A Decision Mechanism for Distributed Variant Nodes Performance in Wireless Adhoc Network

T.K Shaik Shavali [1], Abdul Rais [2], Aeraj Fatima [3]

[1]*Professor*, [2-3]*Assistant Professor*

[1-3]*Department of CSE, LORDS Institute of Engineering and Technology, JNTUH, INDIA*

Abstract— *Ad-hoc networks are usually defined as an autonomous system of nodes connected by wireless links. In wireless Ad-hoc network the connections between the wireless links are not fixed but dependent on channel conditions as well as the specific medium access control (MAC). The channel medium and transmission links are affected by the interference, delay, buffer overflow these may cause the network congestion. To avoid network congestion various congestion control methods were developed in past but they were performed less control of end-to-end congestion and less in per link connection control. To overcome the above problems and to improve the resource allocation an efficient method has to be developed.*

Keywords—*Link persistent, Buffer overflow, Ad-hoc network.*

I. OVERVIEW

A hybrid network is defined to be one that contains mobile devices, relay devices and access points. Mobile devices can be anything from laptops to PDA's to cell phones. Access points give mobile hosts (MHs) access to other devices outside of a MH's transmission range. An example of such a system is shown in Figure 1. In the figure, MH2 can reach MH5 despite the fact they cannot communicate in ad hoc mode. For the purposes of this paper, base station (BS), gateway (GW) and access point (AP) are synonymous. We also use the term mobile host (MH) and node interchangeably. Current wireless infrastructure only provides services, such as DHCP and Internet connectivity, to MHs that are one hop away. That is, the MH is

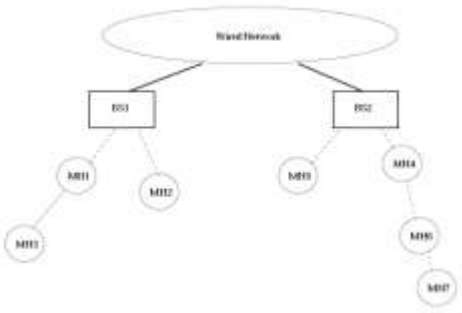

Figure 1: An example hybrid network

within transmission range of the AP and hence the two can directly communicate. This allows MHs to function like hosts on traditional shared media networks that access a gateway for outside connectivity and centralized services. However, this model does not take advantage of the plethora of research that has been done in ad hoc networks (discussed in Section 2) to allow MHs to communicate directly. By incorporating ad hoc networking into the existing infrastructure, the "range" of an AP can be extended to allow for more connectivity. For example, in Figure 1, the services of BS2 can be extended to MH6 and MH7 as opposed to just MH4 and MH5. Additionally, when ad hoc networking is incorporated, not all communication between MHs has to go through the AP. This may increase the spatial reuse of the channel and ease the burden placed on the AP. If MH2 and MH5 can communicate directly, we avoid using resources such as the base stations and wired network.

The protocol allows MHs to register with an AP. The "radius" around an AP is limited to a specified number of hops, K. The protocol is designed to test the efficiency of specialized protocols when compared to more general routing protocols like Ad-hoc On- Demand Vector Routing (AODV) [1] and Dynamic Source Routing (DSR) [2]. We believe the K hop paradigm will be useful in small, busy geographic areas, such as a university campus or airport. In such areas, there is usually a limited range of coverage with respect to the entire area. For example, there may be APs inside a building, but users just outside the building cannot access it. However, if there are other mobile users, or dedicated relays, in the lobby of the building, such devices may serve as routers to extend the range of the AP to the outside users.

Our initial goal in designing the protocol is to have a minimal routing implementation that is complete, but not too complex. Essentially, we wanted to extract from existing research only what is required for a functioning routing protocol while not worrying about the many optimizations which may be possible. The protocol was also designed to function with standard forwarding table

implementations. Standard forwarding is a function of three parameters: the destination (and net mask), the next hop and the interface. When a data packet arrives it should only be forwarded based on these criteria. Therefore, the protocol could not use any type of source routing or forwarding based on the packet's source, destination tuple. Our approach to modeling computer networks and its protocols is to use hybrid systems[4] which combine continuous-time dynamics with event-based logic. These models permit complexity reduction through continuous approximation of variables like queue and congestion window size, without compromising the expressiveness of logic-based models. The "hybridness" of the model comes from the fact that, by using averaging, many variables that are essentially discrete (such as queue and window sizes) are allowed to take continuous values. However, because averaging occurs over short time intervals, one still models discrete events such as the occurrence of a drop and the consequent reaction by congestion control.

II. RELATED WORK

Our proposed routing protocol borrows characteristics of the AODV protocol [1],[3]. Consequently, we provide a brief overview of the AODV protocol. DSR [2] is another widely researched reactive protocol, performing route discovery using RREQs and RREPs. DSR maintains full path information for routing as opposed to AODV which remembers only next-hop addresses for destinations. We chose to avoid maintaining source routes due to complexity issues with the Linux kernel and instead opted for following AODV's style for maintaining routes. AODV contains many optimizations and features that make it quite complex. Since one of the primary goals of our protocol is simplicity, we did not incorporate many similar features into it.

Some of these features include multicast support, unidirectional link support, expanding ring search, hello messages and local repair. Our protocol is also akin to the Zone Routing Protocol (ZRP)[4] in that it contains both proactive and reactive mechanisms. However, ZRP uses proactive routing within a zone, and reactively exchanges routing information between zones, whereas in the protocol

we present, access points proactively maintain routes to reachable mobile devices, with local route discovery being completed in a reactive fashion. Yet another protocol that achieves similar same goals as ours is LUNAR [5]. The Lightweight Underlay Network Ad-Hoc Routing (LUNAR) protocol implements a layer in between the MAC layer and the IP layer to perform a variation of multi-hop ARP. The protocol is simple and is designed to work in small-hop environments. LUNAR uses a combination of reactivity and proactivity for route discovery and maintenance. The protocol performs automatic address assignment and supports Internet access, a similar objective of our protocol. A special gatewaying node handles passing packets back and forth between the Internet and the LUNAR network. Our protocol differs from LUNAR in the following ways:

• We do not support multi-hop ARP, and use DHCP for address assignment rather than a randomized scheme.

• The LUNAR model requires encapsulation of data packets, which we chose to avoid due to the amount of overhead incurred by passing packets back and forth between user space and the kernel.

• In LUNAR, broadcasts are performed using a tree; we use a traditional broadcast scheme.

• We employ AP broadcasting to identify nodes in each zone, LUNAR does not have any registration process.

The approach we take of combining small, K-hop ad hoc networks with a structured backbone has been studied by several others [6]-[11]. The focus of [6] and [7] is on relieving congestion in cellular networks by introducing ad hoc functionality among mobile devices. We are not explicitly concerned with network congestion. In the Multi-hop Cellular Network (MCN) [8] architecture, nodes in a cell are allowed to use ad hoc mechanisms to improve throughput or reduce the number of needed base stations. A-GSM [9] allows devices to relay through each other to reach a base station, in an effort to reduce the occurrence of dead spots in the network. A performance comparison between ad hoc and cellular networks is presented in [10], along with a suggested hybrid system that switches back and forth between cellular and ad hoc modes based on network conditions. Our model requires all nodes to operate in the ad hoc mode, including the APs. Finally, in [11], the authors offer enhancements for ad hoc networks to increase overall throughput. One of the additions, called assisted scheduling, aims at using a central base station to coordinate flows between ad hoc nodes. Our model relies on the DCF mode of IEEE 802.11 [12] for packet communication and does not impose any strict scheduling.

III. HYBRID SYSTEM MODELING FOR CONGESTION CONTROL

We start by describing how TCP's congestion control mechanism can be modelled using a hybrid system, i.e. a system that combines continuous dynamics with discrete logic. We consider here a simplified version of TCP-Reno congestion control [15]-[17] but the model proposed also applies to more recent variations on Reno such as New Reno, SACK [6], and general AIMD [18].

We denote by RTT the *round-trip time*, and by w_i and w_i^{th}, $i \in \{1, 2, \ldots, n\}$ the *window size* and *slow-start threshold*, respectively, for the

congestion controller associated with the ith flow. As we will see shortly, the round-trip time is a time-varying quantity. In Reno, the algorithm to update w_i is as follows: While the window size w_i is below the slow-start threshold w_i^{th} the congestion controller is in the *slow-start* mode and w_i is multiplied by a fixed constant m_{ss} every round-trip-time RTT. When the window size raises above w_i^{th} the controller enters the *congestion avoidance* mode and w_i^{th}, the controller enters the *congestion avoidance* mode and w_i is incremented by a fixed constant $a \geq 1$ every round-trip-time RTT. The above takes place until a drop occurs. A drop can be detected through two mechanisms that lead to different reactions by the congestion controller. When the drop is detected because of the arrival of three consecutive duplicate acknowledgments, w_i is multiplied by a constant $m \in (0, 1)$ and the system proceeds to the congestion avoidance mode. In some cases, three consecutive duplicate acknowledgments never arrive and a drop is detected when a packet remains unacknowledged for a period of time longer than the *retransmission timeout* RTO. In this case, the slow-start threshold is set equal to mw_i and w_i is reduced to one. Unless there are many consecutive drops per flow, timeouts occur mostly when the window size becomes smaller than four and therefore no three duplicate acknowledgments can arrive after a drop. Although the window size takes discrete values, it is convenient to regard it as a continuously varying variable. The following hybrid model provides a good approximation of the ith window size dynamics: While the ith flow suffers no drops, we have in the slow start mode

$$\dot{w}_i = (\log m_{\text{ss}}) w_i / RTT$$

and in the congestion avoidance mode

$$\dot{w}_i = a / RTT.$$

When a drop is detected at time t through three duplicate acknowledgements, we have

$$w_i(t) = m\, w_i^-(t)$$

Where $w_i^-(t)$ denotes the limit from below of $w_i(s)$ as ↑ S, and when the drop is detected through timeout, we have

$$w_i^{\text{th}}(t) = m\, w_i^-(t)$$
$$w_i(t) = 1$$

The round-trip time is given by $RTT(t) = T_p + q(t)/B$ Where T_p denotes the *propagation time* (together with some fixed component of the service time at nodes n_1 and and n_2) and $q(t)$ is the size of the output queue of node n_1 at time t. We assume here that the bandwidth B is measured in packets per second. Denoting by w_i^{adv} the advertised window size for the ith flow, the output queue at node n_1 receives a total of

$$r := (\sum_i \min\{w_i, w_i^{\text{adv}}\})/RTT$$

packets per second and is able to send B packets to the link in the same period. The difference between these two quantities determines the evolution of $q(t)$.In particular,

$$\dot{q} = \begin{cases} 0 & q = 0, r < B \text{ or } q = q_{\max}, r > B \\ r - B & \text{otherwise} \end{cases}$$

The first branch in above equation takes into account that the queue size cannot become negative nor should it exceed the *maximum queue size* $q_{\max}$. When $q(t)$ reaches $q_{\max}$ drops occur. These will be detected by the congestion controllers some time later. As mentioned drops will occur whenever q reaches the maximum queue size $q_{\max}$, and the rate of incoming packets to the queue r exceeds the rate B of outgoing packets. Typically, drops detected through duplicate acknowledgments will be detected roughly one round-trip time after they occur, whereas the one detected through timeout will be detected RTO seconds later. Because of this delay in detecting a drop, the rate of incoming packets will not change immediately after the queue becomes full and multiple drops are expected. To complete our model we need to know which flows will suffer a drop during this interval. To determine exactly the set of flows $\mathcal{D} \subset \{1, 2, \ldots, n\}$ that suffer a drop, one would need to keep track of which packets are in the queue, leading to a complex packet-level model. However, for purposes of traffic flow analysis, it is sufficient to assume that $\mathcal{D}$ is a function of the window sizes of the individual flows (w_i). Denoting by $\mathcal{D}(t)$ the set of flows that suffer drops at time t, we have

$$\mathcal{D}(t) = F_{\text{drop}}(w_1, w_2, \ldots, w_n)$$

We call F_{drop} the *drop model*. As we shall see below, several drop models are possible, depending on the queuing discipline. Although drops are essentially discrete phenomena, we can incorporate them in our hybrid model by considering distinct

modes (or discrete states) for the system. Four modes should be considered to cover all possible cases: slow-start or congestion avoidance and, in each case, queue full or not. The queue-full modes are active from the moment q reaches $q_{\max}$ until the drops are detected and congestion control reacts leading to a decrease in the queue size q. The time it takes for this to happen is either RTO or RTT depending on whether w_i was smaller than 4 or not at the time of the drop. When the drop is detected, the flows in $\mathcal{D}$ will suffer the appropriate changes in their window sizes and slow-start thresholds. The transition from slow-start to congestion avoidance occurs when the window size w_i exceeds the slow-start threshold w_i^{th}

IV. THE ROUTING PROTOCOL

Our routing protocol attempts to take advantage of a combination of reactive and proactive components similar to ZRP. The AP maintains data about routes to all MHs within its own zone. Each AP tries to proactively discover devices in its own zone and then uses this information in answering RREQs.The collection of MHs are the reactive component, only maintaining routes to devices which they are actively using, either as a source or intermediate hop. The routing protocol essentially consists of four separate phases:

1. Beaconing: The process by which APs learn of MHs within their zone.
2. Route Discovery: Occurs when a device needs to send data to a destination for which it does not have a route.
3. Error Handling: The process by which the protocol handles route errors.
4. Data Forwarding: This phase is concerned with forwarding data packets.

All of these phases can operate concurrently, and are described in more detail in the following subsections. The routing protocol is slightly different for APs and MHs. In this section, we describe how each phase is handled and differences in how APs and MHs respond. It should be noted that the routing protocol is designed to function in this hybrid network as well as in a purely ad hoc setting. The MHs essentially run a stripped down version of AODV. The APs respond differently by taking advantage of proactive state and connectivity to other APs whenever possible.

Route Errors

A route error will occur for two reasons. The first is when a data packet is received from another MH with an IP destination that is not in the local forwarding table. Such a situation may occur when one MH's forwarding entry times out before another's. The second situation is when an MH detects the loss of communication with a neighbour in its forwarding table. A MH considers a link to be down whenever the MAC layer has to drop a packet after multiple retransmissions. In both cases, the error must be propagated toward the source of the data packet. Within our implementation, the information we get when one of the two errors occurs is not as complete as we would like. In the case where a destination does not exist in the table, we receive the original IP source and ultimate IP destination of the data packet. In the case where link layer loss is detected, all we are given is the MAC address of the next hop for the data packet.

Another way to detect link loss is to require hello or heartbeat messages to be periodically broadcast with a TTL of 1. This indicates that communication with a neighbour is possible. This approach is rather heavyweight as it requires another control message and a periodic broadcast. Also, the latency of learning about the downed link is usually greater. When a data packet arrives and there is no forwarding table entry, the MH will generate a RERR, unless default routes are used. The RERR simply specifies the unreachable destination IP address. If the MH has a route to the IP source, the RERR is unicast, otherwise it is broadcast with a TTL of K hops. When a data packet is dropped at the MAC layer, a RERR is created with the unreachable neighbour's IP address and sequence number in it. Additionally, all destinations for which this neighbour serves as the next hop are also removed from the forwarding table and placed in the RERR packet. This packet must be broadcast because we have no knowledge of who the original sender was. The broadcast is done with a TTL of K. This procedure could be optimized if the WLAN card driver was modified to specify the source of the dropped packet.

When a MH receives a RERR, it will update its forwarding table. This is done by removing entries which use the RERR sender as the next hop to the specified destinations. This MH then creates a new RERR packet based on its unreachable destinations. If there is at least one unreachable destination, the new RERR packet is broadcast with the TTL decremented by one. If the TTL becomes zero, the packet is dropped. In this manner, the RERR will eventually reach the original data sender (assuming a RERR is not lost along the way and the original sender is reachable). The only difference in the way APs handle errors is when the AP generates and broadcasts RERRs, they are sent on the wireless link as well as to each of the other APs. When APs receive RERRs from other APs, they will broadcast or unicast the message within their zone after the appropriate updates. Any broadcast after a RERR is received from an AP is done with a TTL of K. When APs receive RERRs from MHs in their zone that are broadcast, they will broadcast the RERR on the wireless interface and send it to all the APs. There are a couple of alternative methods to propagate RERRs toward a source. First, as in AODV, a predecessor list for each destination could

be maintained. This would specify the subset of neighbours that uses a MH to communicate with each destination. This subset is all the neighbours for which a MH forwards or generates a RREP. With this information, we could unicast RERRs only to the neighbours which are known to use the link. They can propagate the RERR further back in a similar manner. Another alternative would be to maintain a list of sources that use a MH for a given destination. For each source, the MH maintains a special reverse pointer to the source. This pointer is not placed in the forwarding table or used in routing, but is only used to provide a path from the MH to the source in the event of a RERR. In this case, the RERR can always be unicast to all the sources that use a given destination and all intermediate nodes along the path could also delete the unreachable destinations from their forwarding tables. This scheme would require the maintenance of these special reverse pointers.

V. ROUTING PROTOCOL MODIFICATIONS

To support routing to hosts on the Internet, we have modified our routing protocol. In this section, we describe the differences between the basic and modified protocols.

Modified Route Discovery

The route discovery process for destinations that are within the ad hoc network (determined by the subnet mask of the destination) . This process may be repeated several times by S before S concludes that the node is unreachable. If the subnet mask for D does not match that of the ad hoc network, S attempts to use its default route (if one is available) to reach D. Recall from that a node's default route is synonymous to the path used to reach its AP, and is discovered during the beaconing process. If no such route exists, S temporarily concludes that D cannot be reached. On the other hand, if S does have a default route, data packets are forwarded to D using this path. Processing of RREQs by APs is unchanged . If an AP has no knowledge of the destination specified in the RREQ, it queries the other APs in the network, one of which will respond if the destination is valid. When replying to a RREQ, APs are recommended, but not required, to send gratuitous RREPs to the destination to avoid unnecessary route discovery by the destination for the source in the case of protocols that require feedback from the destination (e.g., TCP).

Modified Data Forwarding

For incoming external packets (i.e., packets that originate from a host outside of the ad hoc network), an AP A checks its routing table to see if it has a route to the destination. If a route does not exist, A performs route discovery for the destination, an ICMP unreachable message is generated and sent to the host on the Internet if A does not receive a RREP after several attempts. If a RREP is heard, A will forward the packet using the newly discovered route. This operation is only necessary if each ad hoc node has a global address. Assignment of global IP addresses for ad hoc nodes in IPv4 can be achieved by enabling DHCP [19] to operate over multiple hops using the Methods mentioned. If the ad hoc nodes are given private IP addresses, A should not expect to receive any incoming packets for which it does not have an associated connection already established. Outgoing external packets (i.e., packets destined for hosts outside the ad hoc network) in our protocol are forwarded using the default route. Intermediate nodes, upon receiving a packet with an unknown destination, simply use their own default route to forward the data. If a node cannot forward the packet using its default route (e.g., due to link breakage), a RERR is generated, indicating that the default route is outdated, and must be recalculated. Packets that reach the AP are simply forwarded on to A's default gateway. From this point on, an ICMP destination unreachable message is issued to the sender in the event of an error.

VI. OBSERVATION

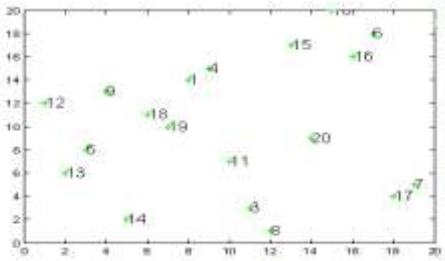

Figure 4:Ad-hoc network

The network parameters used for implementation of the
paper is

Distribution: Random
Number of Nodes: 20
Region : 20 x 20 units
Communication Range : 80 units
Mobility: Static
MAC: 802.11
Link capacitance: 1

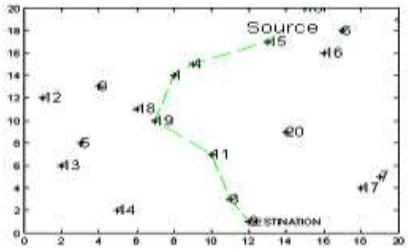

Figure 5: Route from source to destination

The above figure shows the route from the source to destination by the optimal route selection

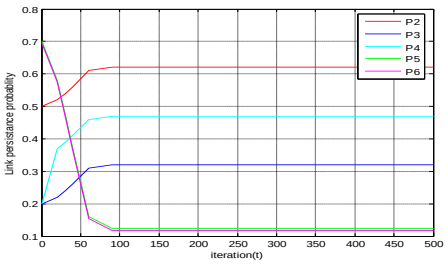

Figure 6: Iterations v/s Link persistence plot

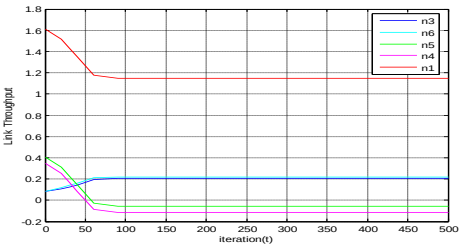

Figure 7: Iterations v/s Link throughput plot

VII. CONCLUSION

The joint design of congestion and contention control for wireless ad hoc networks is studied in this work. While the original problem is non-convex and coupled, we provided a decoupled and dual-decomposable convex formulation, based on which sub gradient-based cross-layer algorithms were derived to solve the dual problem in a distributed fashion for non-logarithmic utilities. Our algorithms decompose vertically in two layers, the network layer where sources adjust their end-to-end rates, and the MAC layer where links update persistence probabilities.

VII. REFERENCES

[1] Jitendra Padhye, Jim Kurose, Don Towsley, and Rajeev Koodli, "A TCP-friendly rate adjustment protocol for continuous media flows over best effort networks," Tech. Rep. TR 89-04, UMASSCMPSCI, 1998.

[2] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in Mobile Computing (Imielinski and Korth, eds.), vol. 353, Kluwer Academic Publishers, 1996.

[3] C. E. Perkins, E. M. Belding-Royer, and S. Das, "Ad Hoc on-Demand Distance Vector (AODV) Routing." IETF Internet Draft, draft-ietf-manet-aodv- 10.txt, Work-in-progress, March 2002.

[4] Z. Haas, "A New Routing Protocol for the Reconfigurable Wireless Networks," in Proceedings of the IEEE ICUPC, 1997.

[5] C. Tschudin and R. Gold, "LUNAR: Lightweight Underlay Network Ad-Hoc Routing," tech. rep., University of Basel, Switzerland, January 2002.

[6] C. Qiao and H. Wu, "iCAR: An Intelligent Cellular and Ad-hoc Relay System," in Proceedings of the IEEE IC3N, October 2000.

[7] X. Wu, S. Chan, and B. Mukherjee, "MADF: A Novel Approach to Add an Ad-hoc Overlay on a Fixed Cellular Infrastructure," in Proceedings of the IEEE WCNC, September 2000.

[8] Y. Lin and Y. Hsu, "Multihop cellular: A new architecture for wireless communications," in Proceedings of INFOCOM, pp. 1273–1282, 2000.

[9] G. Aggelou and R. Tafazolli, "On the Relaying Capacity of Next-Generation GSM Cellular Networks," IEEE Personal Communications, vol. 8, pp. 40–47, February 2001.

[10] H. Hung-Yun and R. Sivakumar, "Performance Comparison of Cellular and Multi-hop Wireless Networks: A Quantitative Study," in Proceedings of the ACM SIGMETRICS, June 2001.

[11] Jamshid Mahdavi and Sally Floyd, "TCP-friendly unicast ratebased flow control," Technical note sent to the end2end-interest mailing list, Jan. 1997.

[12] T. V. Lakshman, U. Madhow, and B. Suter, "Window-based error recovery and flow control with a slow acknowledgment channel: A study of TCP/IP performance," in *Proc. of INFOCOMM*, Apr. 1997.

[13] V. Misra, W. Gong, and D. Towsley, "Stochastic differential equation modeling and analysis of TCP-windowsize behavior," in *In Proceedings of PERFORMANCE99*, Istanbul, Turkey, 1999.

[14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proc. of SIGCOMM*, Sept. 1998.

[15] Van Jacobson, "Congestion avoidance and control," in *Proc. of SIGCOMM*, Aug. 1988, vol. 18.4, pp. 314–329.

[16] Van Jacobson, "Modified TCP congestion avoidance algorithm," Posted on end2end-interest mailing list, Apr. 1990, Available at ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[17] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *RFC 2581*, p. 13, Apr. 1999.

[18] Y. Yang and S. Lam, "General AIMD congestion control," in *Proc. of ICNP*, Osaka, Japan, Nov. 2000.

[19] Stephan Bohacek, Jo˜ao Pedro Hespanha, Junsoo Lee, and Katia Obraczka, "A hybrid systems framework for TCP congestion control: A theoretical model and its simulation-based validation," Submitted to the 39th Annual Allerton Conference on Communication, Control, and Computing, July 2001.

[20] Sally Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACMTrans. On Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.