State Transition Model to Characterize TCP Window Control Behavior in Wired/Wireless Internetworks*

Debashis Saha[1], Amitava Mukherjee[2] and Sanjay K Jha[3]

September 2003

UNSW-CSE-TR-0328

* This work is supported by the faculty research grant on the project "improving tcp performance in 3G/4G mobile wireless networks".

[1] MIS & Computer Science Group, Indian Institute of Management (IIM) Calcutta, Joka, D. H. Road, Kolkata 700 104, India. E-mail: ds@iimcal.ac.in [*He did this work when he visited School of Computer Science & Engg, University of NSW, Australia during September 2003*.]

[2] School of Computer Science & Engineering, University of NSW, Sydney 2052, Australia
E-mail: amitavam@cse.unsw.edu.au

[3] School of Computer Science & Engineering, University of NSW, Sydney 2052, Australia
E-mail: sjha@cse.unsw.edu.au

Abstract

TCP was designed to work well in networks with low channel error rates. Wireless networks on the other hand are characterized by frequent transmission losses. As a result, when TCP is used in wired/wireless internetworks, the losses due to channel errors are mistaken as congestion losses and the sending rate is unnecessarily reduced in an attempt to relieve the congestion, resulting in a degraded performance. There are several studies to model the behavior of TCP in such environments, typically under last-hop wireless scenarios. The consensus is that TCP needs some form of intimations to segregate wireless loss from congestion loss and behave accordingly in its window control. However, it is not an easy task to detect the type of loss from TCP behavior as shown in this report with the help state transition models. In order to further extend the model for more accuracy in capturing the exact TCP window control, we plan to carry out a series of simulation studies for a synthetic heterogeneous environment with multiple TCP/UDP flows, keeping the state diagram in mind.

Key words: TCP/IP, wireless networks, congestion control, wireless loss, multiple flows, UDP, State Transition Model, loss detection.

Introduction

User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are examples of end-to-end transport protocols used in the Internet. Both protocols offer multiplexing functionality, which enables more than one process to use the network simultaneously. However, UDP and TCP differ in the kind of service they offer to the application layer protocols. UDP offers a connectionless, unreliable datagram service. It ensures that corrupted packets are not delivered to the application layer, but makes no effort to request a retransmission of a corrupted or missing segment. It assumes that the application layer will provide error correction if it is necessary.

TCP, on the other hand, offers a reliable, connection-oriented, byte-stream service. TCP assumes that the underlying network infrastructure is unreliable and could lose or duplicate segments during their transmission. It is a full-duplex protocol, which means that each TCP connection supports a pair of byte-streams, one flowing in each direction. TCP is an *end-to-end* protocol. That is, TCP turns a host-to-host packet delivery service, provided by IP, into a process-to-process communication channel TCP makes every effort to ensure that the application protocol receives its data in order, without duplicates, exactly as it was sent. In addition, TCP supports flow control and congestion control. *Flow control* ensures that the sender does not send too much data to overrun the receiver's buffer. *Congestion control* ensures that the sender does not transmit more data into the network than it can handle.

Congestion is the phenomenon that occurs at a router when incoming packets arrive at a rate faster than the router can switch (or forward) them to an outgoing link. Then the queue size in router increases, eventually exceeding available buffer space. There are ***two classes*** of router algorithms, namely queue management (QM) and scheduling, that are related to congestion control. QM algorithms manage the length of packet queues by dropping packets when necessary or appropriate, while scheduling algorithms determine which packet to send next and are used

primarily to manage the allocation of bandwidth among flows [1]. The most common scheduling algorithm is FIFO (First-In First-Out), which means that the packet that first enters the buffer will be the one to leave the buffer first. We also assume FIFO in the rest of the paper and do not discuss scheduling further here as it is out of the scope of our work.

When the buffer is full, some packets will have to be dropped. A straightforward solution is to drop the packets that are just arriving at the input port; that is, if a packet arrives and finds the queue full it will be dropped. This policy is known as tail-drop or drop tail (DT). In essence, it is a *reactive approach* to QM. Other solutions include *proactive approaches*, which come into play before the buffer gets full. It symbolizes active QM (AQM) i.e., informing the sender about incipient congestion *before* a buffer overflow happens so that senders are informed early on and can react accordingly. AQM may involve techniques, such as dropping the first packet in the queue (not very common) or dropping a random packet already stored in the queue. Each of these solutions has certain advantages and disadvantages (which we will not elaborate on due to space limitations), but DT is the most widespread dropping policy. DT is simple to implement, has been tested and used for many years. At the same time, it was shown to interact badly with TCP's congestion control mechanisms and to lead to poor performance. In 1993, Floyd and Jacobson [2] proposed a pro-active random packet dropping strategy, now known as Random Early Detection (RED), in order to solve the above-mentioned problems caused by DT. RED uses randomization in an efficient manner, without requiring any changes at the end hosts. In fact, the subsequent studies of its efficiency in controlling congestion show that RED performs well with TCP that perceive drops as a form of implicit notification of congestion. In particular, a TCP sender will react to these packet drops by reducing its sending window size (i.e., sending rate). This slower sending rate translates into a decrease in the incoming packet rate at the router, which effectively allows the router to clear up its queue. The IETF (Internet Engineering Task Force) has already recommended for the wide deployment of RED some years ago. It is supported by router vendors

and is implemented in a number of products. Nonetheless, ***RED is not currently widely used, despite the eight years that have passed since its introduction***.

Heterogeneous Environment

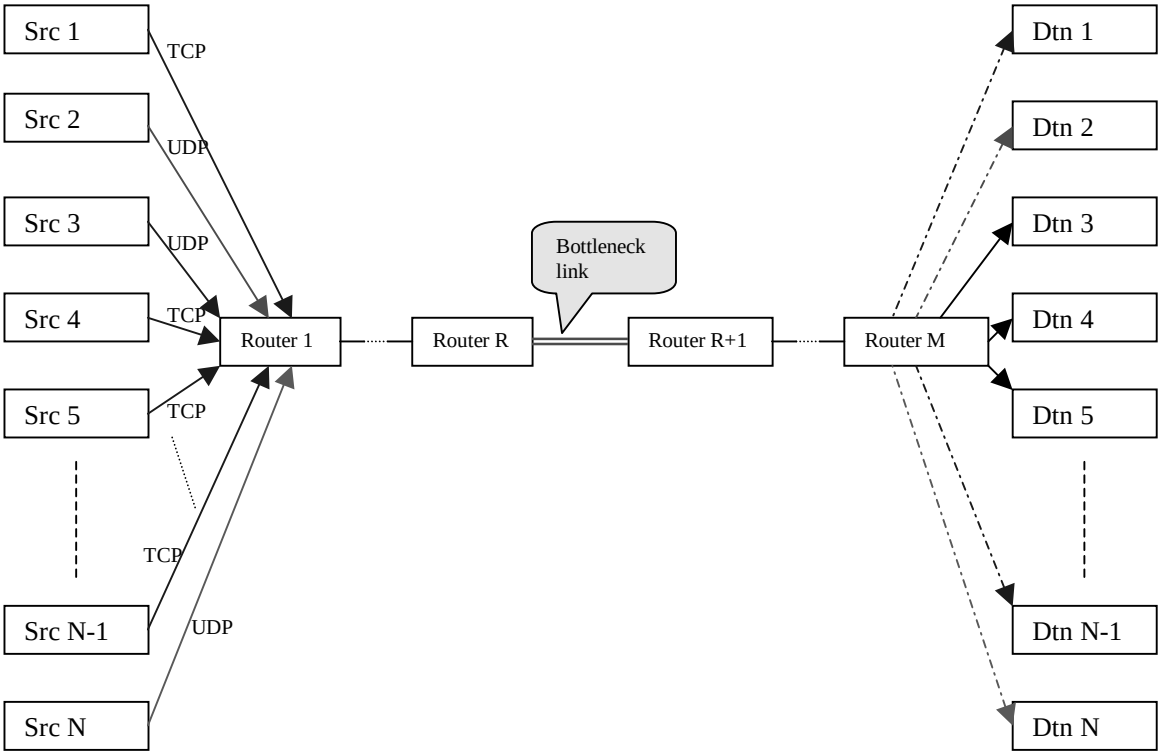

Figure 1. Heterogeneous network under consideration

Heterogeneity (in terms of transmission media) is an attribute that most characterizes the evolution of modern communications networks. The popularity of wireless networking has increased dramatically over the past few years. The recent proliferation of mobile devices and the introduction of efficient wireless communication protocols have placed new requirements on existing protocols, such as TCP (the most prevalent congestion control protocol used on the wired Internet), that must evolve to meet the needs of this new mobile Internet. Integrating high delay, high channel error, wireless networks with existing wired networks still poses significant challenges to the research community. Incorporating end-to-end congestion control for wireless networks is one of the primary concerns. Consider, for example, the problem of providing reliable *end-to-end* data delivery to the application layer in an Internet that includes both wired and

wireless networks. Though wireless last hop is most common today, wireless links are expected to continue to proliferate and will exist anywhere in a future heterogeneous internetwork. It has been found that TCP and UDP may be used (may be inefficiently) in a variety of networks, including Ethernet, FDDI, and WLAN. They can run over a variety of physical media, including wires (such as optical fibers), and wireless (such as radio waves, or infrared signals). However, unlike UDP, TCP being an end-to-end protocol, it shows some unique behaviors in a heterogeneous environment that we discuss below.

Though a TCP connection usually works over all these different media/networks, historically TCP has been designed mainly for wired networks [3]-[5]. So it has been highly tuned with certain assumptions in mind, which are valid only for wired networks. For example, the BER (or the probability that any given bit might get corrupted) is small over wired media (for instance, 10^{-6} to 10^{-7} for copper cable, and 10^{-12} to 10^{-14} for optical fiber). But, in wireless media, BER is normally in the range of 10^{-1} to 10^{-3}, which is quite unacceptable for normal TCP operation. Again, when a data segment is lost, in a wired network it is relatively safe to assume that this was most likely due to congestion (i.e., more segments than what network can handle). But, if a wireless link or sub-network is involved, it could as well be because of channel error resulting in a bad reception at the destination. Thus, packet loss in wired networks is primarily due to congestion, and as such is not applicable to wireless networks in which the bulk of packet loss is due to error at the physical layer (lossy channel). Certainly, TCP was not designed with such heterogeneity in mind; hence, the performance of TCP over heterogeneous internetworks (where links may be wired and/or wireless) is severely degraded due to packet losses.

Packet losses in afore-mentioned heterogeneous internetworks may be due to: a) errors on the wireless channel, b) congestion in a bottleneck wired link, or c) both. The worst of them is, however, wireless losses, which are interpreted to be a (pseudo) congestion by TCP, leading to immediate window cutbacks resulting into poor channel utilization. Even at a loss rate of as low as 10%, TCP throughput is found to degrade catastrophically. In mobile wireless networks where

wireless link is the last hop, errors on that link can cause TCP sender initiated retransmissions, and window cutbacks because TCP interprets the loss as being caused by congestion on some bottleneck link in the wired backbone network. Hence, for heterogeneous networks, optimal TCP window control is still an open issue.

We identify the key differences between wireless loss and congestion loss as: a) wireless error is natural (we have no control over it) and unavoidable, whereas congestion is created by network and we may avoid it by adding extra resources to some extent, b) wireless loss is unpredictable, whereas congestion provides us with some early warning, and c) wireless loss is a transient event, whereas congestion is a continuous event (so packet loss is correlated). The differences are summarized in Table I.

Table I: ***Distinctions between loss types***

Wireless loss	***Congestion loss***
Due to channel error	Caused by packet dropping at buffer
High BER leading to high Packet Error Rate (PER)	Zero BER but high Packet Error Rate (PER)
Mostly random; sometimes bursty (correlated)	Always correlated
Transient (off-and-on) event; non-differentiable in time domain	Continuous event; differentiable in time domain
Beyond our control (not estimable)	We can control (estimable)
Unpredictable	Predictable
Happens without any prior indication	Happens with a priori indication

Review of Wireless TCP Works

Various proposals have been worked out to improve the performance of TCP over heterogeneous internetworks, where both congestion losses and losses due to the transmission errors can occur. We review them briefly here.

The end-to-end schemes (Split-TCP) such as I-TCP [6], WTCP [7] have been proposed to mitigate the performance loss without attempting to decouple the cause of the packet loss. The I-TCP splits the transport link at the wireline–wireless border. The base station maintains two TCP connections, one over the fixed network, and another over the wireless link. Since I-TCP uses TCP over the wireless link, it suffers from poor performance in dealing with wireless losses. In

WTCP, the TCP connection from the source is terminated at the base station, and another reliable connection is created from the base station to the mobile host. However, the base station acknowledges a TCP segment to the source only after the mobile host acknowledges that segment. This way, TCP's end-to-end semantics are preserved.

Snoop [8] is similar to WTCP, except that it is implemented at the link layer of the base station. The base station sniffs the link interface for any TCP segments destined for the mobile host, and buffers them if buffer space is available. If the base station sees a duplicate acknowledgment, it detects a segment loss and if it is buffered then Snoop identifies a loss over the local wireless link, retransmits the lost segment and starts a timer. The duplicate acknowledgment is also dropped to avoid unnecessary fast retransmission at the sender.

A simple solution proposed in many works tries to follow the "divide and conquer" strategy. They employ the simple solution of allowing a link level retransmission scheme to recover the packets lost due to channel errors thereby limiting the response of the transport protocol to mostly congestion losses. This divides the losses into two layers, namely link and transport, because both layers have error recovery capabilities, though at different levels (link layer is hop by hop; transport is end-to-end). So the link layer is responsible to recover from losses due to transmission errors, allowing the transport protocol to recover from congestion losses. We may call it as *localization of wireless loss*. However, in order to maintain the segregation between the different layers, the link layer should not be required to be aware of the semantics of the transport level protocol. Also, in order to make the deployment of the new solution easier, the modifications should preferably be minimal. It has been observed that link-level error recovery protocols, proposed for improving the performance of TCP over wireless networks, achieve considerable throughput performance improvement. The solutions include Snoop [8], Explicit mechanisms [9] such as (Explicit Bad State Notification, Explicit Loss notification etc.), Delayed Duplicate Acknowledgements. When TCP Sack [10] and TCP Vegas [11] are compared with and without Snoop, they display a contrasting behavior, with TCP Sack being the best without Snoop

and TCP Vegas the worst, and TCP Vegas the best with Snoop and TCP Sack the worst. This led to the modification of Snoop code to make TCP Sack to behave the best with Snoop also.

TCP Vegas compares the actual RTT with the expected RTT and accordingly increases and decreases the congestion window when the difference is lower or higher than predefined threshold. It prevents the congestion in the network. The disadvantage of these algorithms is that these require higher computational capability. The same idea is used in TCP-Probing [12] where the detection of a packet loss triggers the sending of probes to measure the current RTT.

Other end-to-end solutions do not mention explicitly the cause of packet loss; it is worth mentioning as these enhancements are caused to improve TCP performance. TCP Westwood [13] measures goodput i.e., the reception rate and uses that rate to set the congestion window when a packet lost is detected. TCP Santa-Cruz [14] follows the same concept in TCP Vegas, except that it replaces the round trip delay measurements of TCP with estimations of delay along the forward path, and uses an operating point for the number of packets in the bottleneck. TCP-real [15] combines the idea from TCP Westwood and TCP Santa- Cruz, by measuring the rate at the receiver.

A *traditional way* of looking at this problem is not to distinguish between packet loss caused by congestions and that due to wireless channel error, and to solve the complete system as a black box [16]-[18]. Thus, by not taking into account the packet loss due to wireless channel error when determining the sending rate, the problem is translated into an old one, for which a known solution exists. Still, researchers are working for the optimal solution.

Another approach is to peep into the TCP and/or IP headers and to understand the cause of error by adding/setting extra bits (such as ECN) as modifications to the network protocol. The solutions based on this point of view are not easy to deploy because they requires modifications to network infrastructure or protocol. Moreover, due to security reasons, this approach is discouraged; also encrypted traffic in protocols, such as IPSec, cannot be seen at intermediate routers.

Another approach is to provide a framework where both channel loss and congestion loss can be treated as is appropriate for wireless access where the channel loss can be dominant or comparable to congestion loss. A preliminary distinction between these two losses is possible on the basis of a fact indicating that congestion loss is correlated while channel loss may vary from random to bursty. In these works [19]-[24], they used an abstraction for wireless loss scenarios for channel-induced packet loss for a bulk TCP transfer between source and destination over a single link (i.e., queue along with channel). On the other hand, other works [25]-[29] considered a typical abstraction for congestion loss model that randomly drops packet at the buffer. These losses are over an end-to-end path with multiple links shared a large number of TCP sources and destinations.

Previous Works on Loss Detection

This work [30] extends TCP-Friendly Rate Control (TFRC) algorithm to use an end-to-end loss differentiation algorithms (LDAs) when a connection uses at least one wireless link in the path between the sender and receiver. When a TFRC receiver detects losses, it invokes the LDA. If the LDA classifies the loss as a congestion loss, then the TFRC receiver includes it in its calculation of the loss event rate. However, if the LDA classifies it as a wireless loss, then the TFRC receiver does not count it in the loss event rate. Either way, a lost packet is not retransmitted. It assumes link level retransmission.

This work [31] proposes an end-to-end method to classify the nature of packet loss in a hybrid wired/wireless environment. Specifically, they monitor the most recent RTT at the time each loss occurs to determine its most likely nature. Proposed end-to-end solutions differ mainly in the measure(s) they use to infer the cause of loss. These measures may be estimated at the sender without any support from the receiver (e.g. round-trip delay), or may require support from the receiver (e.g. one-way delay or delay variance). This approach integrates two techniques: packet loss pairs (PLP) and Hidden Markov Modeling (HMM). Two back-to-back packets form a packet loss pair, where one packet is lost while the second packet is successfully received. The purpose

is for the second packet to carry the state of the network path, namely the round trip time (RTT), at the time the other packet is lost. Under realistic conditions, PLP provides strong differentiation between congestion and wireless type of loss based on distinguishable RTT distributions. An HMM is then trained so that observed RTTs can be mapped to model states that represent either congestion loss or wireless loss.

In this paper [32] the authors define a class of functions named loss predictors, which may be used by a TCP sender to guess the actual cause of a packet loss (congestion or transmission error) and take appropriate actions. These loss predictors use simple statistics on round-trip times and/or throughput, to determine the cause of a packet loss. In this scheme, the receiving host measures the interarrival times of packets. Assuming the last hop is wireless and the bottleneck, if the time between received packets is close to the minimum, then a lost packet in-between is assumed to have been lost due to wireless errors and not congestion.

NCPLD (Non-Congestion Packet Loss Detection) [33] predictors compare the measured RTT with the lowest RTT (or that at the knee of the goodput load curve). If the former is close to the latter, then the cause of a packet loss is assumed to be wireless errors. Otherwise the loss is congestion. In the Spike scheme [34] the receiving host measures one-way delays, and switches to congested (wireless) state as the delay exceeds (drops below) a certain threshold. The ZigZag scheme [30] extends Spike to include the mean and deviation of measured one-way delays as well as number of losses in computing the delay thresholds. Intuitively, the higher the number of losses the higher the threshold beyond which congestion is assumed, i.e. the cause of the loss being wireless errors becomes more likely.

State Transition Model of TCP Window Control

Communication protocols are usually specified by one of the following three methods: 1) *Informal methods*, such as the sequence chart, 2) *Formal methods*, such as State Transition Models (which include Finite State Machines (FSM), Communicating FSM (CSFM), and Petri nets), Programming Languages Models (which include Abstract Programs, CCS (Calculus of

Communicating systems), CSP, and Temporal logic) and Hybrid Model (which includes Extended FSM (EFSM)), and 3) *Language Standards* such as SDL (FSM + extensions), Estelle (EFSM + extended Pascal) and LOTOS (CCS+ADT). In this work, we use CFSM model to specify the TCP window behavior, whosre event-model easily maps to CFSM formalism as shown below.

Communicating Finite State Machines (CFSM)

Communications protocols are idealized as Finite State Machines with state transitions being driven by event invocations from higher and lower layers in the protocol stack, and in some (exceptional) cases by timer events within a layer. In CFSM model, a protocol is described as a set of Communicating Finite State Machines. Each CFSM represents a component (or process) of the network (in OSI term, a protocol entity, e.g. sender, receiver). The channels that connect CFSMs are assumed to be FIFO queues. (An error-prone channel is also modeled as another CFSM). Selecting the model of specification, in order to effectively capture the protocol functionality, is very important, as it has a great impact on the (possibly automatic) implementations. CFSMs are an extension of FSMs, hence, they are intrinsically good in modeling control paths. CFSMs also can include data computations as part of the transitions, and can therefore model protocol data-paths as well. CFSM networks are globally asynchronous, locally synchronous; thus they are able to model effectively asynchronous communication events. In short, CFSMs are a natural candidate for modeling TCP window behavior.

An FSM is often simplest to visualize and to describe in graphical form. This form allows state transition events to be traced by hand, it also allows explicit data to be captured and described. Hence, each CFSM is represented by a directed labeled graph where nodes represent states (conditions) of the process; and edges represent transitions (events) of the process. For example a simple Request-Response protocol is modeled in Figure 2 as a CFSM. Transitions include actions taken the process (e.g. the sending a message) or external stimuli (e.g. the reception of a message). The sending message transition is labeled as -Msg where Msg is the type of messages

being sent. The receiving message transition is labeled as +Msg where Msg is the head message on the incoming FIFO queue of the CFSM (Figure 2). The receiving mode of TCP window can be mapped to states while event processing can be associated with the transitions: an event triggers a transition, which performs the event processing.

Initial node is the starting state of a CFSM. Starting at the initial node, a CFSM traverses the nodes and transitions. The node currently being visited is called the current node. When a machine traverses a sending transition, it sends/appends a message with the same label to its outgoing channel. A machine at a node cannot traverse its receiving transition unless there is a message matched with the same label on the head of its incoming channel. When a machine traverses a receiving transition, it removes the matched head message of its incoming channel. Among several possible transitions, a machine traverses one non-deterministically.

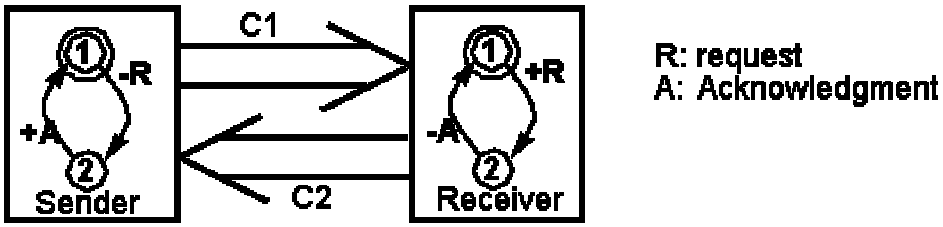

Figure 2. CFSM operating semantic for a Simple request-response protocol

State Transition Model of TCP Window

We first present a 4 state model for TCP window in normal wired environment. The states are numbered as 0, 1, 2 and 3 as shown in Figure 3. State 0 corresponds to the initial slow start phase of TCP window. From this state, it moves to state 1 where the window operates in the congestion avoidance phase. These states correspond to various conditions of the window as shown in Figure 4.

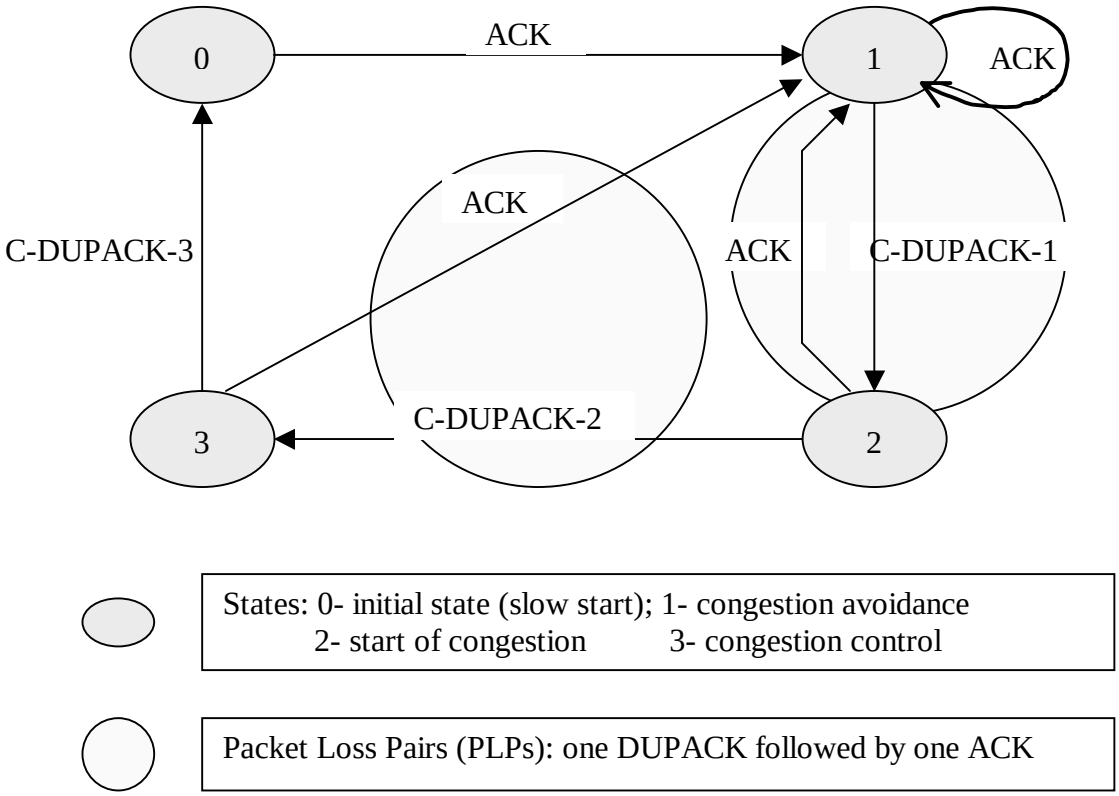

Figure 3. State diagram of TCP window control in wired environment

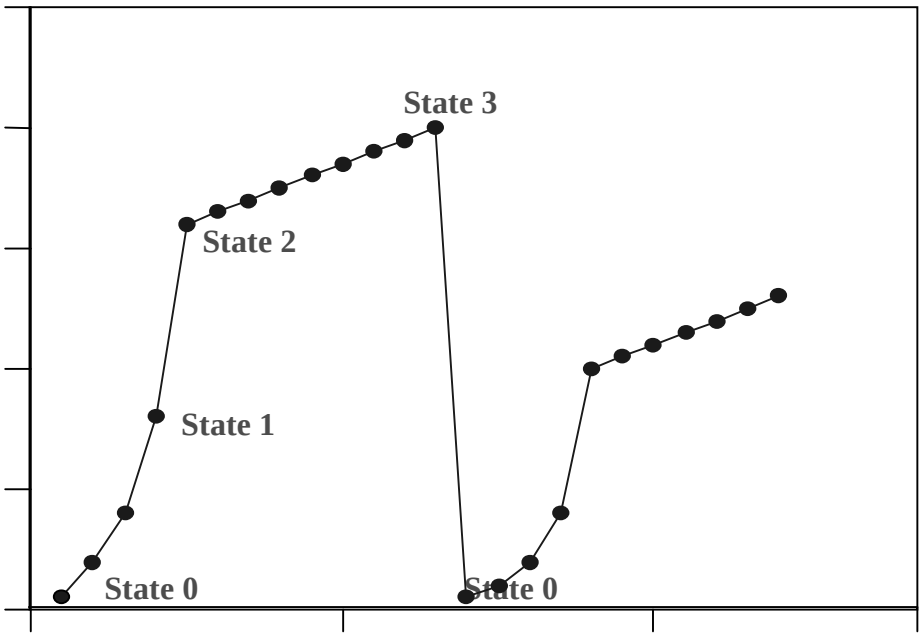

Figure 4. TCP window control with state identifications

It is to be noted that, among states 1, 2 and 3, there are two packet-loss-pairs (PLPs), where each PLP involves a DUPACK followed by an ACK. C-DUPACK represents DUPACK due to congestion. Since this model is for wired environment only, there will not be any other DUPACK except due to congestion in the bottleneck link.

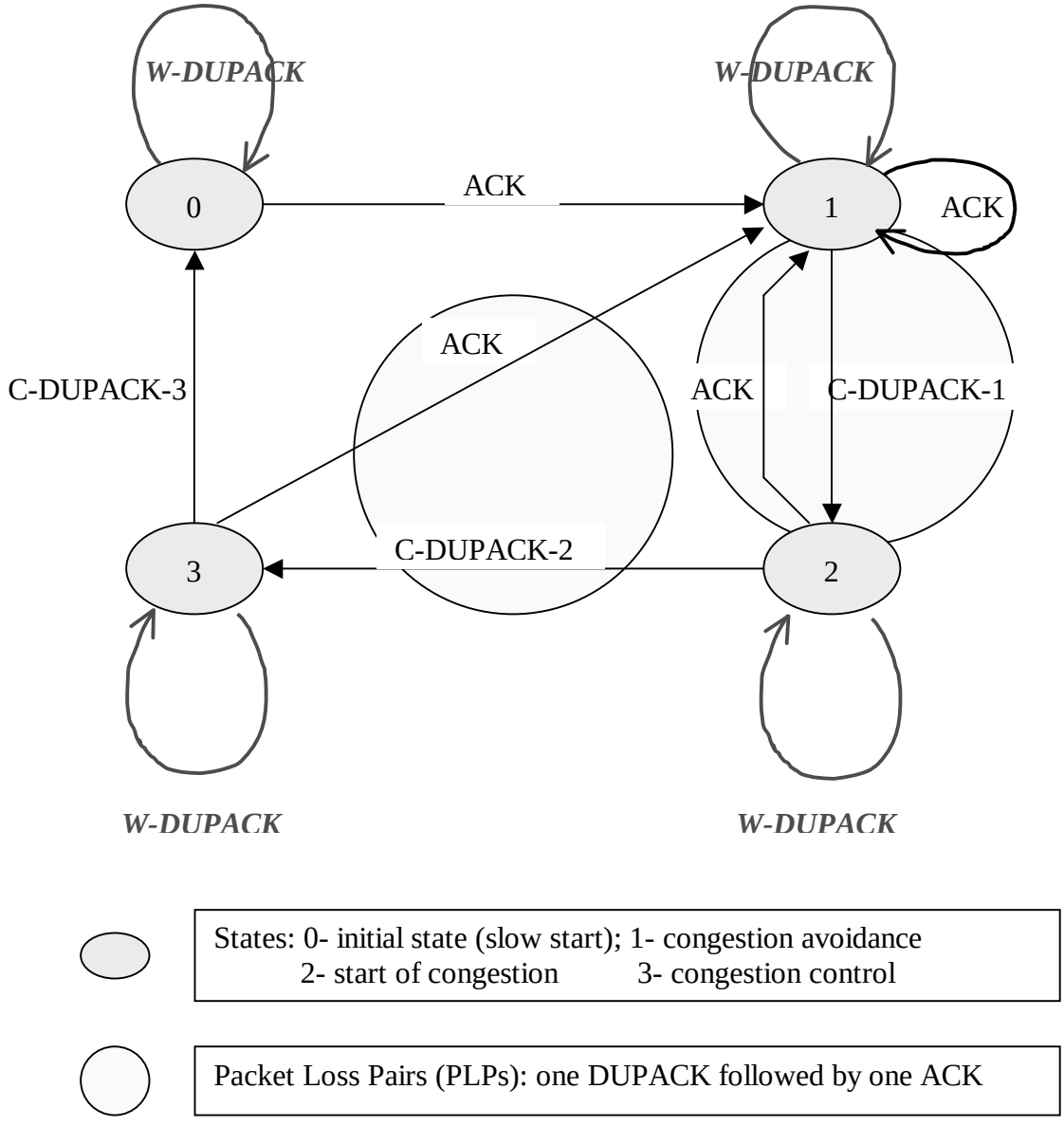

Figure 5. State diagram of TCP window control in heterogeneous environment

Now, if we wish to enhance this diagram to include wireless channel losses, Figure 3 modifies into Figure 5, where red state transitions, named W-DUPACK (Wireless DUPACK), should not incur any state transitions ideally. But the million-dollar question is how to identify that it is a W-

DUPACK transition? In state 0, it is easy to detect wireless DUPACK because there is no other transition from state 0, named as DUPACK. But it is not true for states 1, 2 and 3 because there wireless DUPACKs are co-existent with congestion DUPACKs. Hence, though we do not want any state transition for wireless DUPACKs, TCP will confuse with congestion DUPACK (as they are indistinguishable) and will make a transition to the next state as per congestion DUPACK rule. This will cause undesirable state transitions leading to slow start all over.

The intriguing issue here is to find out some mechanism to differentiate between red W-DUPACK and black C-DUPACK in states 1, 2 and 3. The most common measure used in the literature is RTT based. But our model shows that it is insufficient.

Conclusion and Future Work

TCP can detect packet drops and interpret them as indications of congestion in the network. But, at the same time, it mistakes a channel loss as congestion because channel loss also gives rise to packet loss. This makes TCP, in its current form, a bad choice for wired/wireless environments. In this work, we have modeled the window behavior as a state transition graph to show that why and when TCP window can not distinguish between wireless loss from congestion loss. This provides a new direction to the problem and helps us to rethink about the issue for a plausible solution.

One may argue that including TCP in the limited memory of a wireless phone may not be technically feasible or economically reasonable. Instead, a simpler transport protocol, like UDP, should be used. Or, perhaps, one can avoid the transport layer altogether. In light of the principles of layered system design and the end-to-end argument, one should be cautioned against these approaches.

In future, we plan to evaluate the performance of various TCP modifications using the ns-2 simulator [35] based on the observations obtained from the state transition model. The heterogeneous internetwork topology to be considered in the simulation is the same one as shown in Figure 1, where *N* different sources are connected to the router R_1, which in turn is connected

to the router R_R over some wireless and wired links. R_R is connected to R_{R+1} via a bottleneck link. R_{R+1} is finally connected to the router R_M over again some wireless and wired links. *N* different receivers are connected to the router R_M by wired and wireless links. Several recent studies [30],[31],[36] have noted that the number of flows in the network significantly impacts TCP performance evaluation in heterogeneous networks. To emulate real life scenarios, we consider various traffic conditions together. Most of previous studies have considered the so-called greedy users assumption whereby sources are saturated. Here, in our model, multiple TCP flows are intermingled with multiple UDP flows. All flows are passing over a combination of wired and wireless links. Moreover, some flows for both TCP and UDP are on last hop wireless.

References

1. Braden, B., D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, Recommendations on Queue Management and Congestion Avoidance, RFC 2309, April 1998.
2. S. Floyd, and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. ACM/IEEE Transactions on Networking, August 1993.
3. V. Jacobson. Congestion Avoidance and Control. Proceedings of the ACM SIGCOMM, August 1988
4. R Jain. A time-out congestion control scheme for window flow-controlled networks. IEEE Journal of Selected Communications, Oct 1986, pp 1162-67.
5. T.Ott, J.H.B. Kemperman, M. Mathis. The stationary behavior of ideal TCP congestion avoidance. [Online] available: ftp://ftp.belcore.conffpub/tjo/TCPWindow.ps, 1996.
6. A. Bakre and B. R. Badrinath. I-TCP: indirect TCP for mobile hosts. Proceedings of the 15th. International Conference on Distributed Computing Systems (ICDCS), May 1995.

7. Karu Ratnam and Ibrahim Matta. WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links. In Proceedings of the Third IEEE Symposium on Computer and Communications (ISCC ’98), June 1998.
8. H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. Proceedings of the 1st ACM Int’l Conf. On Mobile Computing and Networking (Mobicom), November 1995.
9. K. Ramakrishnan and S. Floyd, “A Proposal to add Explicit Congestion Notification (ECN) to IP,” RFC 2481, January 1999.
10. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. RFC 2018, April 1996.
11. L. Brakmo and L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas of Communications, October 1995.
12. V. Tsaoussidis and H. Badr. TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains. Proceedings of the 8th IEEE International Conference on Network Protocols, 2000.
13. C. Casetti, M.Gerla, S. Mascolo, M. Y. Sansadidi and R. Wong. TCP Westwood: End-to-end congestion for wired/wireless networks. Wireless Journal, 2002, pp 467-79.
14. C. Parsa and G. Aceves. Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media. IEEE International Conference on Network Protocols (ICNP ’99), 1999.
15. C. Zhang and V. Tsaoussidis. TCP Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks. Proceedings of the 11th IEEE/ACM NOSSDAV 2001, New York, 2001.
16. S. Low. A duality model of TCP and queue management algorithm. IEEE/ACM Trans on Networking. Oct 2003.

17. A. Akella, S. Seshan, R. Karp and S. Shenkar. Selfish behavior and stability of the internet: a game-theoretic analysis of TCP. Proceedings of the ACM SIGCOMM 2002.
18. D. Katabi, M handley and C. Rohrs. Congestion control for high bandwidth-delay product networks. Proceedings of the ACM SIGCOMM 2002.
19. A. Kumar. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. In ACM/IEEE Transactions on Networking, August 1998.
20. A. Kumar and J. Holtzman. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link, Part II: Rayleigh Fading Mobile Radio Link. Rutgers Univ., Piscataway, NJ, Tech. Rep. WINLAB-TR-133, 1996.
21. M. Zorzi, A. Chockalingam, and R. Rao. Throughput analysis of TCP on channels with memory. *IEEE J. Select. Areas Commun*, 2000, pp. 1289–1300.
22. F. Anjum and L. Tassiulas, "On the behavior of different TCP algorithms over a wireless channel with correlated packet losses. *Proc. ACM SIGMETRICS'99*, 1999.
23. T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. IEEE/ACM Transactions on Networking, June 1997 pages 336–350.
24. A. A. Abouzeid, S. Roy and M. Azizoglu. Comprehensive performance analysis of TCP session over a wireless fading link with queuing. IEEE Trans on Wireless Communication, Mar 2003.
25. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Trans. Networking*, Apr. 2000, pp. 133–145.
26. M. Mathis, J. Semke, J. Madhavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Comput. Commun. Rev.*, 1997 pp. 67–82.
27. J. Padhye, V. Firoiu, and D.Don Towsley. A Stochastic Model of TCP Reno Congestion Avoidance and Control. Univ. Massachusetts, Cambridge, MA, Tech. Rep. CMPSCI-99-02, 1999.

28. E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random losses. *Proc. ACMSIGCOMM' 2000*, Stockholm, Sweden, 2000.
29. E. Altman, K. Avrachenkov, and C. Barakat. TCP in presence of bursty losses. *Comput. Commun. Rev.* 2000, pp. 231–242.
30. S. Cen, P.C. Cosman, and G.M. Voelker. End-to-end differentiation of congestion and wireless losses. *Proc. Multimedia Computing and Networking (MMCN) Conf. 2002* pp. 1-15, San Jose, CA, Jan 23-25, 2002.
31. J. Liu, I. Matta and M. Crovella. End-to-end inference of loss nature in hybrid wired/wireless environment. Proc. WiOpt03: Modeling and Optimization in mobile, ad hoc and wireless networks. INRIA, Sophia-Antipolis, France, Mar 2003.
32. S. Biaz and N. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at receiver. IEEE Symposium ASSET '99, Texas, USA, 1999.
33. N. Samaraweera. Non-congestion packet loss detection for TCP error recovery using wireless links. IEE Communications. 1999, pp 222-30.
34. Y. Tobe, Y. tamura, A. Molano, S. Ghosh and H. Tokuda. Achieving moderate fairness for UDP flows by path-status classification. Proc. IEEE Conference on Local Computer Networks (LCN 2000), 2000.
35. ns-2 Network Simulator. **http://www.isi.edu/nsnam/**
36. C. Liu and R. Rain. Approaches of Wireless TCP Enhancement and a new proposal based on congestion coherence. Proceeding of HICSS, Hawaii, USA, 2003.