier. These rules were defined based on an intuitive understanding of TCP behaviors. However, intuition may not always be completely accurate, and as such

these rules are set to be reviewed after the initial results are gathered. At the end of this section we explore other possible methods for generating rule sets for the classifier.

Here are some examples of the rules that have gone into the fuzzy classifier. The full set is presented in Table 1. We have defined three linguistic variables for describing both inter-arrival times and one-way delays. These variables are “high variance”, “medium variance”, and “low variance”. If a loss event occurs due to a link error, we would not expect much variance in the inter-arrival times or the one way delay, because those are indicators of congestion. Thus, one of the rules in our rule set reads

IF Delay is Low AND InterArrival is Low THEN Error is LE

Conversely,

IF Delay is High AND InterArrival is High THEN Error is C

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

Delay	InterArrival	Error
High	High	C
High	Medium	C
Medium	Medium	C
Low	Medium	LE
Medium	Low	LE
Low	Low	LE

It is not necessary, in fact inadvisable, to define rules for every permutation of the linguistic variables. Doing so may only serve to muddle the understandability and accu-

racy of the inference engine. For instance, only under strange network conditions would a high variance be detected in the inter-arrival times and a low variance be detected in the one-way delay. To define an explicit rule for this situation (IF Delay is High AND InterArrival is Low) would complicate most other frequently occurring situations. The goal with a fuzzy inference system is to get the job done with as few rules as possible.

An advantage of fuzzy controllers is that changes to the rule base can be made very easily. In the future it might be worthwhile to explore the possibility of using machine learning algorithms to “train” the fuzzy controller. This approach might yield a more well defined rule base, yet still retain accessibility and interpretability (a characteristic often lost when using solely ML-based approaches).

5. SIMULATION

To create a database of packet losses, along with characteristic data to be used as inputs for the fuzzy inference system, ns-3 will be used to create sets of random network topologies. Ns-3 is our favored simulator because it is open source, has a thriving developer community (it is the successor of the widely-used ns-2), and it allows simulations to be written using modern software-development approaches. Ns-3 currently implements TCP Tahoe. When a reference is made to “normal” TCP, or just TCP sans classifier, the TCP implementation implied is Tahoe.

To create the random network topologies, packet losses are simulated on a network for a fixed amount of time. These losses are collected into a database, which will be broken into two parts: a subset of loss events for training the inference system, and a subset of loss events for analyzing and testing the inference system. A network of be-

tween 10 and 600 nodes is uniformly generated and distributed on a 100 square meter plane. The nodes are linked together randomly by applying an algorithm introduced by [Wax88]. Nodes will either be connected via a wireless link or a peer-to-peer link over ethernet. The probability that a generated link will be wireless is set at 0.5.

Each link has its bandwidth set between 56 Kbps and 100 Mbps. The propagation delay will randomly vary between 0.1 ms and 500 ms. Unless otherwise stated, all variances are uniform in nature. Sixty percent of all traffic generated is TCP, while the remainder is UDP. The the senders, receivers, and lengths of these connections are completely random. The motivation for such a random topology setup is to minimize bias favoring one topology or another. Because the simulation contains rather unlikely topologies, it can only harm, not help, the accuracy of the fuzzy classifier.

6. CONCLUSION

The proposal seeks to endow TCP with a classifier that can infer the cause of a loss event. The purpose of such a classifier is to prohibit TCP from cutting it's cwnd and unnecessarily reducing throughput over a wireless link. Our approach to make use of fuzzy inference arises because of the speed and ease of understandability that these inference systems offer. To build up a set of inference rules, loss events are generated using ns-3 and analyzed for patterns relating normalized values for the one-way delay and inter-arrival times of various groups of packets surrounding a loss event to the cause of that particular loss event. These rules are combined in the fuzzy inference system using a max-min approach. The TCP+classifier should not have an Err_c greater than 18% to ensure TCP friendliness, and should perform better over wireless networks

than standard TCP implementations, yielding an improvement in bandwidth over radio links.

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)
- [Cha01] 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/>>.
- [EIK08] 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.
- [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.
- [Wax88] Waxman, B. M. "Routing of Multipoint Connections." IEEE Journal on Selected Areas in Communications 6.9 (1988): 1617-622.
- [Xyl01] G. Xylomenos, G.C. Polyzos, P. Mahonen, M. Saaranen, TCP performance issues over wireless links, IEEE Communications Magazine 39 (4) (2001) 52-58.
- [Zha05] 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 pre-

diction. Studies in Computational Intelligence (2). Springer-Verlag, Berlin. ISBN
3540260714