uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Nadia Ramirez Moreno
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Performance Evaluation of an Adaptive PI Rate Control Algorithm for Wireless Ad-hoc Networks

TITRE DE LA THÈSE / TITLE OF THESIS

O. Yang
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

A. Boukerche

R. Hafez

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

PERFORMANCE EVALUATION OF AN ADAPTIVE PI RATE CONTROL ALGORITHM FOR WIRELESS AD-HOC NETWORKS

by

NADIA RAMIREZ MORENO

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies in
partial fulfillment of the requirements for the degree of
Master of Applied Science in Electrical Engineering

School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

This research was sponsored by the National Science Council of Mexico (CONACYT) under award number 187271 - 204491, and by the Mexican Ministry of Public Education (SEP).

Your file Votre référence
ISBN: 978-0-494-32474-5
Our file Notre référence
ISBN: 978-0-494-32474-5

To my parents, family, and Samer

for their endless support, patience and love.

Abstract

Congestion control is an important issue and challenge in both wired and wireless networks. This thesis evaluates the feasibility of implementing an adaptive Proportional and Integral (PI) rate control algorithm for streaming traffic applications in wireless Ad-hoc networks. A performance analysis and a comparison of this adaptive rate controller with TCP New Reno have been conducted. The performance of this congestion controller in different stationary and mobile scenarios is evaluated by means of simulation. The simulation measurements take into consideration the impact of the number of nodes, the number of simultaneous traffic flows, node mobility, as well as the interaction of different routing protocols. The simulation results demonstrate that the adaptive PI rate control algorithm can improve the overall average throughput.

Acknowledgments

Thanks to Prof. Oliver W.W. Yang who has continuously supported me with his research guidance and valuable suggestions throughout all the work during my Master. I would like to extend my thanks to Prof. Chung-Horng Lung of Carleton University, for his kind help, co-supervision, research guidance and valuable feedback.

Special thanks to the "National Council of Science and Technology of Mexico" (CONACYT) under Grant Number 187271 - 204491, and to the "Ministry of Public Education of Mexico" (SEP), who jointly supported this research, giving me the opportunity to follow graduate studies at the University of Ottawa, Canada.

Table of Contents

List of Figures

List of Tables

List of Acronyms

		Section of 1st appearance
ACK	Acknowledgement	2.2.1.1
AIMD	Additive Increase Multiplicative Decrease	2.4
APIs	Application Programming Interfaces	3.2.4
AQM	Active Queue Management	3.2
ATCP	Ad-hoc TCP	2.2.1.2
ATP	Ad-hoc Transport Protocol	2.3.1.1
DCF	Distributed Coordination Function	3.4
DSR	Dynamic Source Routing	3.2
ECE	ECN-Echo	3.2.2
ECN	Early Congestion Notification	2.2.1.2
FIFO	First Input First Output	C.1
FSM	Finite State Machine	D.1
FTP	File Transfer Protocol	3.4
LRED	Link Layer Random Early Detection	2.2.1.1
MAC	Medium Access Control	1.4
MANET	Mobile Ad-hoc Network	1.1
PI	Proportional and Integral	1.4
PRE	Proactive Route Errors	2.2.1.1
R-DSDV	Random Direct Sequence Distance Vector	2.2.2.1
RFP	Route Failure Reduction	2.2.1.1
RTHC	Round Trip Hop Count	2.2.1.1
RTO	Retransmission Time Out	2.2.1.1
RTS/CTS	Request To Send/ Clear To Send	3.4
RTT	Round Trip Time	1.6
SAFT	Store and Forward Transport	2.3.2.1
SANET	Static Ad-hoc Network	1.1
SRP	Symmetric Route Pinning	2.2.1.1
TCP	Transport Control Protocol	1.2
TCP-AP	TCP with Adaptive Pacing	2.4
TCP-Bus	TCP Buffering Capability and Sequence Information	2.2.2.2
TCP-DOOR	TCP Detection Of Out-of-Order and Response	2.2.1.1
TCP-ELFN	TCP Explicit Link Failure Notification	2.2.1.2
TCP-F	TCP Feedback	2.2.1.2
TCP-PR	TCP for Persistent Packet Reordering	2.2.1.1

List of Symbols

		Section of 1st appearance
A_m	Gain margin of AQM control system	C.1
$C(s)$	Transfer function of the PI rate controller	C.1
$e(t)$	Queue deviation	C.1
$G(s)$	Open-loop transfer function of the AQM control system	C.1
K	Buffer size of the AQM router	C.1
K_P	Proportional gain of PI rate controller	C.1
K_I	Integral gain of PI rate controller	C.1
N	Total number of TCP sources or sessions	C.1
$P(s)$	Transfer function of the AQM control plant	C.1
$q_i(t)$	Controlled transmission rate of the source node i	C.1
$q_i'(t)$	Advertised transmission rate of the controlled source node i	C.1
$u(t)$	Uncontrolled transmission rate of guaranteed traffic flowing into the AQM router	C.1
$win_i(t)$	Controlled window size of the source node i	C.1
$win_i'(t)$	Advertised window size of the controlled source node i	C.1
win_i^{max}	The maximum window size allowed for the controlled source node i	C.1
$x(t)$	Instantaneous queue length of the buffer in the AQM router	C.1
$\overline{\mu}$	Service rate (output link capacity) of the AQM router	C.1
ϕ_m	Phase margin of AQM control system	C.1
τ_{fi}	Varying forward time delay from the controlled source node i to the router	C.1
τ_{bi}	Varying feedback time delay of both from the router to the destination and from the destination to the source node i	C.1
τ_i	Varying round trip time for the controlled source node i	C.1
$\overline{\tau}$	Varying average round trip time for all the controlled sources	C.1

Chapter 1
Introduction

1.1. Overview in Ad-hoc Wireless Networks

Wireless communications continue to experience considerable growth especially due to the fact that many different applications (voice, data and multimedia) are migrating to wireless networks in order to offer a wide range of services to end users. Despite their benefits, wireless communications bring new and strict performance concerns as well as requirements due to the portability and mobility of wireless devices, the wireless access medium, and their open, dynamic and easily accessible network architectures. These concerns are introduced in addition to those of wired networks. Additional factors include the particular functionalities of a network, the difficult traceability and the capabilities (processing and resource management) of their nodes (mobile devices). Therefore, depending on the type of wireless network providing a particular service, one should also consider the tradeoffs between meeting network requirements and reducing the impact in network performance.

Ad-hoc wireless networks, including sensor networks, offer users convenient wireless communications without any infrastructure support, as well as a wide range of applications in areas such as health, military, commercial, command, control, communications, computing, intelligence, survey and targeting systems. Particularly, the demand on Ad-hoc networks is increasing because of their capability for rapid deployment, self-organization and fault tolerance.

Wireless Ad-hoc networks consist of both mobile and static nodes that can freely and dynamically self-organize. Each node may interconnect with its neighbors within its transmitting range and rely on their capability to forward its data (through multiple hops) to a destination. Ad-hoc wireless networks can be categorized in two groups depending on the mobility of their nodes: MANETs (Mobile Ad-hoc NETworks), and SANETs (Static Ad- hoc NETworks).

1.2. Congestion Control in Ad-hoc Wireless Networks

Ad-hoc networks are more prone to suffer from congestion than common wired networks. This is because the capability of a node is not only used to forward messages, it also has to manage and process the resources. An important resource is obviously the limited capacity (spectrum) that has to be shared with other nodes within its transmitting range. The action of one single node, either accidentally or intentionally can collapse the whole network due to traffic overload. Hence, there is an important need for congestion control algorithms that would regulate the overload and allow for the optimal utilization of the capacity in Ad-hoc networks while adapting to the particular characteristics of the wireless Ad-hoc environment. These traffic control mechanisms are important especially in light of a demand for wider range of applications and a convergence of services such as voice, data, and video. These potential applications can range from multi-hop wireless broadband Internet access, remote control and monitoring, to voice and video communications for disaster areas.

Wired and wireless networks are significantly different in terms of bandwidth, propagation delay, and link reliability. Ad-hoc networks bring new problems and challenges to the congestion control issue due to power constraints and to their own multi-hop communication characteristics such as path asymmetry, channel conditions and errors, medium contention, and hidden and exposed stations. Additional problems to control congestion in Ad-hoc networks are caused by mobility and include network partitions and route failures.

The work that has been conducted in Ad-hoc networks to solve the congestion control problem and the challenges mentioned above include different approaches that work in different communication layers. The approaches show solutions that include either (1) the modification within only one communication layer or (2) the feedback of more than one communication layer and the interaction between them. Additionally, each of these solutions could be classified as either end-to-end or hop-by-hop schemes.

In general, most of the research work in congestion control for Ad-hoc networks are either TCP variants, i.e. focusing on modifications and improvements to the mechanisms used by the Transmission Control Protocol (TCP) as the transport layer protocol, or non-TCP variants, i.e. focusing on entirely new protocols far away from TCP.

1.3. Ad-hoc routing protocols overview

Many Ad-hoc routing protocols have been proposed for wireless Ad-hoc networks. A comprehensive and thorough comparison with those protocols is beyond the scope of this thesis. In general, they can be divided into two main categories:

a) Table-driven or proactive protocols, in which nodes periodically exchange routing information in order to maintain the routing table for the entire network (e.g. Destination-Sequenced Distance Vector (DSDV) protocol [PeBh94]).

b) On-demand or reactive protocols in which routes are discovered and maintained when needed. Examples are DSR (Dynamic Source Routing) [JoMa96] and AODV (Ad-hoc On-demand Distance Vector Routing) [PeRo99].

The advantage of proactive schemes is that when a route is needed, there is little delay until the route is determined. However, purely proactive schemes are not appropriate for reconfigurable wireless ad hoc network environment, as they continuously use a large portion of the network capacity to keep the routing information current. Since the node movement may be quite fast and the topology changes may be more frequent than the route requests, most of this routing information may never be used. This results in a further waste of the network capacity.

DSR and AODV are the two most popular on-demand routing protocols used in the literature. Ad-hoc On-demand Distance Vector Routing (AODV) [PeRo99] is improved from DSDV and operates in an on-demand manner such that routes are discovered in case that the source needs to send data to some destination and does not have a route to the destination in its routing table. DSR uses source routing to add route information in the header of the packet so that the intermediate nodes do not need to maintain the routing table and just forward the packet according to the route information contained in the packet header. On-demand routing protocols have better performance than table-driven ones [RoTo99] due to the low routing overhead required in the on-demand route discovery. For this reason, we chose for our analysis two protocols of this type (DSR and AODV).

1.4. Motivations

TCP is considered as the standard transmission protocol for the reliable delivery of packets for the majority of the networking applications among networks nowadays. With the convergence of advanced electronic, wireless technologies and the Internet, ubiquitous computing and pervasive Internet applications using TCP are becoming a trend. In addition, there is always a need to connect wireless networks to the Internet or to other wired networks which also use TCP. Therefore, congestion control algorithms that are TCP-based or TCP-compatible for wireless ad-hoc networks is a natural research topic to pursue.

Even though TCP appears to be a very useful protocol on wireline networks, it has its own deficiencies when implemented on wireless networks. First of all, TCP congestion control algorithm is an end-to-end scheme that assumes the intermediate nodes/links are "reliable" and that implicitly controls congestion by measuring network congestion just at the end nodes. In Ad-hoc networks, this has caused problems in TCP fairness and efficiency due to the unreliable nodes/links due to the dynamic changes of the nodes in the network as a link breaks as soon as two neighboring nodes move out of the transmitting range of each other. Secondly, the current TCP control mechanism leads to packet bursts that can provoke an increase in link contention and congestion and therefore packet drops. All of them cause significant performance degradation. Therefore, we are motivated to identify algorithms that will combat these two deficiencies.

The controller we are looking for should ideally utilize local congestion information in each intermediate node along the path for the control of the transmission rate at each source node. It would control the transmission of data packets according to a computed and updated value at each intermediate node along the path, while preserving TCP backward compatibility. This controller must proactively identify incipient use of the link channel and adjust the source transmission rate. Furthermore, we know that whenever congestion happens in wireless networks, severe medium contention leads to further packet losses and large queue length. By providing and updating a metric to control congestion at each intermediate node, this controller will indirectly reduce contention on the Medium Access Control (MAC) layer. All these have motivated us to investigate a rate controller which, as reviewed later and below, appears to have such capability.

Based on the literature to be reviewed in Chapter 2, we noticed that most of the studies that evaluate rate control algorithms in Ad-hoc networks are based on heuristics, and there is only limited work done based on control theory such as the Proportional and Integral adaptive rate control scheme [HoYa06]. Thus, we would like to extend, implement and evaluate this rate controller for wireless Ad-hoc networks. The TCP compatibility of the Proportional and Integral (PI) adaptive rate controller as well as its adaptability and self-tuning features to network fluctuations makes it appealing for its study on Ad-hoc networks.

Nobody has implemented this PI adaptive rate controller for wireless networks so far, even though its implementation does not require complex modifications to intermediate nodes. Since the simple implementation requirement will make it easily deployable in real networks, we are particularly interested in knowing its feasibility for achieving reliable data transport in static and mobile Ad-hoc networks.

Several researchers identify the interaction of TCP with the underlying routing and MAC layers as a key factor for the poor performance of TCP in IEEE 802.11 multi-hop wireless networks. The adaptive PI rate controller does not rely on modifications on the routing protocols as other approaches. Additionally, most congestion control approaches studied in the literature simply focus on testing their performance under one specific Ad-hoc routing protocol. Therefore, we are interested on evaluating the adaptive PI rate controller under different Ad-hoc routing protocols to see the influence they have on the performance of the controller. Since earlier studies as reviewed reveal that table-driven routing protocols have worse performance due to high routing overhead, we are motivated to investigate on-demand routing protocols.

1.5. Objectives

In general, we are interested in studying the congestion control problem in Ad-hoc networks running TCP traffic. We are specifically interested in the following:

1. Conducting an analytical evaluation of a congestion control/avoidance mechanism based on control theory (not heuristics) for Ad-hoc wireless networks.
2. Extending and implementing the PI rate controller proposed in [HoYa06] to control the network congestion in Ad-hoc wireless networks by regulating the transmission rate in

different source nodes according to the queuing feedback information from the intermediate nodes.

3. Conduct a simulation-based performance evaluation and comparison of the PI rate controller [HoYa06] with TCP New Reno in wireless Ad-hoc networks under various static and mobile scenarios.
4. Demonstrating the capabilities and feasibility of the PI rate controller in achieving reliable data transport in static and mobile Ad-hoc networks.

1.6. Methodologies

To identify and understand the main issues/concerns in implementing traffic control in wireless ad-hoc networks, we conduct a systematic review of different existing congestion control mechanisms proposed for Ad-hoc wireless network, their advantages and shortcomings, as well as the tradeoffs in regard to network performance, complexity and interoperability. All these existing approaches are classified and presented in the next chapter.

Due to the mathematical intractability of the analysis in congestion control in Ad-hoc wireless networks, we have opted for simulation to evaluate the performance of our PI rate congestion control algorithm. Of the many existent network simulators, we have chosen Qualnet [Qual05] because it can provide a complete set of tools for network modeling and simulation. Qualnet provides the source code of different models and their components to build our static and mobile Ad-hoc networks. The modular design of Qualnet allows us to modify protocols at different communication layers, to allow the layers to interact with each other and to create built-in measurements at each of the communication layers when experimenting with different network functionalities.

Different parameters are set on each communication layer in order to create various scenarios to evaluate the network performance of the investigated congestion control algorithm. These parameters are considered in order to avoid lower layer factors from affecting the performance evaluation of higher layer protocols. Therefore, we carefully choose parameters related to the signal reception, path loss, fading, and physical layer models in general. Without proper choice, these parameters could cause simulation tools to yield quite different results even when they are configured with the same set of higher layer protocols [TaMa03].

Instead of designing a completely new transport protocol, we will modify the existing PI-rate controller [HoYa06] to make it compatible to TCP. Then we would like to conduct the following analytical simulation-based performance evaluation and comparison with TCP New Reno, under several different simulation scenarios.

We would like to first evaluate the performance of static Ad-hoc networks (SANETs) as a preliminary investigation of the PI controller performance in multi-hop networks without node mobility. This allows us to verify whether the PI rate controller is effectively controlling the transmission rate of nodes along the paths of each particular session. The problem of packet bursts is intended to be solved by controlling the transmission of data packets according to a computed and updated value at each intermediate node participating on the function of forwarding packets to a destination. This will help by proactively identify incipient use of the link channel. It will be able to adjust the transmission rate and, hence indirectly reduce contention on the MAC layer. The next set of experiments will be performed on MANETs in order to evaluate the performance of our PI rate controller under mobility. We would like to find out how our PI rate controller adjusts itself to the dynamic scenarios of a network.

We use a basic congestion scenario to see how the PI rate controller reacts to congestion within a wireless Ad-hoc scenario. We shall study the time evolution of throughput, queue size, window size and Round Trip Time (RTT) values of a particular source-destination pair. Since throughput is an important measure, we would emphasize the study of the total overall average throughput per session and the aggregated total average throughput in the network for the rest of the scenarios. This performance measure is convenient and more effective for the analysis of the dynamic changes in topology and routes which reflect on different levels of congestion at each different node.

We would also like to conduct a comparison study on the performance of the PI rate controller versus the performance of TCP New Reno under various network environments. This includes the analysis of tradeoffs such as performance and adequacy of the algorithms and their implementation. Of particular interest is the influence from different types of routing protocols. We choose DSR as the main routing protocol used for our analysis while we additionally include AODV for the comparison under different routing protocols. For each scenario, several simulations are run to provide meaningful statistics such as the mean,

variance and 95% confidence intervals. A sample size of 50000 is used for each simulation runs.

1.7. Thesis Contributions

The following are the contributions of this thesis:

1. A flexible implementation and extension of the PI congestion control mechanism in the network simulator Qualnet.
2. Systematic simulation-based performance comparison of the modified PI-rate congestion control algorithm with TCP New Reno in terms of queue size, throughput and fairness.
3. Development and verification of the congestion control algorithm in both static and mobile scenarios.

1.8. Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 presents a background overview in congestion control for wireless Ad-hoc networks. Chapter 3 describes the network operation, configuration and assumptions used in this thesis. The simulation-based performance evaluation and the comparison of the congestion control algorithm with TCP New Reno in different static scenarios are shown in Chapter 4. Chapter 5 shows the same but for different mobile scenarios. In Chapter 6, performance curves of the adaptive PI rate controller in different mobility conditions and under different Ad-hoc routing protocols are presented. Chapter 7 lists future work in congestion control for Ad-hoc networks to finally, conclude this thesis.

Chapter 2
Background

There are different approaches proposed to deal with the problem of congestion control in wireless Ad-hoc networks. This section offers a classification of these works and gives detailed description of the main work related to this thesis.

2.1. Current Research Approaches in Congestion Control

Reliable transport in Ad-hoc networks have been approached either by introducing modifications to TCP aiming to enhance its performance [BaKr97], [ChRa98], [HoVa99], [GeTa99], [MoSi00], [DyBo01], or by the design of entirely new protocols, [LiSi01], [KaVa03].

There are several main arguments for keeping and improving TCP despite its shortcomings (poor bandwidth utilization, network unfairness and performance degradation) over multi-hop networks. These include the facts that TCP/IP is already the standard networking protocol on the Internet; that it is the most widely used transport protocol for data services (FTP, email, WWW browser); and that TCP might remain as major transport protocol for these networks because most wireless applications depend on TCP for communicating with wired hosts nowadays.

On the other side, there are strong arguments that some design elements in TCP are inappropriate for the characteristics and operating conditions of Ad-hoc networks [KaVa03]. Therefore, non-TCP-based approaches are sought for the study and design of entirely new transport protocols. The focus of this thesis will not be centered on this type of approach.

For either the TCP or the non-TCP approaches, there are two types of schemes: end-to-end and hop-by-hop. In an end-to-end scheme the transmission is controlled by the source and destination nodes of the connection while intermediate nodes are not involved in the transport process except for running the routing protocol to forward data. The research on end-to-end schemes mainly focuses on the scalability and deployability [RaKl05]. However, they could

sometimes result in large transient overloads (due to delayed feedback or link failures) at a single node [YiSh04].

In a hop-by-hop scheme the data is transferred one hop at a time. Here the intermediate nodes play an important role because new functions are added. The intermediate nodes must be capable of storing, processing, routing and forwarding data. Their main function is to detect congestion (long queue length) and to take action on a hop-by-hop basis. It prevents starvation of nodes and requires per-flow state management in intermediate nodes which can be feasible because wireless networks usually have fewer flows per node than the wired Internet [YiSh04].

In a different classification, cross-layer communication is considered for the design of congestion control [Chia04] as opposed to a layered approach where just one layer is consider to improve the congestion control mechanism.

The layered approach relies on adapting the communication layers independently from other layers. This could be done by adapting TCP to routing changes without the use of feedback from other network layers or by enhancing lower layers' mechanisms such that these could be transparent to TCP sources. Any lower layer problem specific in Ad-hoc networks should be hidden to TCP sources. This idea aims to maintain the end-to-end mechanism of TCP to seamlessly interconnect with the wired internet. The work related to this approach includes the modification of any of the transport [AlJi03], [BiNi05], [SuGr05]; network [BoDa03], [CoDa03], [LiXu03]; or link layers [FuZe03], [XuGe03], [YaSe03], and [YiSha04].

On the other side, the cross layer approach includes the use of feedback information from lower layers to help the transport layer perform better under network congestion [ShRa03]. This is dealt with the detection, control and corresponding reaction to particular network conditions such as channel errors, link contention and route failures. By exploiting this information, the network itself will be able to determine the current load situation and protect itself from congestion [ShRa03]. There can be different combination of layers involved in this approach including application and network layers [ZeShu06]; transport and network layers [Ch98], [KiTo01], [LiSi01], [HoVa02], [Yu04], [HoXi05]; transport and link layers [LiSt05]; transport and physical layers [KlKr03]; as well as network and physical layers [Go01].

Despite expecting better performance than layered solutions; cross layer approaches are more complex to implement and design. Their implementation requires at least the modification

of two communication layers, thus violating layer transparency that keeps a layer independent of other non-neighboring layers [Chia04]. The arguments that support layered approaches consider that designing independent protocols for each communication layer will create long term solutions.

The selection of the type of solution depends on the priority given to either the performance optimization or the architecture design. The performance optimization can lead to short term benefits while an optimal architecture design could carry long term considerations. Table 1.1 shows a classification of the current research to address the issue of congestion control in Ad-hoc networks described in the section above.

Table 1.1: Classification of Current Approaches to Address Congestion Control in Wireless Ad-Hoc Networks (SANET & MANET)

Classification Of Current Approaches In Congestion Control For Ad-hoc Networks (SANET & MANET)			
TCP-based (Section 2.2)		Non TCP-based (Section 2.3)	
End-to-end schemes	Hop-by-hop feedback schemes	End-to-end schemes	Hop-by-hop feedback schemes
• Single layer solution • Cross-layer feedback solution	• Single layer solution • Cross-layer feedback solution	• Single layer solution • Cross-layer feedback solution	• Single layer solution • Cross-layer feedback solution

2.2. TCP-based Protocols

Approaches to enhance TCP performance in Ad-hoc networks include (1) solutions that can modify TCP's response without the use of explicit feedback, (2) solutions that incorporate network feedback information into their designs, and (3) solutions that tune lower layers behavior in order for TCP to operate normally while leaving TCP unmodified. These solutions can either use end-to-end or hop-by-hop approach, as will be shown in the following sections.

2.2.1. End-to-End Protocols

2.2.1.1. Single-Layer Approaches

These approaches addressed the problems of TCP at one of the OSI layers by making use of heuristics in order to extract information and react to packet drops and duplicated acknowledgement packets (ACKs).

There is some work that focuses on *distinguishing between losses due to route failures and those due to network congestion.* In a transport layer solution [DyBo01], a "Fixed RTO (Retransmission Time Out)" value is defined so that two consecutive timeouts are considered to be an indication of route failure. Whenever this happens, the sender assumes route failure rather than network congestion. Unacknowledged packets are transmitted again without doubling the RTO. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged. In TCP-DOOR (TCP Detection of Out-of-order and Response) [WaZh02] the transport layer solution does not require explicit notification from the routing layer nor cooperation of other nodes to detect route failures. A failure is assumed whenever the source or destination receives out of order packets. This solution outperforms the Fixed RTO [DyBo01] but at the cost of more modifications to the mechanisms of TCP. The solution is entirely based on TCP header options added to the ACK packets at the receiver. Unfortunately, out of order events can not only be attributed to route failures and more important they vary according to the routing protocol used. In the TCP-PR (TCP for Persistent Packet Reordering) approach [BoHe03], there is no information extraction about the order in which segments or ACKs arrive to determine the state of the network. Instead, it just relies on timers. TCP-PR does not require changes at the receiver and it works without special TCP header options. It was designed to fit at multi-path routing protocols. In TCP-PR the sender maintains segments in one of two lists: either in to-be-sent or in to-be-acknowledged. When a segment drop is detected, the segment is moved from the to-be-acknowledged list back to the to-be-sent list. Loss of segments is discovered by the use of a timer. Whenever a packet is placed in the to-be-acknowledged list, it is time-stamped and the current value of the congestion window is saved with the segment as well. If a segment remains in this list for more than an estimated maximal possible round-trip time, it is considered dropped. This estimation is updated continuously. In another approach [ChXu03] working at the transport layer that does not require feedback information, the protocol adaptively sets a congestion window limit according to the current round-trip hop-count of the path. Its main idea comes out from the observation that TCP often over-shoots, causing to overload the network and to increase contention at the MAC layer. In order to fully

utilize the capacity of a network, TCP flows should set its congestion window limit no greater than the bandwidth-delay product[1] nor the round trip hop count (RTHC) of the path in order to avoid network congestion. This resulted in a tighter upper bound (approximately 1/5 of the round trip hop count of the path in a chain topology) that helps to determine the maximum RTO which enables TCP to probe the route quickly to detect link failures.

There are other proposals e.g., [LiXu03] and [AnPa03] that rely on network layer enhancements and that mainly focus on the issue of *reducing the number of routing failures.* This is particularly important because routing failures usually induce long idle times which affect to TCP performance. "Backup Path routing" [LiXu03] is used to improve the availability of a TCP path and connections by storing the information of an alternate path that can be used whenever a routing failure is detected. A network layer enhancement framework [AnPa03] called ATRA aims to improve TCP performance. It consists of 3 mechanisms that work at the MAC and routing layers. These are the Symmetric Route Pinning (SRP), Route Failure Prediction (RFP), and Proactive Route Errors (PRE) that can respectively minimize route failures probability, predict route failures occurrence and minimize the delay on route failure information back to the source.

Other proposals mainly work on *addressing the problem of contention in the wireless channel.* A transport layer approach [AlJi03] addresses this problem by decreasing the number of TCP ACKs transmitted and adaptively delaying them. The authors studied the impact of delaying ACK packets concluding that in a chain topology, improvement may be achieved by delaying 3-4 ACK packets (except at the startup) if the sender timeout expires. The sender uses a fixed interval and does not react to packets that are out-of-order. Their results are positive but limited to a single flow only. Among the approaches suggested to reduce contention but that are based on lower layer enhancements we find a network layer approach named "COntention based PAth Selection" [CoDa03] which address the problem by using disjoint forward and reverse routes, and by dynamically updating these routes based on the contention level. Two link layer enhancements called LRED (Link Random Early Detection) and Adaptive Pacing are proposed to address contention in the wireless link [FuZe03]. Their function is to detect

[1] It is defined as the product of the forward path bottleneck bandwidth and the packet transmission delay in a round trip time.

contention and increase the back off time at the link layer before packet transmission. The Adaptive Pacing technique aims to distribute traffic on the link layer among intermediate nodes in a more balanced way while LRED aims to throttle TCP senders when incipient congestion is detected.

Finally, there are proposals that mainly deal with *improving fairness.* "Neighborhood RED" is a network layer approach which defines a distributed neighborhood queue which contains the node local queue as well as the upstream and downstream queues of its one-hop neighbors [XuGe03]. RED is then applied to this queue in order for TCP to regulate its transmission rate and to control average queue size at nodes whenever it sense packets being dropped. It throttles TCP senders when incipient congestion is detected and improves fairness by purposely dropping TCP packets on intermediate nodes. Non Work-Conserving Scheduling [YaSe03] is also a link layer proposal but it addresses unfairness by penalizing greedy nodes with high output rate, and by increasing their queuing delays at the link layer.

2.2.1.2. Cross-Layer Feedback Approaches

In this section, the approaches studied explicitly employ information from lower layers to improve TCP performance while offering an end-to-end connection. The majority of these approaches mainly deal with *distinguishing between losses due to route failures and those due to network congestion* by means of cross layer feedback between the transport and network layer. "TCP Feedback" (TCP-F) [Ch98] uses feedback information from intermediate hops which send route failure notification packets to the source in order to freeze transmission without reducing window size until the sender receives a route re-establishment notification. "Explicit Link Failure Notification" (TCP-ELFN) [HoVa02] is similar but unlike TCP-F the sender, while on stand-by mode, periodically starts probing the network with a small packet to see if a new route has been re-established. If there is one, the stand-by mode ends, and the sender restores its RTO to continue working normally. "Ad-hoc TCP" (ATCP) [LiSi01] propose the insertion of an intermediate layer below TCP that interacts with the network layer and puts the TCP sender into either persist, congestion control or retransmit state, respectively, whenever the network gets disconnected or according to packet drops due to route breakage, network congestion and high bit error rate. It does not require any additional modification, as it

reacts according to TCP, ECN (Early Congestion Notification), and ICMP message information.

"Joint Congestion and Media Access Control" [LiSt05] is a cross-layer approach between the transport and link layer whose aim is to *address contention on the wireless channel.* It makes use of contention graphs and matrixes to formulate resource allocation as a utility maximization problem. MAC layer imposes contention prices based on local aggregate source rates and TCP sources adjust them based on the prices.

A cross-layer approach between the application and network layer is introduced in [ZeShu06] to *adaptively control the transmission rate* for video transport over wireless Ad-hoc networks by adapting the video encoder's output bit rate to the available network bandwidth in order to improve the quality of the receipt video. It determines a target bit rate according to the congestion in the network as well as based on routing information. It introduces an interesting alternative to avoid including losses due to routing failures or channel contention. However, it does not offer and effective way to determine the available bandwidth, nor a way to tune the optimal bit rate maximum and minimum values to run in their algorithm.

2.2.2. Hop-by-Hop Protocols

These protocols use a hop-by-hop scheme to address the congestion control problem in multi-hop Ad-hoc wireless networks. They do this by running the protocols at each of the intermediate nodes within the network. In the majority of cases, the use of cross-layer communication mainly at the transport and routing layer is required to enable hop-by-hop transfer. All the protocols of this category require modifications to lower layers as well as to intermediate nodes.

2.2.2.1. Single-Layer Approaches

A link layer scheme that offers hop-by-hop congestion control using an optimized time-based MAC constraint to access the channel is introduced in [YiSh04]. It makes use of two types of constraints: a link constraint and a time constraint. The link constraint avoids that the sum of data rates of all sessions traversing a link is greater than the capacity of the link. The time constraint restricts the existence of only one instance of communication at a given node at any

instant of time. [SuGr05] is a transport layer scheme for wireless multi-hop networks that aims to improve fairness by adopting explicit congestion control in intermediate nodes.

In each intermediate node an estimation of congestion is done. Then, based on this estimation the senders' transmission rate is determined.

"Random Direct Sequence Distance Vector" (R-DSDV) [BoDa03] is a routing layer approach that introduces a randomized protocol to reduce congestion by allowing overloaded nodes to tune local parameters related to the routing protocol so as to probabilistically divert traffic to other paths. It propagates the routing messages according to a routing probability distribution, instead of on a periodic basis as in basic DSDV[2] to reduce the additional message traffic and the extra overhead. In this way an overloaded node along a path could influence the routing decision taken by sources of the traffic through that node by using a metric that reflects the congestion level of a node. R-DSDV outperforms DSDV by reducing the average queue size associated with each mobile node and hence the average packet delay. R-DSDV enables each node with a randomized decision to decide whether or not to forward/drop a packet, and what type of packet to forward/drop. Each node maintains a congestion value, which can be static or dynamically assigned. When a packet arrives, the current queue size is checked and if it is above a given congestion threshold the packet is either dropped or queued according to a probability.

2.2.2.2. Cross-Layer Feedback Approaches

"Split TCP" [Ko02] is a TCP scheme that aims to reduce route failures that causes long idle times mainly due to node mobility. It works by splitting long connections into short segments at intermediate nodes in order to decrease the number of routing failures. Whenever a node determines there is an intermediate node within a distance parameter, it intercepts TCP packets that need to pass through that intermediate node, buffers them and acknowledges them to the previous hop with a local acknowledgement. Later it forwards the buffered packets to the next

[2] DSDV [PeBh94] is a table-driven Ad-hoc routing protocol in which nodes periodically exchange routing messages in order to maintain the routing tables for the entire network. It guarantees loop-free operation. DSDV tags each route with a sequence number and considers a route more favorable than other if it has a greater sequence number, or if the two routes have equal sequence numbers the one that has a lower metric. When a node finds that its route to a destination has broken, it advertises the route with an infinite metric and a sequence number greater than the one for the route that has broken.

hop. Whenever a local acknowledgement is returned, the node will free the packet from its buffer. The destination sends an acknowledgement back to the source to guarantee end-to-end reliability. The source deals with two windows: a congestion window for the connection to the next intermediate node and an end-to-end window for the total connection to the destination.

"TCP Buffering capability and Sequence information" (TCP-BuS) [KiTo01] is another approach that uses the capability of intermediate nodes to buffer packets whenever a route is down. It makes use of messages in order to explicitly notify about route disconnections and successful route re-establishments. This is done by the sender of the message checking whether it can overhear the next hop forwarding the message in the appropriate time.

2.3. Non-TCP based Protocols

There are other protocols proposed to address the congestion control problem in Ad-hoc networks that are non-TCP-based. They are based on the idea of designing completely new protocols that fit the specific conditions of wireless Ad-hoc scenarios. These protocols offer end-to-end connections but require all intermediate nodes to run the same protocol. This becomes a great limitation for their deployment and interoperability with other networks. Additionally, they are complex and difficult to implement, especially when communicating with networks running TCP.

2.3.1. End-to-End Protocols

Existing work on end-to-end protocols appear to be confined to single-layered approaches as follows.

2.3.1.1. Single Layer Approaches

"Ad-hoc Transport Protocol" (ATP) [KaVa03] is a reliable Transport Protocol for Ad-hoc Networks that does not share the fundamental principles of TCP. It uses the information feedback of intermediate nodes to estimate the transmission rate and retransmissions. It detects congestion by means of delays; therefore, the ambiguity of distinguishing congestion losses from non-congestion losses is not an issue here. ATP has no retransmission timeout, and the functions that provide congestion control and reliability are separated from each other. It achieves better performance in regard to delay, throughput, and fairness than any TCP variants.

However it is not a good solution for the integration to the Internet or many other wireless networks. [AnPa03] presents TCPA, a radical approach of a window-based protocol quite different from TCP. It employs ELFN messages similar to other approaches described before. However, no performance evaluation results are given.

2.3.2. Hop-by-Hop Protocols

Unlike the end-to-end protocols, existing work on hop-by-hop protocols appear to be confined to cross-layered approaches as follows.

2.3.2.1. Cross-layer Feedback Approaches

"Store and Forward Transport" (SAFT) is a framework for transport protocols that aims to be flexible to route changes and failures by transferring data one hop at a time [HeBa06, HeBa05]. It is mainly a cross layer scheme between the transport, network and MAC layers. It uses a hop-by-hop approach in two sub-layers by store-and-forwarding every incoming segment to the next hop until an acknowledgement is received. It requires the state of the link to any single-hop neighbor and the queue length as cross-layer information.

2.4. Main Related Work

The main interest of this thesis is on congestion control over multi-hop wireless Ad-hoc networks, which has been an active research area over the past few years (see. e.g., [Ch04], [YiSh04], [XuGe03], [FuZe03], [WaZh02], [CgRa01], [DyBo01], and [HoVa99]). Unlike most of previous work described in the section above, this thesis will focus on the interaction of congestion control at the transport layer in order to control the transmission rate.

As mentioned before, congestion control is critical to provide for efficient and fair allocation of network resources among communication flows. In the Internet, this concern issue has been addressed by applying end-to-end congestion control algorithms (i.e., TCP, TFRC [ChNa04]) that could be either window based or rate-based protocols. Several studies have showed that TCP do not perform well in a multi-hop wireless environment and suffers fairness and efficiency problems [ChNa05], [FuLu05], [NaHe05], [KhSh04], and [JoRi01].

As presented before, many flow control schemes have been proposed for Ad-hoc networks. Some of them adopt flow control by implicit measures of congestion at the end-nodes to

dynamically infer the network bandwidth they will share. Others explicitly measure congestion by relying on network nodes feedback information.

TCP's AIMD (Additive Increase Multiplicative Decrease) algorithm is a representative example of implicit flow control. There are some disadvantages it has when applied in Ad-hoc networks including the additive increase of the TCP's congestion window, which limits the ability to quickly use the available bandwidth especially after re-routing events. TCP tends to keep the queues of bottleneck nodes full due to its reaction to packet loss, this causing long queuing delays and leading nodes to drop packets as the bandwidth of a wireless link varies. In addition, the TCP's window-based transmission can lead to packet bursts.

There are some approaches that use explicit rate control to improve the performance of reliable data transport in Ad-hoc networks [HeBa06], [HeBa05], [HoXi05], [RaKl05], [ChNa04], and [SuAn03]. Most of these are non-TCP protocols, and they transmit packets at a calculated rate instead of sending new packets after old packets have been acknowledged.

From these approaches, [HeBa06] and [HeBa05] present a cross layer scheme that uses a hop-by-hop approach on two sub-layers. It is based on the store and forward of every incoming segment to the next hop until an acknowledgement is received. It requires the state of the link to any single-hop neighbor and the queue length as cross-layer information.

On the other hand, [RaKl05] introduces an end-to-end congestion control algorithm named TCP with Adaptive Pacing (TCP-AP) that uses rate-based scheduling for the transmissions within the TCP congestion window. Here the TCP sender adaptively set its transmission rate based on an estimate of the current 4-hop propagation delay[3] and the coefficient of variation of recently measured round-trip times. It considers dynamic delaying of acknowledges as in [AlJi03]. This algorithm requires slight modifications on the transport layer only and does not require cross-layer information from intermediate nodes along the path. It does not rely on modifications on the routing layer [XuGe03] and link layer [FuZe03] nor on extra communication between neighboring nodes, but relies on the end-to-end measurement of the interference experienced by a TCP connection.

[3] The *4-hop propagation delay* describes the time elapsed between transmitting a TCP packet by the TCP source node and receiving the packet at the node which lies 4 hops apart from the source node along the path to the destination.

Two non-TCP-based transport protocols for multi-hop wireless networks are introduced in [SuAn03] and [ChNa04]. Both protocols employ pure explicit rate-based packet transmission, where the transmission rate is determined using feedback information particularly in this case from intermediate nodes along the network path. The transmission rate can be dynamically adjusted [SuAn03] according to the maximum packet queuing delay while in a cross-layer scheme [ChNa04] feedback information from both the MAC and the routing layer is used. The sending rate of all the flows is included to each data packet as a special control header.

Note that the algorithm in [RaKl05] is different from those of [HeBa06], [HeBa05], [ChNa04] and [SuAn03] because it complies with the definition of TCP and does not use any cross-layer information from intermediate nodes along the path. It focuses mainly only on the TCP performance and fairness in static wireless networks.

A rate-based transport layer control scheme which uses channel occupancy ratio as sign of network utilization and congestion is proposed in [HoXi05]. This approach makes use of cross-layer information from network layer. Each forwarding node allocates channel resources to passing flows by monitoring and modifying feedback packet fields according to its channel occupancy.

The algorithm in [LiSt05] touches on two protocol layers. 1) the network layer at the intermediate nodes calculates link congestion prices based on local aggregated source rates. It mainly uses contention graphs and matrixes to formulate resource allocation as a utility maximization problem. 2) the MAC layer imposes contention prices based on local aggregate source rates and a TCP source adjusts them based on the aggregated prices in its data path. This work has limited performance evaluation done in simple static scenarios.

Chapter 3
Network Operation, Models and Assumptions

This chapter provides the general communication network operation mechanism as well as the network configuration used for the evaluation of the rate congestion control in Ad-hoc scenarios. The characteristics of each scenario, as well as the assumptions used in this thesis will be presented. Since all the simulation modeling is developed using Qualnet network simulation tool [Qual05], a detailed description of the network simulation modeling in Qualnet is described. This chapter additionally presents the properties of the adaptive PI rate control algorithm [HoYa06] and its implementation. The modifications to the two main layers required for the adaptive PI rate control algorithm implementation (transport and network layers) will be discussed, and the interaction with the intermediate nodes will be also briefly described.

3.1. Network Components and Operation

We consider in general Ad-hoc networks consisting of a number of geographically distributed source/intermediate/destination nodes using TCP. The studied networks are limited both in its dimension and in the number of nodes. We considered common low density networks with an area of 0.49 square kilometers (700m x 700m) and with less than one hundred nodes. Different topologies are to be described in later sections.

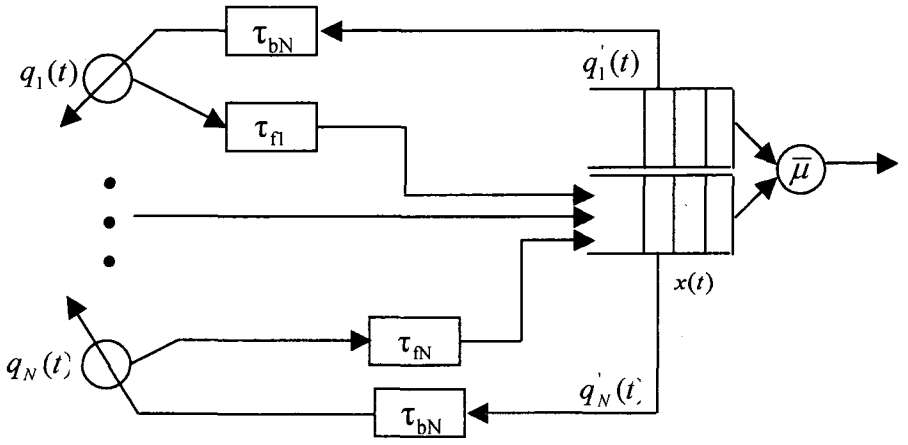

Figure 3.1: Model of an Intermediate Node with the PI rate Controller.

Traffic is generated at every source node and integrated into the IP packets to be delivered to their destination through a sequence of intermediate nodes that act as routers. Figure 3.1 depicts an intermediate node with N traffic flows (each with a controlled source). The intermediate node has a finite buffer space to store the incoming packets and an output link to serve them at a constant data rate μ. The buffer in the intermediate node is organized into a logical queue whose service is assumed first-input first-output (FIFO). There are two kinds of time delays, both considering the propagation, queuing and processing delays. The sum of these two delays $\tau_i = \tau_{fi} + \tau_{bi}$ defines the varying round trip time τ_i of the controlled source node i: (1) the forward time delay τ_{fi} from the controlled source node i to the destination node; (2) the feedback time delay τ_{bi} from both the intermediate node to the destination node and from the destination node to the controlled source node i.

An adaptive PI rate controller is implemented in each network node. It can calculate a desirable source transmission rate $q_i'(t)$ based on the instantaneous queue length of the buffer in the intermediate node, and advertise it as a window size to the source through the IP and ACK packet so that the source can regulate its current transmission rate $q_i(t)$.

The adaptive PI rate controller is based on the one proposed in [HoYa06] which employs classical control theory and adaptive control in order to achieve good performance of active queue management (AQM). Our controller self-tunes whenever the number of active controlled source nodes changes or the average round trip time becomes longer. We make use of these capabilities for Ad-hoc scenarios where the dynamic of the networks (e.g. topology and mobility) are high.

3.2. Implementation of the Adaptive PI Rate Control

The algorithm is implemented in all the nodes along a data path and can calculate a desirable source transmission rate based on the instantaneous queue length of the buffer at each node. By controlling the transmission rate, it maximizes the bandwidth utilization and network resources, while adapting and adjusting to the network fluctuations.

Our flow control scheme in Ad-hoc networks explicitly provides rate and congestion information back to the flow source by including and updating this information in the TCP

header of each data packet going to the destination. The information in the header is modified by intermediate nodes as they are able to detect and react to congestion. The source node then adjusts its data sending rate based on the feedback information received from the destination node. The intermediate nodes will advertise and update the transmission rate as a window size to the source through the IP and ACK packets so that the source can regulate its current transmission rate. Note that although congestion control information is computed along the path, the overall control is end to end based on the transmission rates advertised by all the controllers. More details about the functionalities of the PI rate controller are included in Appendix C.

For the implementation of the PI rate control algorithm, we consider the modification of three main communication layers: the network, the transport and the application layers. Particularly, this requires the modifications to the protocol's code of IP, TCP and FTP.

The adaptive PI rate congestion control algorithm is mainly implemented at the *network layer* and located in every node. At the *transport layer*, we use and modify TCP New Reno [Jaco90] to maintain an end-to-end communication and control through a number of different intermediate nodes. TCP New Reno is one of the most widely adopted TCP variant implementations. TCP New Reno includes the Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery mechanisms]. The source nodes in this layer are in charge of the generation and sending of data packets. These packets are passed to the *network layer* by encapsulating the packet with the IP header containing the source and destination addresses and other control information. They are then delivered to the first intermediate node on their way to their destination through a sequence of intermediate nodes by means of an Ad-hoc routing protocol also at the *network layer*.

At the *network layer*, the intermediate nodes will forward packets according to the destination address they get from the IP header of the packet (provided that the packet is not discarded). This forwarding function is mainly performed according to the Ad-hoc routing protocol that it is used, which in the case of this thesis is Dynamic Source Routing (DSR) [IETF04]. DSR is an on-demand source routing protocol that do not make use of intermediate nodes to update the routing information. It consists of the route discovery and the route

maintenance phases. It has multiple route cache entries for each destination. AODV is another on-demand Ad-hoc routing protocol used in this research that makes use of distance vector routing. It adapts well to highly dynamic topologies because each node contains routing table information of the next hop to any other node. The nodes ping with "hello" messages to test the links and therefore being able to notify of topology changes. The Ad-hoc routing protocol used is in charge of finding a route to any other node if there is a route to it. Every intermediate node along the route knows the next hop towards the respective destination. Upon reaching the destination node, the header of the IP-packet is stripped and passed to the transport layer where the destination node of the end-to-end pipe will generate the corresponding ACK packet to be sent back to the source node.

To achieve the above functionalities, we also modify the queuing and scheduling mechanisms in the *network layer* so that the header of a passing IP-packet is updated by a calculated number of packets that the sender will be able to send. Note that although the routing protocol has the capability of providing accurate information about the state of the next hop link, this capability is not integrated in our congestion control scheme in order to properly evaluate the performance of the congestion control algorithm alone.

3.2.1. Qualnet Simulation Models

We have modified and made use of the source code of different communication models in Qualnet to build and experiment network functionality in static and mobile Ad-hoc networks from which we can analyze the results of our experiments in the comprehensive environment provided by Qualnet. Qualnet is composed of a set of six tools with components for network modeling and simulation (Scenario designer, Animator, Analyzer, Protocol Designer, Packet Tracer and Simulator). The modular design of Qualnet is used to implement our adaptive PI-rate algorithm by modifying protocols in the network and transport layers and allowing cross layer communication between them ant the end-nodes. Built-in measurements at each of the communication layers are also created when needed.

3.2.1.1. Network implementation

We make use of the *Scenario Designer* to create and design each of the topologies. The Scenario Designer allows us to model the network, to include the different network nodes as well as to set their configuration in each communication layer according to the applications (FTP sessions), the transport layer protocol (TCP New Reno), the network layer protocols (IP, queuing and scheduling functions, DSR), and the MAC layer protocol (802.11). In order to modify the above protocols and fulfill the implementation of the PI rate controller, the *Protocol Designer* is used to create new modules with the appropriate functions and to modify existing functions within the Qualnet code of each of the above protocols. After the design and coding is done, the *Animator* was initially used to execute and animate the experiments. Similarly, the *Analyzer* was used to display graphical statistics of the simulation results. However, because the processing time required by the Animator and the Analyzer is long, we switched to run the experiments on the command line interface provided by Qualnet. We additionally created scripts that allow us to parse the raw results information. Finally, for the simulation results analysis, we displayed the information using Microsoft Excel application. For the case of deep analysis we used the *Packet Tracer* in order to understand a particular network behavior. The details of Qualnet six main tools can be summarized in Appendix D.

As mentioned above, our network model considers the modification of the network layer, the transport layer and the application layer. Particularly, this requires the modifications to the codes of the IP, TCP and FTP protocols.

At the network layer, the IP protocol is modified in order to process the incoming packets, to access the transport header, to get the respective fields of interest in such way that the advertised window size could be calculated and finally to sent the packet to the transport layer again. At the transport layer, the TCP protocol is modified so that the congestion control mechanism and the congestion window size could be adjusted according to the updated value of the calculated advertised window size coming from the network layer for each packet.

The simulation in Qualnet works as a collection of network nodes, each with its own protocol stack parameters and statistics. It mainly makes use of messages as the unit defining the interaction between protocols and between nodes. Whenever a new message is received, an

event counter is initialized and sent to an event queue where the message will be scheduled according to the "layer type" field value of the respective message.

The network model includes all the possible states in which the TCP source nodes could be (initialization, idle, ACK, and finalization). For example, when the simulation begins, the system starts with the initialization state first, and the simulation is initialized. Then the system goes into the wait state. If the TCP source node receives an ACK, then it enters the ACK state to respond accordingly.

Additionally, the IP and queuing models in network layer will deal with the process of queuing, scheduling and handling the packet. After receiving a packet from the transport layer and if the server is not busy, the packet will be inserted in a queue. Each node deals with three different types of queues used for different purposes. Incoming packets from neighboring nodes are en-queued in the input queue. Node originated packets are introduced in the CPU queue. The packets of both queues are then moved to the output queue after the routing decision is made by a routing policy, and before the packets are sent to the next hop. After being successfully inserted the packet, the server finishes processing the packet and if no more packets are waiting for service, it enters into an idle state until next event is present.

Most of the metrics in this implementation are made by measuring bytes instead of packets. This is to efficiently calculate and measure the information (window sizes, bytes sent and received to calculate throughput, queue size, etc.) along the different communication layers. In the transport layer, the window sizes consider bytes because TCP keeps its own buffer after the arrival of data packets from the Application layer. This buffer allows TCP to fragment the data collected in its buffer into packets of different size as needed, for processing them in lower layers. Thus, we measure more accurately by checking the number of bytes sent and received in each particular node, rather than the packets, as for this case we might need to rebuild the fragments at each step along the path. The only case where the number of packets is considered is at the network layer by using a counter that increases whenever a piece of information arriving to a node is completely dropped or rejected. In this particular case, the information dropped is considered as packets, because no treatment or consideration to the information for measurements (throughput, queue size, etc.) is done at all.

3.2.1.2. Controller Implementation

This section describes the data types with which the PI rate control algorithm operates with, the interaction of the layers and the data management. More details regarding the tasks of each type of node and of each separate layer will be thoroughly discussed in later sections.

3.2.2. Data Types

The data types used include messages, segments and packets. Messages are received from the application layer and they have no limited size. They are split into segments and again reassembled upon reentering the application layer on the target destination node. The segments are used by all nodes along a route from the source node to the respective destination node. They are composed of packets that correspond to data packets which are managed by the underlying network layer. Prior to sending data, segments are split into packets which will be processed by the network layer.

For the PI rate control algorithm implementation the format of the segment headers have been modified. It uses the regular TCP header plus three new fields. A C data structure was defined for the TCP header definition. Table 3.1 lists the regular header fields and flags, as well as those fields added for this particular implementation. The three new fields added to the packet's TCP header are: (1) the explicit rate converted to a window size value and whose initial value is set as win_max, (2) a timestamp in order to calculate the Round Trip Time (RTT), and (3) the value of the RTT estimated by the source using the method proposed by [HaFl03] and which later will be sent to the network layer.

Each data packet carries this special TCP header, which is modified by the intermediate routers to update the flow's sending rate. When the packet reaches the destination node a timestamp is registered and the rate information is explicitly returned to the sender in a feedback packet. In this way the rate information is updated along the path and returned to the sender within one RTT.

Only the round trip time between the source node and its destination node is needed in order to design the rate-based PI controller and obtain the advertised source transmission rate. The allowed sending rate of a flow is initially set at the sender as its maximum requested rate.

Subsequently the transmission rate is either reduced or modified by the intermediate nodes according to their window size calculations. This is mainly elaborated in order to communicate a possible rate reduction along the path in case of congestion.

Table 3.1: TCP Header Structure

TCP header: struct tcphdr{		
Header Field	Data Type	Description
th_sport	unsigned short	source port
th_dport	unsigned short	destination port
th_seq	tcp_seq	sequence number
th_ack	tcp_seq	acknowledgement number
th_x2:4	unsigned int	(unused)
th_off:4	unsigned int	data offset
th_flags:8	unsigned int	*
th_win:16	unsigned int	window
th_sum	unsigned short	checksum
th_urp	unsigned short	urgent pointer
th_timestamp	float	timestamp
Added fields for PI rate control implementation:		
th_adv_wnd_size	float	advertised window size
th_rtt	float	round trip time

th_flags*	
TH_FIN	TH_PUSH
TH_SYN	TH_ACK
TH_RST	TH_URG

TCP header's Bit 9 in the reserved field is designated as ECN-Echo (ECE) flag while Bit 8 as the Congestion Window Reduced (CWR) flag.	
TH_ECE (ECN flag)	TH_CWR (ECN flag)

It is important to mention that the PI rate control algorithm is able to work as a separate flow control module that can be attached to any Ad-hoc routing protocol module.

3.2.3. Data Transmission

A node can have any of the following roles to handle data in each connection: the end-node (source node or destination node) behavior and the intermediate node behavior.

3.2.3.1. End-Node Behavior

The end-nodes' behavior corresponds to that of the source and destination nodes.

Source:

Functionality at the transport layer

For (each arriving ACK)

a. extract the advertised window size from the TCP header

b. update the window size according to the advertised window size value received by the destination

Figure 3.2: PI Rate Control Algorithm (Source Node)

Figure 3.2 and the following describe the ***source node behavior*** and the main functionalities:

- Initially, the source node is allowed to send out packets using an initial sending rate.
- Once the first data packet is received and acknowledged by the destination node, the source node gets the advertised window size field value on the reception of each ACK packet. Then, according to the advertised window size value included in the header field of the feedback packet it adjusts its transmission rate.
- Whenever the source sends out a data packet, it will register the timestamp on the timestamp header field of the respective packet.
- Whenever the source receives an ACK packet acknowledging the last packet sent, it registers the timestamp when it receives the packet and gets the timestamp field value from the header in order to estimate the round trip time. This is done for every packet sent into the network.
- After this round trip time estimation is conducted, it registers the round trip time value in the RTT header field before sending it to the network layer. The RTT value will be then used in the network layer by the intermediate node, as described in Section 3.4.2.2.

Figure 3.3 and the following describe the ***destination node behavior*** and main functionalities:

- After a data packet reaches its destination successfully, and on the reception of it, the destination node extracts the advertised window size header field value from the incoming data packet.

Destination:

Functionality at the transport layer

For (each arriving packet)

a. extract the advertised window size from the TCP header

b. copy the advertised window size into the ACK TCP header

c. send out the ACK back to the appropriate source node

Figure 3.3: PI Rate Control Algorithm (Destination Node)

- The destination copies the value of the advertised window size header field into an ACK packet for sending it back to the source node. In such way, the packet's fields including the rate feedback information are set to the value of the advertised window size of the incoming data packet.
- In order to indicate the flow rate of each flow, the value of the advertised window size is adjusted in each intermediate node to the smallest value calculated along the route path. However, the destination node just extracts and copies it into the ACK packet.

3.2.3.2. Intermediate Node Behavior

In Ad-hoc networks, normally a packet sent by the source nodes is routed through a series of intermediate nodes before it reaches the destination node. The intermediate nodes play a main role in executing the PI rate flow control algorithm. Figure 3.4 and the following describe the ***intermediate node's*** behavior and main functionalities:

- Intermediate nodes keep track of current flows and their respective transmission rates. They modify the flow control header of each data packet to explicitly communicate a flow's allowed transmission rate.

Intermediate node:
Functionality at the network layer

[1] for (each arriving packet)
  a. if the instantaneous queue size >= buffer size
    i. discard the packet
  b. else
    i. extract round trip time value from the TCP header
    ii. update the queue size
    iii. update K_p and K_I values to self tune the controller
    iv. calculate the advertised window size
    v. if (advertised window size < previous advertised window size value)
      1. update advertised window size in TCP header field
    vi. else
      1. keep previous advertised window size value

[2] send out the packet to next intermediate node or destination

[3] select the corresponding ad hoc routing protocol to forward packet

Figure 3.4: PI Rate Control Algorithm (Intermediate Node)

- On receiving a data packet, the intermediate nodes get the round trip time value from the header field and use it to calculate the maximum transmission rate based on the queue information and the value of the round trip time carried in the current packet. The estimated value is then updated and registered in the packet's header in order to send it to the next intermediate node or final destination. This transmission rate computation is performed at the network layer for every current flow at a particular time. The intermediate node updates the advertised window size header field of each data packet passing through it after converting the transmission rate into a window size value and before sending the data packet to the transport layer.
- In order to prevent all the intermediate nodes along a path from congestion, the current advertised window size is compared with the previous advertised window size at every node or next hop, keeping the smallest of all values in the header of the TCP/IP packet that will

finally reach the destination. As a result, the packet carries, within the advertised window size header field, the minimum allowed transmission rate of the intermediate nodes along the path.

3.2.4. Supporting Mechanisms in Different Layers

In this section the tasks and functionalities done at the transport and network layer are explained and their implementation is described. They help to illustrate the general protocol operation under which our PI-rate controller is implemented.

The network layer mainly deals with the scheduling and queuing of packets, the forwarding functions and hop-by-hop transfer as well as with the routing of segments towards the destination node. Depending on which role a node plays, this layer behaves differently. On the source node, the main task is to receive segments from the application layer, split them into packets and transfer them to the next hop with the help of the selected routing policy. On the destination node, the packets need to be re-assembled into segments after they have passed through the network layer in order to send complete segments to the transport layer. On intermediate nodes, the network layer receives incoming packets and stores them. When a segment is complete, the layer forwards the packets to the next hop in direction of the destination.

The network layer helps providing to the transport layer the following services: Buffer segments, forward buffered segments to the next hop in direction of the destination, notify about control packets related to the connection state, notify when a complete segment was sent, and notify when a complete segment has arrived.

Additionally, the following services from the lower layers are required: (1) Delivery of datagrams to any intermediate node on a best-effort basis, and (2) Retrieval of IP address of the next hop to any destination node to forward data packets by means of a routing policy or protocol.

The main forwarding operations interacting with other layers include: receiving segments from the transport layer, queuing and scheduling the packet service, updating the advertised window size field value, and forwarding the segment to a next hop until it reaches the destination. Upon receiving a segment from the transport layer (Algorithm of Figure 3.4), the

forwarding functions at the network layer split it into packets and buffer them. This means they enter into a queuing and scheduling process where the packet is inserted into a queue and the time where it will be de-queued is scheduled. After a particular service and processing time, the segment will be forwarded to the next hop after extracting the next hop IP address and detecting the specific routing protocol the network is working with. It will repeat the same task as long as there are next segments in the queue. Then it tries to forward the next segment from the queue to the next hop.

To forward a segment to the next hop, the network layer routing functions will look up the routing table for the next hop address. Upon obtaining a valid next hop, packets are transmitted to this intermediate node until either all packets are sent or a link failure is detected. For this case, the respective Ad-hoc routing protocol (DSR in this case) will generate an error message. After receiving a failure message, a new route is requested. After the complete transmission of a segment is finished, a retransmission timer is set and a next segment is sent.

The above process is repeated at each intermediate node until eventually reaches the destination leading to the completion of the corresponding segment. The network layer and its forwarding functions verify at each intermediate node if it is the destination of the connection by comparing the "Destination_Id" of the header of the packet with its address. If this is the case, it passes the segment to the application layer. If the layer determines that it is an intermediate node, the host continues forwarding the segment.

The implementation of the PI rate controller required modifications mainly to the network layer and the transport layer. Firstly, we deal with different input and output files. These files include the general simulation parameters configuration of each of the network communication layers. There is a file with the information about the nodes interfaces and the location of the nodes in the network. For the case of file-based mobility, there is a file where the mobility pattern for each node is defined. Finally, there is a file where the applications configuration used in the simulation are defined.

For the case of the adjacent cross layer communication, Qualnet makes use of messages and Application Programming Interfaces (API) functions. These functions allocate messages and provide them with standard events, layers and protocol information; allocate space for the

packet within the message; add a header to the packet (usually called by each layer in the protocol stack); remove a header from the packet as it passes through a different communication network layer.; send messages as special events to the specified layer and protocol; and free the message once it has reached its final destination.

In order to include the PI rate congestion control in Qualnet, we added a module within the IP protocol at network layer. By adding this module whenever a packet is received from transport layer it enters this module and update the advertised window size. To achieve this, three functions are needed: the initialization function, the message processing function and the finalization function. Something similar happens with the transport layer and the TCP protocol. In order to modify TCP's congestion control algorithm the TCP input and output functions were modified. Additional changes where done to the TCP implementation in Qualnet, which includes modifications to the transport header (tcp_hdr.h and tcpip.h), as well as to the timers and subroutines (tcp_subr.cpp and tcp_timer.cpp) of TCP. The transport layer modifications will guarantee a reliable end-to-end connection between source and destination nodes.

3.3. Discussion

This scheme used by the adaptive PI rate controller offers a simple way to provide flow congestion control through a mechanism that controls the transmission rate of source nodes. The self tune capabilities of the PI rate control algorithm offer the ability to react to the fluctuations of the network. By controlling the transmission rate this scheme offers a way to improve the throughput performance of TCP.

It incurs an additional overhead at the nodes as it needs to compute the transmission rate for the respective flows and update it whenever it is needed. However, it is not targeted for large scale networks as the Internet, but rather for Ad-hoc networks which are often small scale networks for a group of mobile users.

Unlike Internet, there is no "core" router in Ad-hoc networks. Therefore, flows are more evenly routed throughout the network. Additionally, in Ad-hoc networks a node is more likely to talk to another node physically close to itself. As a result, the number of concurrent flows going through a node in Ad-hoc networks is likely to be relatively small.

Even with the additional processing overhead, this scheme is feasible for Ad-hoc networks as it is simple to implement without requiring complex modifications. It requires simple cross layer communication as it is between adjacent layers. It does not require huge modifications to many communication layers and its protocols, which ease its deployability and interoperability with other networks. Because of its simplicity, the PI rate control algorithm offers the possibility to be extended by means of the use of feedback information from other layers. However, in this thesis the main objective was to evaluate the performance of the PI rate control algorithm by itself and provide comments for future extensions.

3.4. Network Configuration

SANETs and MANETs are two types of Ad-hoc networks tested in this thesis. Their scenarios will be described in the next section while the particular configuration for each scenario will be described at the simulation settings section. Before we do this, we provide more functionalities in various communication layers.

At the physical layer, every node is equipped with an omni-directional antenna. The 802.11 WLAN interfaces of Qualnet as well as Qualnet's statistical propagation model are used. This propagation model allows for the evaluation of the mobile radio channel without specific terrain data considerations, and where the channel parameters are modeled as stochastic variables. Based on the transmission power and other default physical parameters (Table 4.1), the radio propagation range for each node was set to about 270 meters. The channel bandwidth capacity used has a bit-rate of 11 Mbps.

At the MAC layer, the Distributed Coordination Function (DCF) of the IEEE 802.11 standard was used. The 802.11 DCF uses Request-to-send (RTS) and Clear-to-send (CTS) control packets for uni-cast data transmission to neighboring nodes for packets larger than a given threshold (set to 512 bytes). Broadcast data packets and the RTS/CTS control packets are sent using an un-slotted CSMA technique with collision avoidance (CSMA/CA). For the address resolution, we use the basic ARP protocol implemented on Qualnet.

At the network layer, all nodes and experiments run the Dynamic Source Routing protocol (DSR) [IETF04] implementation for Qualnet. The default buffer size at each node used by DSR

is set to 50 packets. AODV is only used for the mobile scenarios in order to make the performance comparison under different Ad-hoc routing protocols. AODV use a default buffer size of 100 packets used to perform route requests.

Finally, at the application layer, each source node uses a generic/FTP application for each flow to allow data to be continuously transmitted at the highest rate possible. The File Transfer Protocol (FTP) application is used to model continuous sending of packets of a particular size from a particular source node to a particular destination node during a certain period of time.

3.4.1. Scenarios

The scenarios can be generally classified as static and mobile. These two major types of scenarios are detailed in the following sections.

3.4.1.1. Static Scenarios

Static scenarios refer to the environment where the network nodes are not moving at all. These scenarios consist of different topologies for static Ad-hoc networks (SANETs). We use these scenarios to demonstrate particular characteristics of the congestion control algorithm in an observable environment as well as to verify its correct operation. The legend on the right hand side of the scenarios topology gives the details of the flow configurations:

a) the type of application used: FTP/GEN means GENeric File Transfer Protocol is used.
b) the source and destination nodes Id, .e.g. in the first flow of Scenario-1 above, the source node is Node-1 and the destination node is Node-3
c) the packet size in bytes, e.g. 512 bytes,
d) the starting and ending time of the application (in seconds), e.g. the first flow in Scenario-1 has a duration of (250-1) =249 seconds

Three static topologies are used to test and compare our algorithm. Various number of nodes and number of flows are used to study the impact and performance under different number of hops and simultaneous flows.

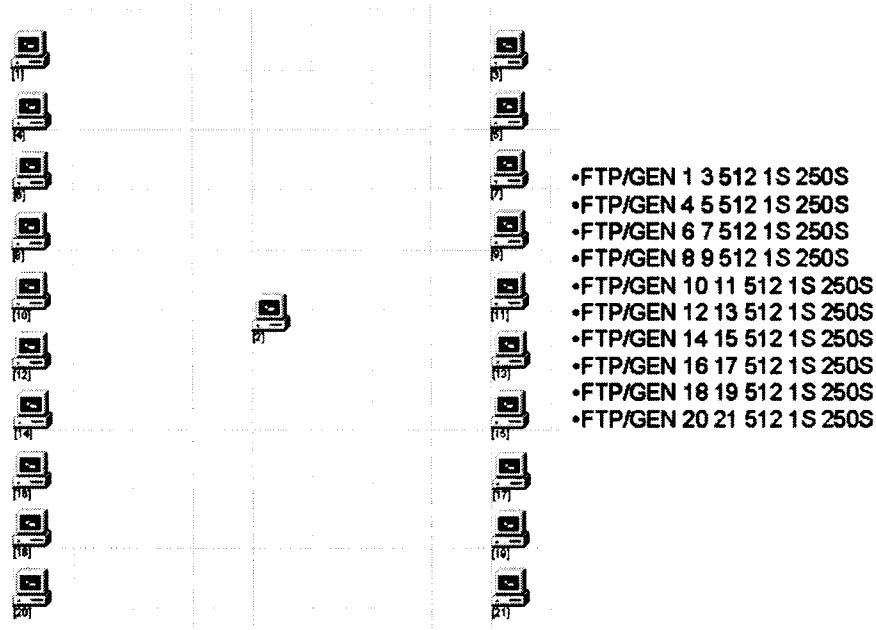

Figure 3.5: Bottleneck Two-Hop Connection

Scenario 1: Bottleneck in a Two-Hop connection

The first scenario is the simplest one. Figure 3.5 shows the topology as well as the parameters set for each FTP flow (Section 4.1). TCP traffic is used by means of setting 10 FTP/Generic application flows continuously sending packets of 512 bytes along all the simulation time.

This scenario contains a bottleneck without background traffic. In this scenario 10 nodes are set within a distance of 250 m from an intermediate node, node 2, which is the only one to be able to relay information from these 10 source nodes to the respective 10 destination nodes. The distance between the source node and a destination node in this scenario is set so they cannot reach each other directly but only through the intermediate node 2. This will force each pair of source-destination nodes to communicate through the intermediate node 2 and therefore, to test the congestion control performance on it.

This scenario will allow us to verify the main performance of the PI rate congestion control algorithm by generating congestion on the intermediate node 2 and without any kind of interference or external factor. Afterwards, we will be able to introduce other factors such as number of hops and number of simultaneous nodes, as well as mobility.

Scenario 2 - Four-hop connection (5 nodes, 1 flow) in a chain topology.

Figure 3.6 shows a chain topology where we connect source node 1 and destination node 5 via three intermediate nodes (2, 3, and 4) that are aligned. The distance of every hop is 250 m.

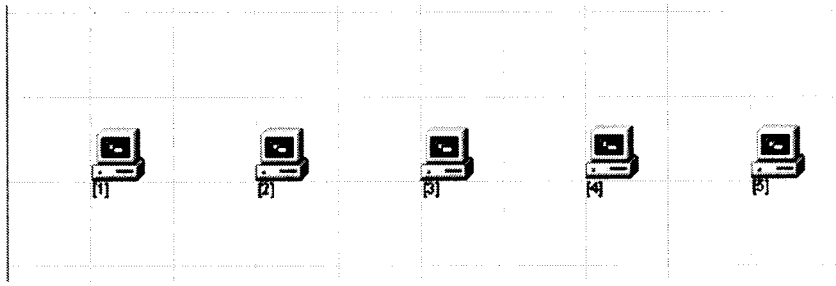

Figure 3.6: Four-hop Connection (5 nodes, 1 flow) in a Chain Topology.

We use this scenario to see the effect on bandwidth and throughput performance at a different distance from the destination node as well as the effect of the number of intermediate nodes between them.

Scenario 3 - Nine-hop connection (10 nodes, 1 flow) in a chain topology.

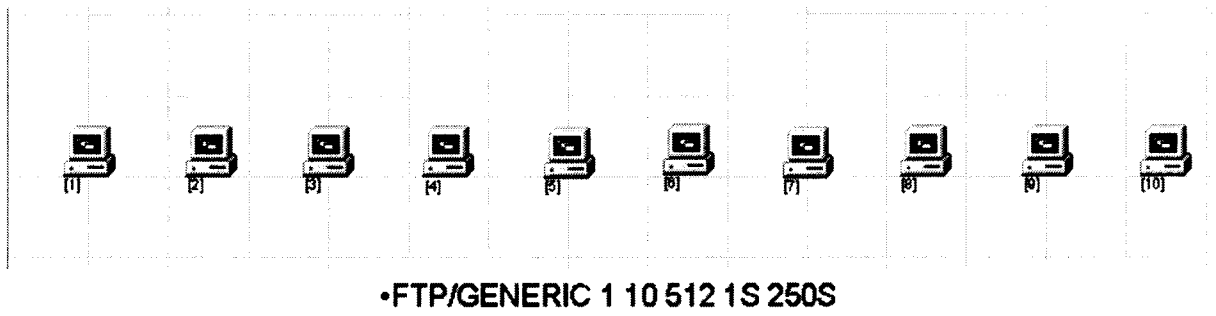

Figure 3.7: Nine-hop Connection (10nodes, 1 flow) in a Chain Topology.

The scenario in Figure 3.7 is similar to that in Figure 3.6, except for the fact that the number of intermediate nodes and therefore hops is increased. Source node 1 sends data to the destination node 10 via the aligned intermediate nodes (2, 3, 4, 5, 6, 7, 8, and 9). The distance between every two consecutive nodes is 250 m.

We use this scenario to determine the effect of increasing the number of hops on bandwidth and throughput performance.

Scenario 4 - Four-hop connection (5 nodes, 4 flows) in a chain topology.

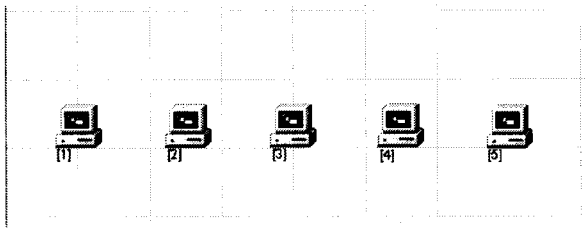

Figure 3.8: Four-hop Connection (5 nodes, 4 flows) in a Chain Topology.

Figure 3.8 shows scenario 4. This scenario is the same as that in Figure 3.6, but instead of having one flow from the source node 1 to the destination node 5, there are 4 simultaneous FTP flows running during the time of the simulation with slightly delayed starting times.

The purpose of this scenario is to test the impact of the number of simultaneous flows in a connection as well as to see the performance of the PI rate control algorithm for achieving fairness.

Scenario 5 - Nine-hop connection (10 nodes, 4 flows) in a chain topology.

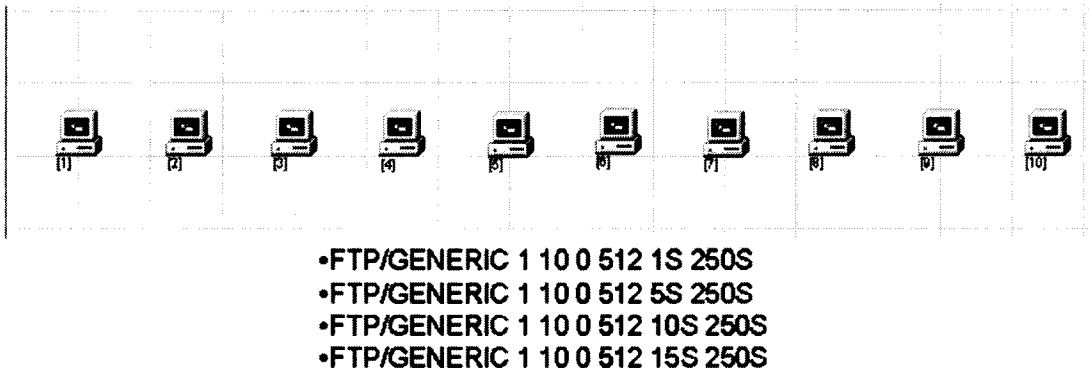

Figure 3.9: Nine-hop Connection (10 nodes, 4 flows) in a Chain Topology.

Scenario 5 in Figure 3.9 is similar to that in Figure 3.7, except for the fact that the number of simultaneous flows is increased. Source node 1 sends data to the destination node 10 via the aligned intermediate nodes (2, 3, 4, 5, 6, 7, 8, and 9). There are 4 simultaneous flows running along all the simulation and with slightly delayed starting times. We use this scenario to test the effect of multiple hops and multiple simultaneous nodes on the congestion control algorithm performance.

Scenario 6 - 50 nodes randomly placed and uniformly distributed

Figure 3.10 shows the most complex static scenario tested on this thesis. It is intended to test the PI rate control behavior in an Ad-hoc network of 50 nodes which are randomly placed and uniformly distributed in an area of 700 x 700 meters.

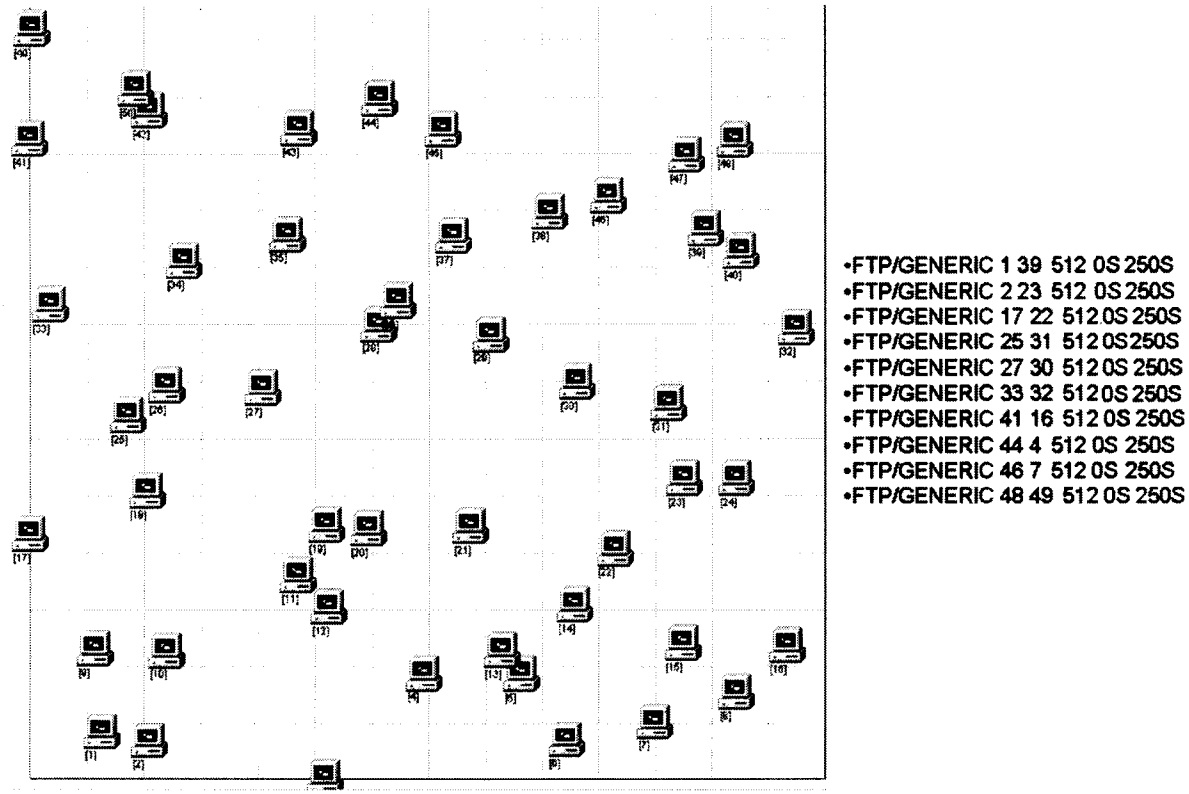

Figure 3.10: 50 Nodes Randomly Placed and Uniformly Distributed

The distance between any two nodes is variable and is not restricted to the maximum transmission range as in the chain topologies where interference was reduced. In other words, in this scenario it is possible to have mutual interference between nodes.

3.4.1.2. Mobile Scenarios

The aim of these scenarios is to evaluate the performance of the PI rate control algorithm under various Ad-hoc scenarios with different node speeds. For example, Figure 3.11 shows the starting topology with which we intend to test a simple mobile scenario in a chain topology by adding mobility to a particular node. The explanation of the legend can be found in the Static Scenarios section.

Scenario 7 – Source node moving back and forward towards a destination in a four-hop connection and chain topology (5 nodes, 1 flow, 3 m/s)

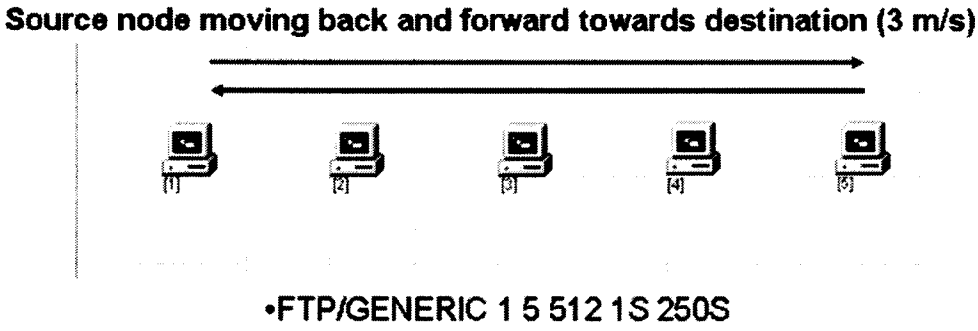

Figure 3.11: Source Node Moving Back and Forward Towards a Destination in a Four-Hop Connection and Chain Topology (5 nodes, 1 flow, 3m/s).

Figure 3.11 shows a chain topology where the source node 1 and destination node 5 are connected via three intermediate nodes (2, 3, and 4) that are aligned. The source node 1 is moving back and forward towards the destination at a speed of 3 m/s. We use this scenario to add node mobility and see its effect on bandwidth and throughput performance when the source gets nearer or farther from the destination.

Figure 3.12 shows a more complex topology used for scenarios 8, 9, 10, 11, and 12 to be described below. In these scenarios 10 random source nodes and 10 random destination nodes

are selected in order to establish 10 FTP connections within the network. We tried one mobile topology with 5 different moving speeds.

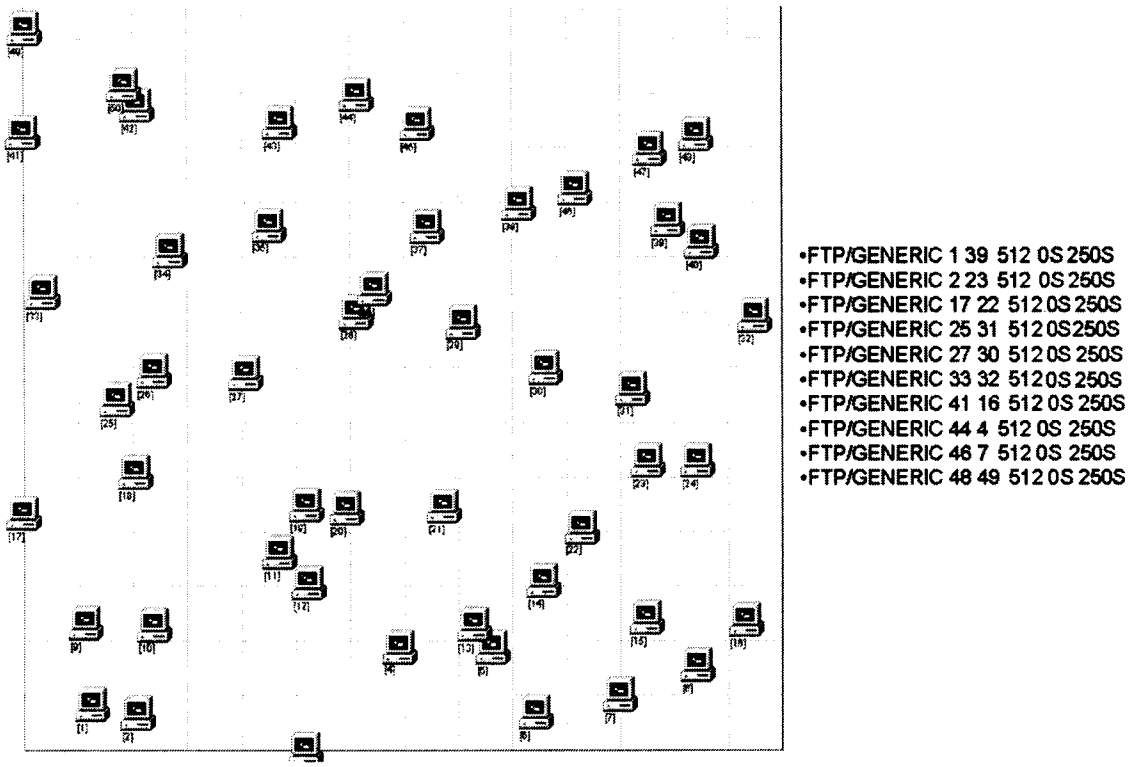

Figure 3.12: 50 Nodes Randomly Placed and Uniformly Distributed Using 1, 5, 10, 15 and 20 m/s Speeds, respectively.

Scenario 8 - 50 nodes randomly placed and uniformly distributed using 1 m/s speed.

Every node in the network is randomly placed and uniformly distributed in an area of 700 x 700 meters. The mobility model used is the Random-waypoint model [CaBo02] available from Qualnet. Using this mobility model every node randomly selects a destination from the physical terrain as well as speeds for each node. Each node starts moving towards a randomly chosen destination with a randomly chosen speed within the range defined by the following parameters: MOBILITY-WP-MIN-SPEED and MOBILITY-WP-MAX-SPEED, which respectively defines a minimum and a maximum node speed.

After the nodes reach their selected destinations, the node stays there for a time period specified by the parameter MOBILITY-WP-PAUSE. The process is then repeated, i.e. another random destination and speed is chosen for the next operation. A fourth parameter (MOBILITY-POSITION-GRANULARITY) is set in order to determine the resolution at which node positions will be updated. A smaller value means the more often a node position is updated by a smaller increment.

For this scenario, we have the following setting. The MOBILITY-WP-MAX-SPEED is set to 1 m/s while the MOBILITY-WP-MIN-SPEED is set to 0.5 m/s. The MOBILITY-POSITION-GRANULARITY value is set to 0.5 meters.

Scenario 9 - 50 nodes randomly placed and uniformly distributed using 5 m/s speed.

Same as Scenario 8 except that the nodes are moving randomly at a speed also set accordingly to the mobility way point minimum and maximum speeds. The MOBILITY-WP-MAX-SPEED is set to 5 m/s while the MOBILITY-WP-MIN-SPEED is set to 1 m/s.

Scenario 10 - 50 nodes randomly placed and uniformly distributed using 10 m/s speed.

Same as Scenario 8 except that the nodes are moving randomly at a speed also set accordingly to the mobility way point minimum and maximum speeds. The MOBILITY-WP-MAX-SPEED is set to 10 m/s while the MOBILITY-WP-MIN-SPEED is set to 5 m/s.

Scenario 11 - 50 nodes randomly placed and uniformly distributed using 15 m/s speed.

Same as Scenario 8 except that the nodes are moving randomly at a speed also set accordingly to the mobility way point minimum and maximum speeds. The MOBILITY-WP-MAX-SPEED is set to 15 m/s while the MOBILITY-WP-MIN-SPEED is set to 10 m/s.

Scenario 12 - 50 nodes randomly placed and uniformly distributed using 20 m/s speed.

Same as Scenario 8 except that the nodes move randomly according to the mobility way point minimum and maximum speeds which are respectively set to MOBILITY-WP-MAX-SPEED

equal to 20 m/s and MOBILITY-WP-MIN-SPEED equal to 15 m/s.

3.5. Assumptions

The following assumptions about the target networks are used throughout this thesis:

1. Each source and destination node pair has reverse traffic; however, the amount of traffic can be different in each direction.
2. The destination node is not able to send packets actively, unless it has received a data packet from the source (then, an ACK is sent back to the source). This will simplify the simulation and control the effect of any other transport layer factor different from the congestion control algorithm used in the performance evaluation,
3. The receive window size of the destination node is limited according to an upper bound provided by the size of the receiver buffer. In such way the source node is not able to send more data than the available space of the receive buffer at the receiver.
4. The advertised window size is upper-bounded by the sender buffer size and receiver buffer size. The TCP sender can send neither more data than what is stored in the sender buffer nor more data than the available space of the receive buffer at the receiver. This assumption would allow us to evaluate the performance of the congestion control algorithm under conditions that closely represent a real network implementation.
5. The performance of each intermediate node contributes to the overall performance of a particular session. Therefore, the buffer size of each intermediate node is made finite to test scenarios that closely represent real networks where the buffer size of the nodes is limited.
6. No link layer approaches are considered to improve the TCP performance over wireless Ad-hoc network. This would help to isolate the performance of the congestion control algorithm alone. Any other layers' feedback information or cross layer communication that could improve the performance of the congestion control algorithm could be proposed later to form part of a complete congestion control framework.

7. Every node has a CPU with an unlimited processing power and unlimited energy resources. This would allow us to evaluate the performance under congestion without having to consider external factors such as the impact of shortage and limited resources on node performance.
8. All nodes in the network are willing to collaborate so that malicious behavior of nodes is not considered.
9. This congestion control algorithm does not consider any mechanism to detect delays due to link failures as it could be provided depending on the routing protocol used. In fact, this delay can be indirectly considered in the total round trip time as this is the total time the packet takes since the source node sends the packet until it receives back a reception acknowledgement. Consequently, the round trip time cannot distinguish its components due to propagation delay, queuing delay, processing delays or delays due to link failures.

3.6. Performance Measures

The main performance measure is throughput. Both average throughput per connection and aggregated throughput will be measured in the simulations. They will also be used to compare fairness among different scenarios with multiple simultaneous connections. The throughput measurements are based on the data received from the transport layer at the destination nodes. The following are their definitions as well as the ones of some other parameters:

1) Average Throughput: is defined to be the total number of bits of all packets sent successfully so far per session time. Session time is the time duration between the time instant when the first byte is sent from the source and the time instant the last packet is received at the destination. We measured this in our simulation from the beginning of the simulation to the current simulation time. This allows us to compare at any point in time how much data was successfully transmitted.
2) Round Trip Time (RTT): This is defined to be the total time that the first bit of a packet is sent out from the source node until the last bit of an ACK is received back at the same source node.

3) Fairness: We used Jain's fairness index [JaCh84] defined as:

$$F(x) = \left[\sum_{i=1}^{n} x_i\right]^2 \Big/ n\sum_{i=1}^{n} x_i^2$$

where n is the number of flows and x_i denotes the throughput achieved by flow i. This particular metric is mainly used for the Scenario 1 to measure the equality allocating bandwidth to users in a shared channel of a bottleneck node, and therefore the capability of our PI rate controller under congestion. Similarly, this index is used for Scenarios 4 and 5 where multiple and simultaneous flows pass through a same route path.

3.7. Concluding Remarks

We have provided in this chapter an overview of the main operation of the network along with the general parameters and configuration. The PI rate control algorithm mechanism as well as it implementation has been described in detail. A brief description of the network simulator used as well as its models and main structure were described along with the data types used and the interactions among different communication layers and functions. The test network scenarios to be used for evaluation have been described as well as the performance measures.

Of particular controversial is the fairness index which, although it can consider equity in the allocation of resources to contending nodes, is unable to consider the different traffic and mobility patterns in more dynamic topologies where the distribution among neighboring nodes must also need to be considered. Fairness evaluation in multi-channel wireless ad-hoc networks is still an open issue which so far has focused on the channel assignment problem and the multiple access control protocol for multi-channel operation. A method [LeMi06] has been introduced to evaluate the effectiveness of channel assignment and distribution among neighboring nodes to measure unfairness depending on the network topology and traffic pattern of the network. However, we did not implement it as it was not tested the effect to improve the channel utilization and because it is also out of the scope of this work.

The evaluation, simulation and performance results of the PI rate congestion control algorithm in different scenarios given in this chapter will be presented in Chapter 4 and 5.

Chapter 4
Static Scenarios

We shall evaluate the performance of our PI-rate control algorithm in various static scenarios. Not only will this verify the correctness of our algorithm implementation, but it will also establish a benchmark for comparison with the mobile scenarios in the next chapter. The main performance measures are average throughput, round trip time and fairness already defined in Chapter 3. A description of the chosen simulation parameter settings is given. Additionally, some properties of SANETs are observed.

4.1. Qualnet Simulation Environment

We have both implemented the PI rate control algorithm and used TCP New Reno [FlHe04] in Qualnet so that a comparison to our scenarios with and without the PI congestion control algorithm can be made. The Qualnet model, operation and implementation, especially those of the PI rate controller, have been detailed in Chapter 3.

A generic FTP application with a packet size of 512 bytes is used. This generic FTP is a more configurable model of the File Transfer Protocol where the size of the packets sent is specified by the user instead of determined by network traces with distributions randomly taken from the tcplib library. The FTP/Generic traffic in the application configuration file can specify parameters according to the format: FTP/Generic <Source> <destination> <packets to send> <packet size><start time><end time><precedence value>. Some examples are already described in Section 3.4.1.1.

All network systems are simulated 250 seconds, which is determined to be adequate to establish reasonable steady-state results. The result values are the average of 3 different simulation runs using 3 different seeds. This enabled for each particular scenario to compute a 95% confidence. The confidence varies more for the scenario of nodes randomly located and uniformly distributed, as expected. However, they are generally small.

The computer used for the simulation runs under the Operating System Windows XP

Table 4.1: Simulation Configuration Settings

********** General Configuration **********		
Parameter	**Value**	**Observations**
SIMULATION-TIME	250s	Total duration of the simulation. About 50000 samples per seed.
SEED	1, 3, 5	The random number seed is used to initialize part of the seed of various randomly generated numbers in the simulation. Use different seeds to see the consistency of the results of the simulation
TERRAIN-DIMENSIONS	• (2700m, 500m) chain topology • (700m, 700m) randomly placed and uniformly distributed topology	The size (x, y) of the physical terrain in which the nodes are being simulated. x and y are dimensions in meters.
DUMMY-NUMBER-OF-NODES	• 5 nodes (scenarios 2 & 4) • 10 nodes (scenario 3 & 5) • 21 nodes (scenario 1) • 50 nodes (scenarios 6, 8 to 12)	The number of nodes being simulated depending on the scenario.
NODE-PLACEMENT	UNIFORM	The node placement strategy.
MOBILITY	NONE (static scenarios)	The mobility model used.

********** Wireless Settings **********		
Parameter	**Value**	**Observations**
PROPAGATION-CHANNEL-FREQUENCY	2.4 GHz	Propagation frequency of each simulated channel.
PROPAGATION-MODEL	STATISTICAL	Propagation model used on each simulated radio channel where no specific terrain data is considered, and channel parameters are modeled as stochastic variables.
PROPAGATION-LIMIT	-111.0 dBm	Signals with powers below PROPAGATION-LIMIT (before the antenna gain at the receiver) are not delivered.
PROPAGATION-PATHLOSS-MODEL	TWO-RAY	Two-ray model uses free space path loss for near sight, and plane earth path loss for far sight. This model considers ground reflection on flat earth [Rapa95]. This model is suited for Ad-hoc networks due to the environmental conditions of low transmit power and low antenna height.

********** Radio/Physical Layer **********		
Parameter	Value	Observations
PHY-MODEL*	PHY802.11	Physical layer model to transmit and receive packets.
PHY802.11-DATA-RATE*	11 Mbps	Maximum supported data rate in.
PHY802.11b-TX-POWER-11MBPS	15.0 dBm	Transmission power in
PHY802.11b-RX-SENSITIVITY-11MBPS*	-83.0 dBm	Receiver Sensitivity of the nodes.
PHY802.11-ESTIMATED-ANTENNA-GAIN	0.0 dBi	Estimated antenna gain.
ANTENNA-MODEL*	OMNIDIRECTIONAL	The antenna model which in this case returns 0 dBi gain in all directions.
*Note: These values were set according to wireless cards specifications determined by manufacturers.		

********** MAC Protocol **********		
Parameter	Value	Observations
MAC-PROTOCOL	MAC802.11	Wireless LAN media access protocol.
PROMISCUOUS-MODE	YES	YES if nodes want to overhear packets destined to the neighboring node.

********** Network Protocols **********		
Parameter	Value	Observations
NETWORK-PROTOCOL	IP	
IP-QUEUE-NUM-PRIORITIES	1 priority	We consider only one priority queue for each network interface. The packets are queued for transmission at the output buffer by a set of functions that IP uses to en-queue and de-queue packets.
ROUTING-PROTOCOL	-DSR -AODV	DSR is used for most of the simulations. AODV is only used for the comparison of the PI rate controller under different Ad-hoc routing protocols.
DSR-BUFFER-MAX-PACKET	50 packets	Specifies the maximum size of message buffer.

********** Transport Layer **********		
Parameter	Value	Observations
TCP-USE-RFC1323	YES	This option allows sending timestamps in the TCP options header.
TCP	• New RENO • PI_rate	Transport layer protocol variant used.
TCP-MSS	512 bytes	Maximum segment size.

Professional, Version 2002 and Service Pack 2.0. It uses an Intel SIPP Pentium 4 CPU processor working at a 3.4 GHz speed and with a 1.49 GB of RAM memory size. A typical simulation run takes about 180 seconds.

4.2. Simulation Configuration Settings

Table 4.1 presents the parameters and their values in Qualnet that were used for each of the different communication layers. The packet size is set to 512 bytes. In the first network scenario, it is verified that the PI congestion control framework transfers data and control the transmission rate correctly. Then by testing the rest of the scenarios it is observed how it performs in order to cope with congestion control, cross traffic, number of hops and simultaneous nodes.

4.3. Static Scenarios

All static scenarios have been described in Chapter 3 along with their rationale. They are used here for the analysis in SANETs.

4.3.1. Scenario 1: Two-Hop Connection with a Bottleneck

This scenario has been depicted in Figure 3.2 where 10 transmitting nodes on the left must rely on the center node 250 m away to the 10 destination nodes on the right, which are another 250 meters away. Congestion is created at the bottleneck node, where the behavior of the PI rate control algorithm will be evaluated. The source of each connection continuously and simultaneously sends packets of 512 bytes to the destination.

We will plot and present a comparison result between TCP and the PI integral congestion controller by graphing different parameters (throughput, instantaneous and average queue size, window size and RTT). More details are provided here than other scenarios in order for us to verify the main performance of the adaptive PI rate congestion control algorithm.

Throughput

Figure 4.1a shows the transmission rate of one of the controlled FTP source nodes. The controlled FTP source node starts to increase its transmission rate until it reaches and maintains

a steady state rate around 82.68 Kbps. However, though the same source node using TCP New Reno reaches also the steady state, the rate that it achieves is just 71.59 Kbps as depicted in Figure 4.1b.

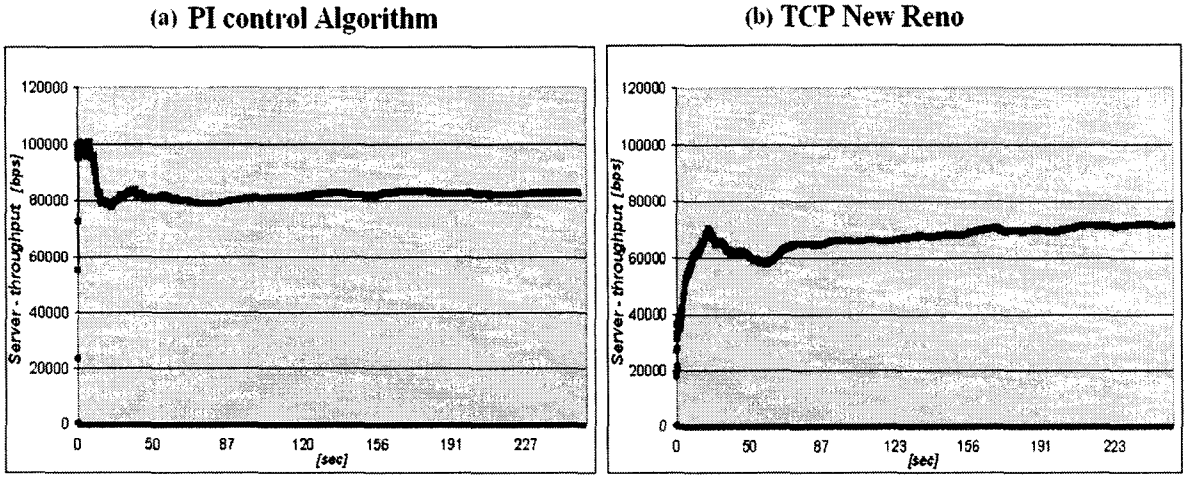

Figure 4.1: Throughput Performance of the Two-Hop Bottleneck Scenario

The plot shows that in comparison with TCP New Reno, the PI Rate controller is able to get more average data throughput. We can also observe that for the PI rate controller there is an initial peak in throughput at the beginning of the simulation. At that point there is still no congestion; therefore, the source node tries to send the maximum number of packets which is limited by the buffer size. It is until the source node gets the receiver's feedback from each path that this value is decreased to reach the steady state. For TCP, the throughput shows a slower behavior, reaching the steady state slower. This is basically due to its additive increase and multiplicative decrease congestion process.

RTT

It is also noticeable, as Figure 4.2a shows, that the PI rate controller shows a better performance in regard to the average round trip time delay. The RTT average value for the PI rate controller is almost half shorter (275 *ms*) than that of TCP New Reno (521 *ms*). This is due to the fact that by controlling the number of packets that the source nodes are able to transmit,

congestion is avoided as it reduces the processing time that packets could experience in the queue of the intermediate node, node 2.

(a) PI control Algorithm (b) TCP New Reno

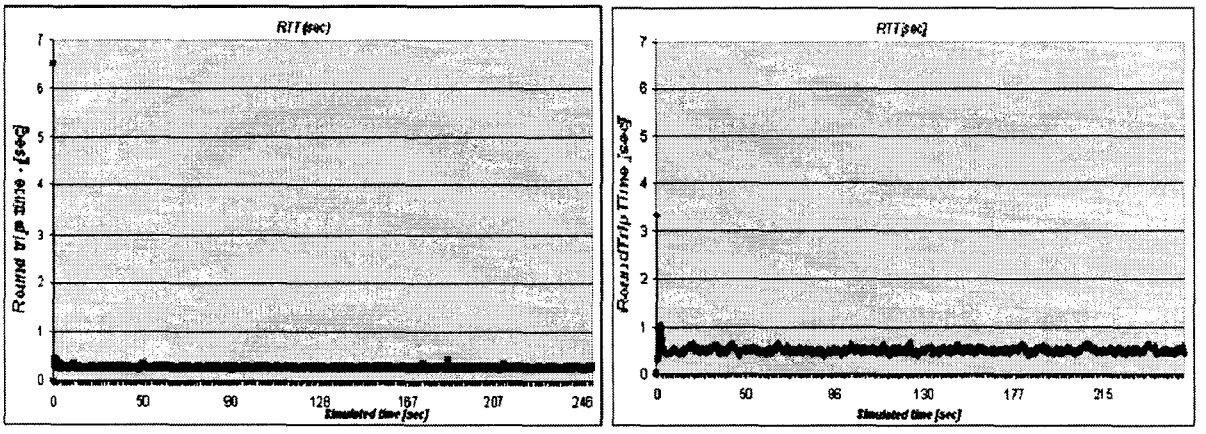

Figure 4.2: RTT Performance of the Two-Hop Bottleneck Scenario

Average Queue Size

(a) PI control Algorithm (b) TCP New Reno

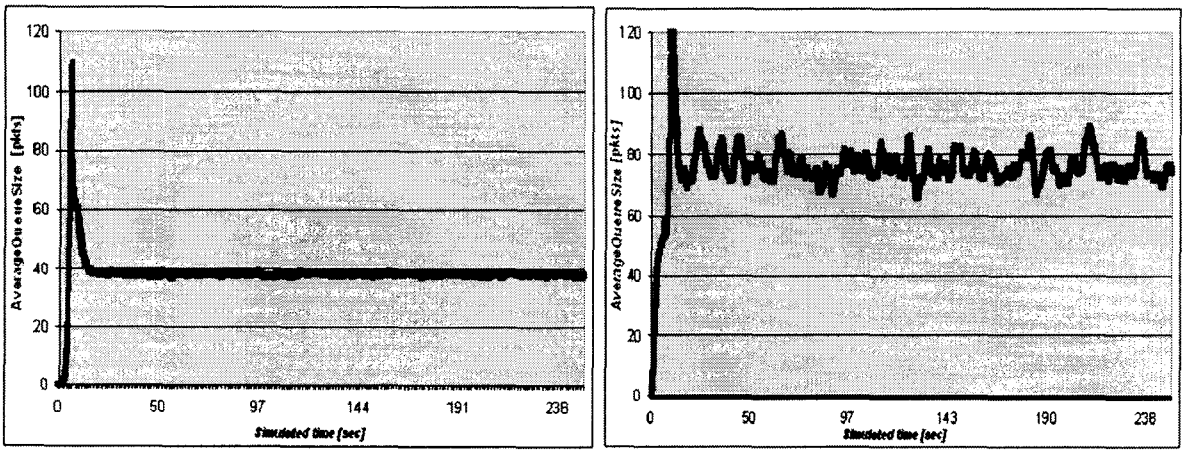

Figure 4.3: Average Queue Size Performance of the Two-Hop Bottleneck Scenario

Figure 4.3 shows the average queue size of the intermediate node 2 for both TCP New Reno and the adaptive PI rate controller. It can be seen that with TCP New Reno, the queue size in the intermediate node is bigger than the PI rate controller (80 vs. 40 packets) and fluctuates with a larger amplitude.

Instantaneous Queue Size

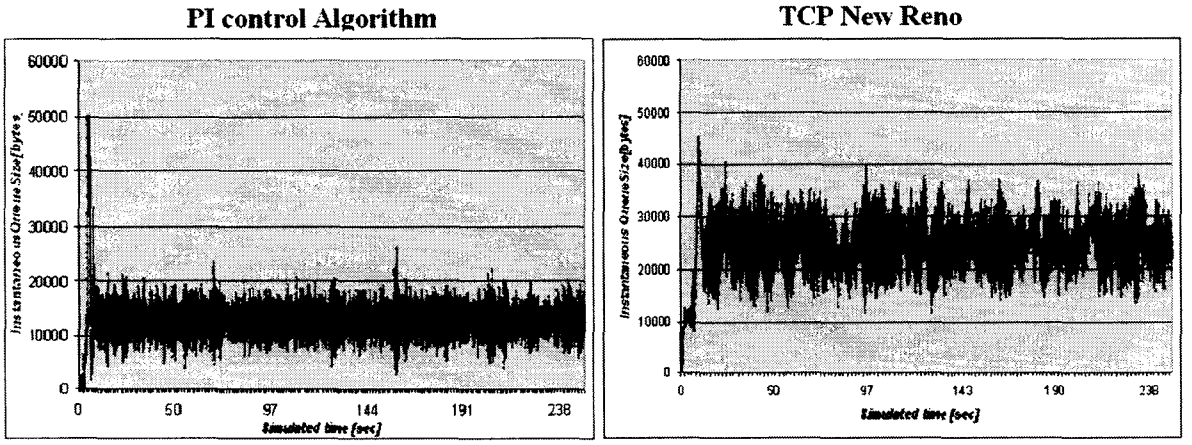

Figure 4.4: Instantaneous Queue Size Performance of the Two-Hop Bottleneck Scenario

Figure 4.4 shows the instantaneous queue size of the intermediate node 2 for TCP New Reno and the adaptive PI rate controller. It can be seen that for TCP New Reno the instantaneous queue size in the intermediate node-2 fluctuates with a larger amplitude. For the PI rate controller the instantaneous queue size increases and approaches to its target buffer occupancy (13000 bytes).

Window Size

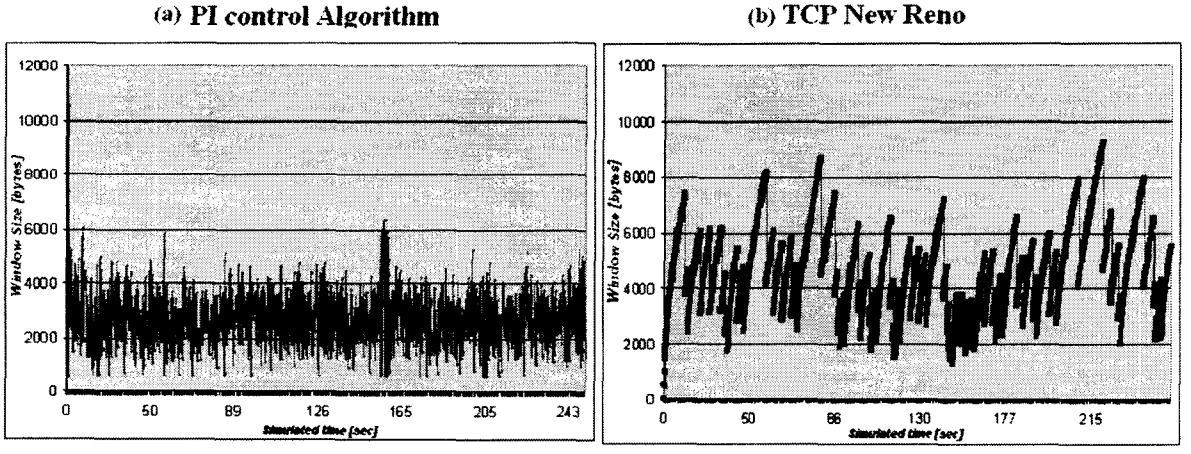

Figure 4.5: Window Size Performance of the Two-Hop Bottleneck Scenario

Figure 4.5 shows the instantaneous window size of one of the source nodes for TCP New Reno and the adaptive PI rate controller. It can be seen that for TCP New Reno the window size in

the intermediate node fluctuates with larger amplitude. Its average window size is around 4696.27 bytes while for the PI rate controller the average window size value approaches 2701.9 bytes. The PI rate controller shows a smaller fluctuation in the amplitude of the window size than TCP New Reno. This helps to avoid bursts and peaks of traffic that could cause congestion in intermediate nodes along a path. Therefore, it also helps on reducing the delays and packet drops as well as the contention on the wireless channel.

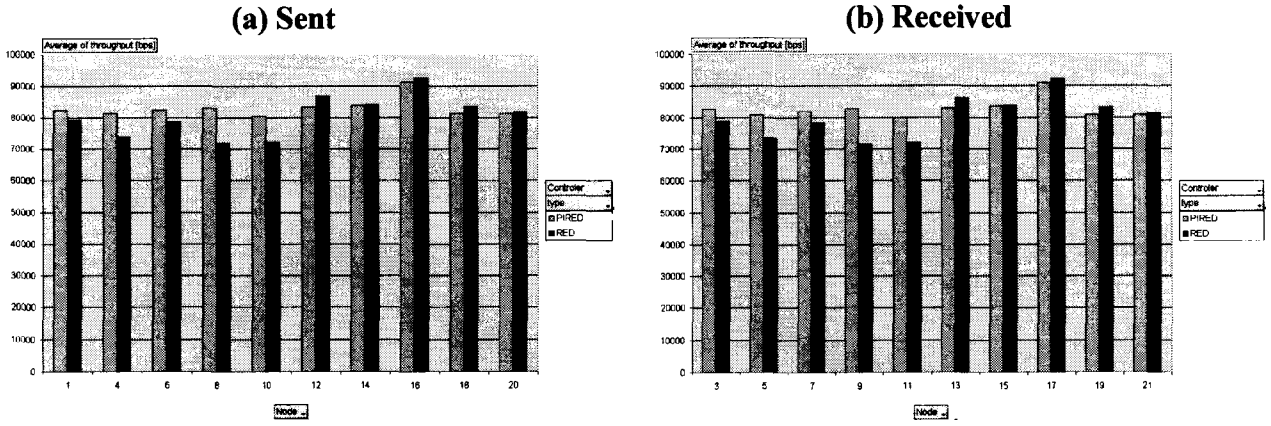

Figure 4.6: Average Throughput Performance per Node of the Two-Hop Bottleneck Scenario

Figure 4.6 shows the average throughput sent and received by each node on the Two-hop Bottleneck Scenario. The sent-throughput refers to the total bytes sent out by the source while the received-throughput is the actual number of bytes effectively received by the destination (the performance measure we are interested in). They are not always the same, as congestion in the intermediate nodes and the wireless contention of all the nodes at a particular time might cause a delay or drop of some packets. The adaptive PI rate controller in general received more average aggregated throughput (around 6% in this static scenario) than TCP New Reno because it is actually effectively controlling the source rate transmission while avoiding congestion on the intermediate node. These results confirm the performance already described for a particular node on figure 4.1, but for the rest of the nodes sharing and contending the wireless channel.

Table 4.2 shows the performance results of the Two-hop Bottleneck Scenario. Note that the

adaptive PI rate controller has achieved a better average queue size performance than TCP New Reno (its average queue size of 12927 bytes is roughly half of that (24601 bytes) of TCP New Reno). The PI rate controller also shows a better performance in regard to the average round trip time (275 ms which is about half that (521 ms) of TCP New Reno). Moreover the average aggregated throughput for the PI rate controller is slightly better (5.8%) than that of TCP.

Table 4.2: Simulation Results of the Two-Hop Bottleneck Scenario

Average Throughput [bps]	3	78837	82797
	5	73527	81013
	7	78291	81962
	9	71598	82680
	11	71845	79929
	13	86464	82928
	15	83775	83653
	17	92380	91019
	19	83223	80964
	21	81461	80911
Aggregated throughput		801401	847856
Fairness Index (Section 3.6)		0.993	0.998
Average RTT [ms]		521	275
Average Window Size [bytes]		4696.27	2701.9
Average Queue Size [bytes]		24601.71	12927.28
Total packets dropped at network layer		122	0
Packet drops due to retransmission limit		0	0

Additionally, there are no packet drops for the PI rate controller either at network layer or at MAC layer. This is due to the fact that the adaptive PI rate controller is able to control the transmission rate avoiding dropping packets in network layer and having an indirect effect on MAC layer. Moreover, since the nodes are far from each other (as explained in Section 3.6.1.1), the effect of interference and contention is much reduced in the wireless channel.

With this scenario we observed and verified the performance of the PI rate controller in a controlled environment forcing to cause congestion in one intermediate node. This was done in order to evaluate its ability to control the source nodes transmission rate by means of a calculation and update of the advertised window size at the intermediate node. Through these

observations we will later test the PI rate controller performance in different ad-hoc scenarios increasing their complexity in terms of number of nodes, flows and mobility.

This scenario was mainly analyzed and proposed in order to evaluate the general performance of the PI rate controller in a restricted wireless scenario where other factors such as interference of multiple nodes, hidden node problem, cross traffic, multiple hops, and random and dynamic topologies are not present. Because these conditions are characteristics of Ad-hoc networks the next scenarios intend to test the PI rate controller under this environment. However, the statistics and performance measures presented for these scenarios will mainly reflect the global results of the different simulations, taking into consideration aggregated values for the throughput and average values for the rest of the performance metrics at each node.

4.3.2. Scenario 2: Four-Hop Connection, Chain Topology (5 nodes, 1 flow)

In this experiment, we connect the source and destination nodes via three intermediate nodes that are aligned as shown in Figure 3.3 and whose distance between them is 250 m. The source node opens an FTP session and continuously sends packets of 512 bytes size to the destination node. The purpose is to observe the performance of the PI rate control algorithm with the number of hops between a source node and its respective destination node.

Table 4.3: Simulation Results of the Four-Hop Chain Topology (5 nodes, 1flow)

Average throughput [kbps]		145.657	159.627
Aggregated throughput [Kbps]		145.657	159.627
Fairness Index (Section 3.6)		N/A	N/A
Average RTT [ms]		40.1	63.2
Average Queue Size [bytes]	[Node1]	256.87	94.74
	[Node2]	243.30	83.03
	[Node3]	79.28	29.38
	[Node4]	29.38	8.99
	[Node5]	0.74	0.40
Total packets dropped at network layer		0	0
Number of Packet drops due to retransmission limit	[Node1]	78	25
	[Node2]	255	43
	[Node3]	54	10
	[Node4]	8	5
	[Node5]	1	0

It is seen from the results of Table 4.3 that the PI rate controller achieves a better average throughput (159.63 Kbps) than that of TCP New Reno (145.66 Kbps) which in general represents a 9.6% improvement. The result values of Table 4.3 are the average of 3 different simulation runs using 3 different seeds. This enabled us to obtain a 95% confidence interval with an error criterion of ±18.87 Kbps.

The PI rate controller also shows in comparison to TCP New Reno, a better control of the queue size at each node along the path. TCP New Reno keeps an average total of 609.57 packets in the queue while with the PI rate controller the average total number of packets in the queue is 216.54. This is particularly important because the more packets in queue, the more packets to be sent at a particular moment and therefore the more contention in the wireless channel. It is seen that MAC layer drops increase due to contention in the wireless channel; however, with the PI rate controller, the packet drops due to retransmission limit at MAC layer are fewer than those of TCP New Reno (396 drops with TCP New Reno versus 83 with the PI algorithm). This is due to the indirect effect that the control of the transmission rate of the source produces.

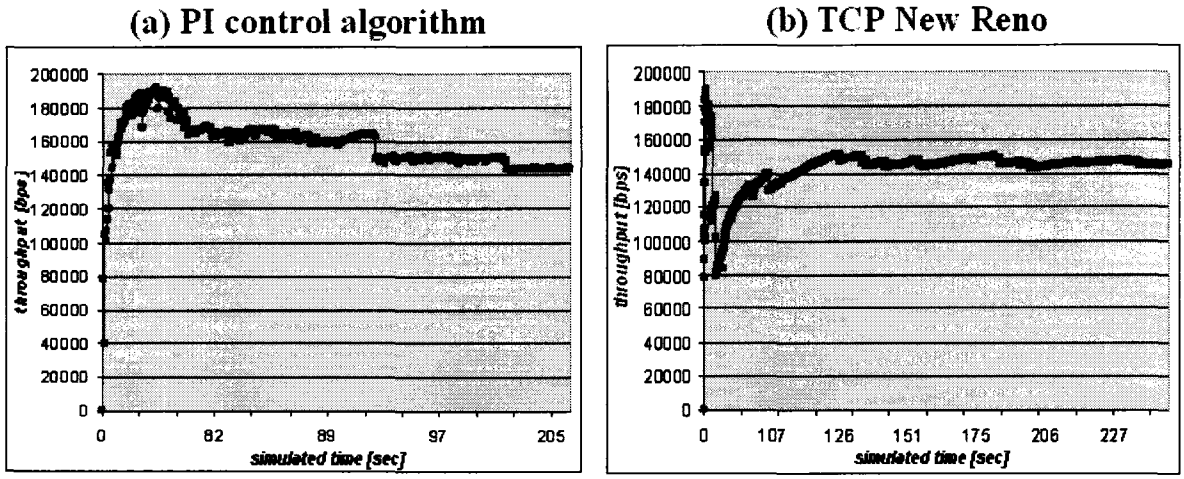

Figure 4.7: Average Throughput Performance of the Four-Hop Chain Topology (5 nodes, 1flow) Scenario.

In this scenario, however, we can see the average RTT of the PI algorithm is longer than that from TCP, probably due to the fact that the number of packets to be sent by the source are controlled based on the minimum value of the advertised window size from all the nodes along

a path. The value of the aggregated throughput in Table 4.3, as well as in all the following tables of scenarios with just one flow, is the same as the average throughput because just one flow is considered. For the same reason in the same tables mentioned above, there is no fairness index value available as there is just one flow sharing all the available bandwidth.

Figure 4.7 shows for both TCP New Reno and the PI rate controller, the average throughput of the controlled FTP source node in [bps]. The FTP source node starts to increase its transmission rate until it reaches and maintains a steady state. At this point, the source node achieves a transmission rate of 159.627 Kbps for the PI rate controller while the same source node using TCP New Reno reaches at the steady state a rate of 145.657 Kbps.

Table 4.4: Simulation Results of the Nine-Hop Chain Topology (10nodes, 1flow)

Average throughput [kbps]		46.224	66.881
Aggregated throughput [kbps]		46.224	56.881
Fairness Index (Section 3.6)		N/A	N/A
Average RTT [ms]		150.7	151.6
Average Queue Size [bytes]	[node1]	16.14	113.57
	[node2]	29.87	174.42
	[node3]	12.27	90.20
	[node4]	4.76	24.62
	[node5]	7.20	60.07
	[node6]	4.78	35.17
	[node7]	3.21	32.22
	[node8]	10.94	68.42
	[node9]	7.39	33.37
	[node10]	0.064	1.17
Total packets dropped at network layer		20	0
Number of Packet drops due to retransmission limit	[node1]	33	9
	[node2]	121	84
	[node3]	32	18
	[node4]	12	2
	[node5]	45	17
	[node6]	22	1
	[node7]	9	5
	[node8]	81	50
	[node9]	40	10
	[node10]	2	0

4.3.3. Scenario 3: Nine-hop Connection, Chain Topology (10 nodes, 1 flow)

In this experiment, we connect again the source and destination nodes in a chain topology but now the number of intermediate nodes is increased as shown in Figure 3.7.

The distance between nodes is also around 250 m. In this first case we will see the behavior of the PI rate control algorithm with just one FTP flow where the source node continuously send packets of 512 bytes size to the destination node.

Table 4.4 that the PI rate controller still achieves better average throughput than that of TCP New Reno (56.881 Kbps vs. 46.224 Kbps) without packet drops at network layer. This increase represents a 23.05% throughput improvement. The result values of Table 4.4 are the average of 3 different simulation runs using 3 different seeds. This enabled us to obtain a 95% confidence interval with an error criterion of ±12.85 Kbps.

The average throughput decreases with the number of nodes. However, it is seen from the effect of MAC layer drops and contention in the wireless channel start to increase as more nodes are contending to get the best available throughput. As mentioned before, because there is only one flow in this scenario, the fairness index value does not apply and has no meaning.

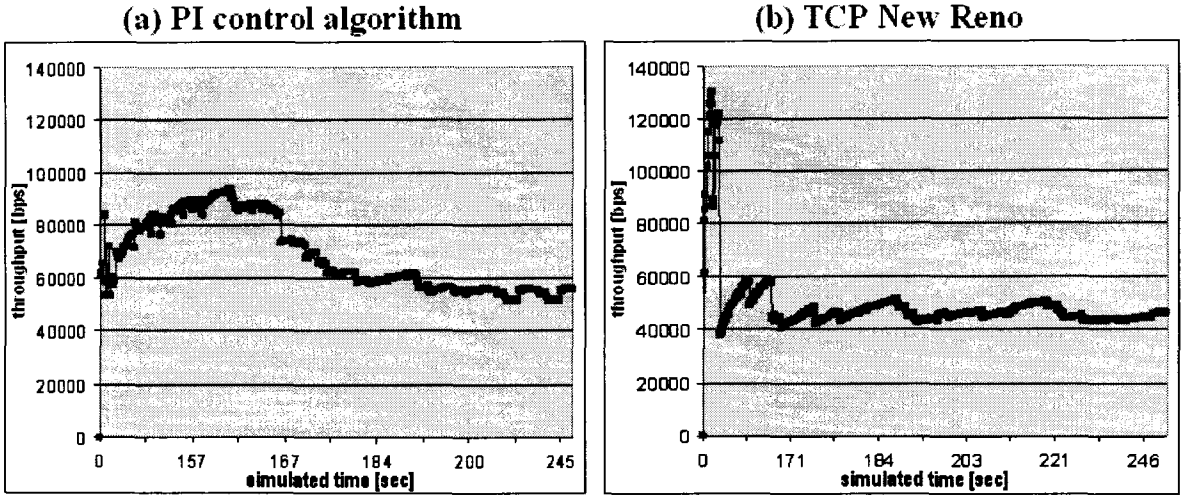

Figure 4.8: Average Throughput Performance of the Nine-Hop Chain Topology (10 nodes, 1flow) Scenario.

Figure 4.8 plots the average throughput in [bps] for the scenario of a Nine-hop chain topology with 10 nodes and one flow. We can observe the throughput performance comparison

between the PI rate controller and TCP New Reno as well as the enhancement mentioned above along the simulation time

4.3.4. Scenario 4: Four-hop Connection, Chain Topology (5 nodes, 4 flows)

In this experiment, we connect again the source and destination nodes via three intermediate nodes aligned as shown in Figure 3.8. The distance between them is 250 m. In this case, the source node opens four FTP sessions to continuously send packets of 512 bytes size to the destination node. The aim is to observe the performance of the PI rate controller with an increase of simultaneous flows between the source and destination.

It is seen from Table 4.5 that the PI rate controller achieves better average throughput (around 37.14% improvement) without packet drops at network layer. However, the effect of MAC layer packet drops and contention in the wireless channel increases as more flows contend to get the best available throughput. The result values of Table 4.5 are the average of 3 different simulation runs using 3 different seeds. This enabled us to obtain a 95% confidence interval with an error criterion of ±8.45 Kbps.

Table 4.5: Simulation Results of the Four-Hop Chain Topology (5nodes, 4flows)

Average throughput [kbps]	[Flow Id5]	29.23	67.313
	[Flow Id4]	43.13	43.906
	[Flow Id3]	45.28	44.410
	[Flow Id2]	24.03	38.665
Aggregated throughput [kbps]		141.67	194.294
Fairness Index (Section 3.6)		0.939	0.950
Average RTT [ms]		105.47	128.9
Average Queue Size [bytes]	[node1]	520.37	498.04
	[node2]	958.84	845.44
	[node3]	191.15	224.89
	[node4]	39.24	55.99
	[node5]	1.88	2.54
Total packets dropped at network layer		0	0
Number of Packet drops due to retransmission limit	[node1]	73	84
	[node2]	181	191
	[node3]	26	35
	[node4]	18	31
	[node5]	1	6

We can observe that in terms of fairness both TCP New Reno and the PI rate controller show similar abilities to provide fair amount of bandwidth. However, it is observed that some flows receive more available bandwidth.

Additionally, the increase in throughput observed whenever the PI rate controller is used, has an effect on the RTT value, which tends to increases. The RTT value for the PI rate controller is bigger than the average RTT obtained by using TCP New Reno.

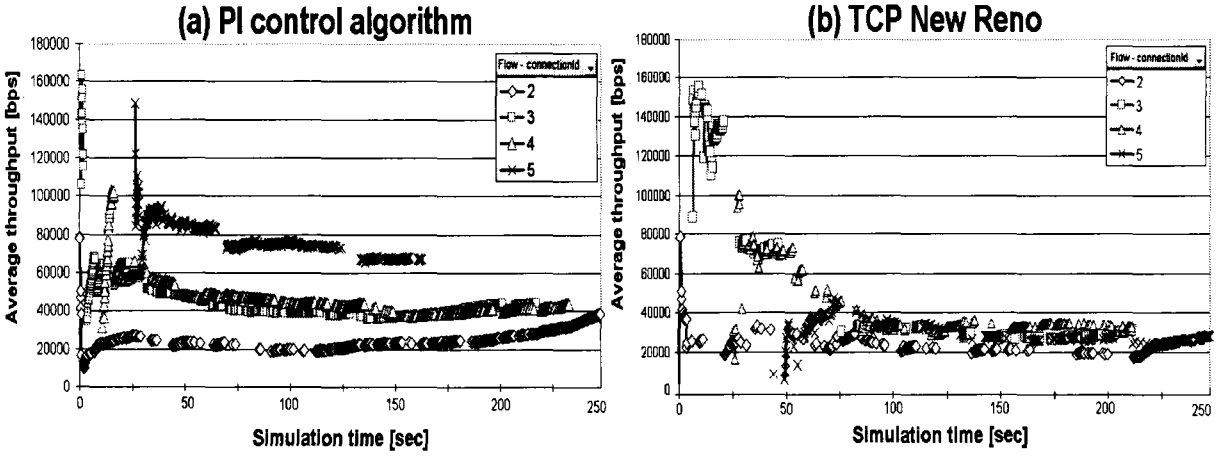

Figure 4.9: Average Throughput Performance of the Four-Hop Chain Topology (5 nodes, 4 flows) Scenario.

Figure 4.9 shows the throughput performance of simultaneous flows and indirectly their fairness behavior (defined on Section 3.7) in contending the wireless channel for both TCP New Reno and the PI rate controller. These curves compare the time-evolution of the average throughput sustained in Flows #2 to #5 (Table 4.5) along the duration of the simulation. Flows #2 and #3 experienced throughput peaks of ~155000 bps and ~77000 bps, respectively at the beginning of the simulation. However, as they start to contend with each other the throughput decreases to reach their average throughput values and at the same time to allow the other two flows to start their transmission. This initial contention provokes a delay in the transmission of Flows #4 and #5. They start the transmission at about t=10s and t=20s, respectively. Flow#3 has similar performance to Flow#2. Flow#2 has the lowest starting point and hobbles around 20,000 bps. However, it strangely rises to about 40,000 bps while packets from Flow #5 are not received at the destination (probably due to contention, route failures, packet drops, etc...) allowing for

more bandwidth sharing among the rest of the flows. At the beginning of the simulation, Flow #5 experienced a delay due to contention in the wireless channel from the transmission of the other flows. However, its throughput rises quickly (to ~100,000 bps) in about 20 seconds and presents a sharp peak (~150,000 bps) at about t=30s before leveling off to about 60,000 bps. In all, the throughput of Flow #5 is higher but less constant in comparison to the other flows. The reason could be that after a high transmission of packets, these are not effectively received at the destination during the end of the simulation due to contention or queuing processing. Flow#4 on the other hand, rises much less (to about 70,000 bps) before leveling off to about 40,000 bps at t=250s. In all, Flow #2 has the lowest throughput because whenever the other flows are transmitting and sharing the same wireless channel this is not able to acquire more of the available bandwidth. With TCP New Reno the performance seems to better in terms of sharing the bandwidth among the simultaneous flows. However, there are moments where there is no continuous TCP transmission from simultaneous flows due to more contention on the wireless channel.

4.3.5. Scenario 5: Nine-hop Connection, Chain Topology (10 nodes, 4 flows)

In this experiment, we connect again the source and destination nodes in a chain topology; with the number of intermediate nodes increased to nine as in the scenario above but now the number of simultaneous flows is increased too as shown in Figure 3.9. The distance between adjacent nodes is around 250 m. There are four simultaneous FTP flows with different start time. The source node continuously sends packets of 512 bytes size to the destination node.

Again it is shown that the average throughput decreases with the number of nodes and moreover with the number of simultaneous flows which contend in the wireless channel. However, it is seen from Table 4.6 that the PI rate controller still achieves better average throughput (107.68 Kbps) than that of TCP New Reno (99.86 Kbps) as shown with the average aggregated throughput value counted as the sum of all the flows and which represents a general enhancement of 7.83%.

The result values of Table 4.6 are the average of 3 different simulation runs using 3 different seeds. This enabled us to obtain a 95% confidence interval with an error criterion of ±13.48

Kbps. The effect of MAC layer drops and contention in the wireless channel also increase as more nodes are contending to get the best available throughput.

Table 4.6: Simulation Results of the Nine-Hop Chain Topology (10 nodes, 4 flows)

Average throughput [kbps]		31.46	13.945
		17.00	19.696
		29.55	43.472
		21.85	30.568
Aggregated throughput [kbps]		99.86	107.683
Fairness Index (section 3.6)		0.94	0.86
Average RTT		177.8	231.8
Average Queue Size [bytes]	[node1]	57.52	170.18
	[node2]	102.03	503.38
	[node3]	51.73	214.54
	[node4]	16.95	46.14
	[node5]	44.69	125.33
	[node6]	22.12	96.93
	[node7]	13.78	54.56
	[node8]	38.19	129.51
	[node9]	22.38	54.80
	[node10]	0.20	0.94
Total packets dropped at network layer		0	0
Number of Packet drops due to retransmission limit	[node1]	49	40
	[node2]	208	188
	[node3]	67	53
	[node4]	11	6
	[node5]	60	45
	[node6]	28	9
	[node7]	29	13
	[node8]	101	112
	[node9]	51	19
	[node10]	1	0

Figure 4.10 plots the average throughput performance in [bps] for the scenario of a Nine-hop chain topology with 10 nodes and 4 simultaneous flows. This figure and Table 4.6 help to qualitatively show, for the four flows, how fair both algorithms (TCP and the PI rate controller) are in sharing the bandwidth while contending for the wireless channel. These curves compare the time-evolution of the average throughput sustained in Flows #7 to #10 (Table 4.6) along the

duration of the simulation. The flows attain different throughput due to the different delays the packets encounter while contending for the wireless channel resources.

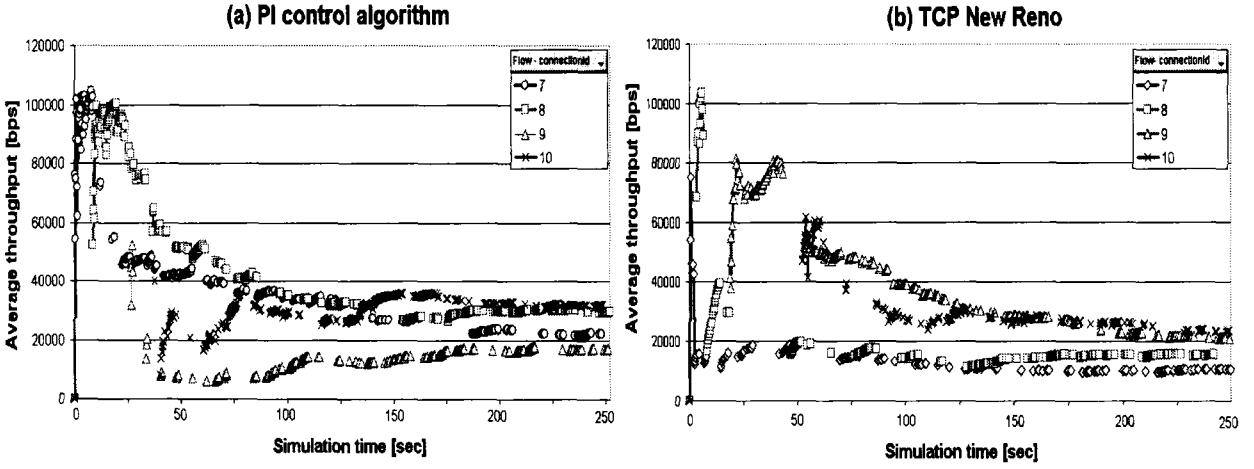

Figure 4.10: Average Throughput Performance of the Nine-Hop Chain Topology (10 nodes, 4flows) Scenario.

From the beginning of the simulation, all the flows start contending among each other to get the available wireless channel bandwidth. For the case of the PI rate controller, Flows #7 and #8 reached throughput peaks of ~100000 bps. However, as the contention increases the throughput decreases. They reach their average throughput values and at the same time allow the other two flows to start their transmissions. This initial contention creates a delay in the transmission of Flows #9 and #10 which start transmitting at about t=25s and t=35s, respectively. Flows #7 and #8 have similar throughput performance. Flow #9 has the lowest starting point and hobbles around 8,000 Kbps. However, it rises to about 16,000 Kbps by the end of the simulation. This behavior could be explained by the impact of flows contending the channel and the effect of the adaptive PI rate controller. Since Flows #7 and #8 won initially most of the available bandwidth, after a delay and reattempt of the other flows to gain the wireless channel, as well as the nodes' update of an optimal window size, the direct effect on Flows #7 and #8 is a decrease in average throughput while Flows #9 and #10 start sharing the transmission and increasing their own effective throughput. This process can be observed until all the flows stabilize and reach an average throughput which is fairer to all of them.

With TCP New Reno the performance seems to be better in terms of sharing the bandwidth among the simultaneous flows. However, there are moments where there is no continuous TCP transmission from simultaneous flows due to more contention on the wireless channel. In comparison to TCP New Reno, the PI rate controller observes a more constant transmission or effective throughput among the different flows. However, both reach a moment where most of the flows receive a fair share of the available throughput and stabilize their average values.

For both the PI rate controller and TCP New Reno, whenever the flows are transmitting simultaneously, there is a decreasing effect on the achievable throughput due to the other flows transmission. Flows #7 and #8 (1st and 2nd curves in the graph) for the PI rate controller, for example, tend to get more available bandwidth than with TCP New Reno, where the same flows #7 and #8 (3rd and 4th curves of TCP New Reno graph) receive the least throughput. Again, this depends on which flow wins the initial transmission, on how the other flows contend and the different delays the packets of each flow suffer on this situation at any point within the network.

4.3.6. Scenario 6: 50 Static Nodes Randomly Placed and Uniformly Distributed (50 nodes, 10 flows)

In this experiment, there are 50 nodes randomly located and uniformly distributed over an area of 700 m x 700 m as shown in Figure 3.7. The distance between nodes varies according to the distribution of the nodes. There are 10 simultaneous FTP flows with different start times going from 10 randomly chosen source nodes to 10 randomly chosen destination nodes. These flows continuously send packets of 512 bytes size to the destination node.

Table 4.7 shows that the average aggregated throughput of the PI rate controller (656.93 Kbps) is better than that of TCP New Reno (588.76 Kbps). This difference represents an enhancement of about 11.58%. The values given on Table 4.7 are the average of 3 different simulation runs using 3 different seeds. This enabled us to obtain a 95% confidence interval with an error criterion of ±51.9 Kbps in the value of aggregated throughput.

Table 4.7: Simulation Results of the Scenario of 10 Flows Over 50 Static Nodes Randomly Located and Uniformly Distributed

50 static nodes, 10 flows			
Parameter		TCP New Reno	PI
Average throughput [kbps]	4	79.21	76.81
	7	96.52	18.99
	16	99.09	7.51
	22	33.69	11.94
	23	23.75	187.09
	30	106.47	5.19
	31	58.01	245.39
	32	31.42	24.64
	39	38.58	74.02
	49	22.02	5.35
Aggregated throughput [kbps]		588.76	656.93
Average RTT [ms]		245.8	118.50
Total packet drops at network layer		50	21
Number of Packet drops due to retx limit		2104	831

For the PI rate controller we can see that the traffic flows show an interesting and not uniform distribution of traffic. This could be due to the different paths each source node takes to reach the destination. Based on a trace and analysis of the results obtained, it is observed that a greater throughput is assigned to those flows that have shorter round trip delay. Table 4.8 illustrates this observation by two sessions that are selected after analyzing the ten different flows.

Table 4.8: Examples of RTT Average Values for Scenario 6 Using the PI Rate Control Algorithm

50 static nodes, 10 flows - PI Rate Control Algorithm		
Node	RTT [ms]	Throughput [Kbps]
22	1059	11.94
31	195	245.39

Table 4.7 shows in addition that the packet drops on both Network and MAC layers are greater for TCP New Reno than for the PI rate controller. The effect of MAC layer contention in the wireless channel as well as the interference among nodes also increases as more nodes are contending to get the best available throughput. Nevertheless, the effect is less for the case of using the PI rate controller.

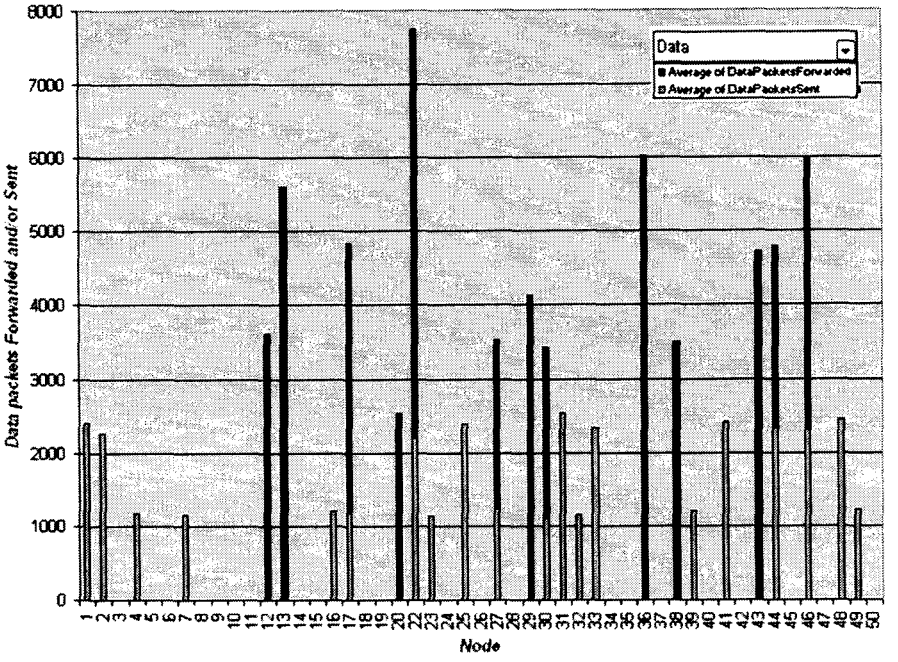

Figure 4.11: Average Number of Data Packets Forwarded and/or Sent per Node for the Scenario of 10 Flows Over 50 Static Nodes Randomly Located and Uniformly Distributed

In Scenario 6 the packets follow different routes depending on the congestion of each particular node in the network as well as the contention in the wireless channel for each group of neighboring nodes. These factors influence the throughput behavior on each node. Figure 4.11 helps to show this varied environment for a period of time (~ 90 sec) during the simulation. We can see from the graph that there are nodes with no traffic at all as there are neither packets to send nor packets to be forward. On the other hand, there are nodes that act just as sources of traffic, while others need both to send as well as forward traffic. These roles of each node affect the level of congestion, the delay for the total transmission of packets and the performance of a particular route to a destination. To give an example, let us choose again nodes 22 and 31. Figure 4.11 shows that Node 22 has both packets to send (2189 packets) and packets to forward (5548 packets) while Node 31 has just packets to send (2732 packets). Whenever a node queue size increases, the queuing and processing delays increase too, having a direct effect on the total delay along paths and hence on the total round trip time (RTT). This could be one of the reasons that explains why Node 22 experiments a longer RTT than Node 31 as showed in Table 4.8. As

we mentioned before, whenever the RTT registered is shorter, the nodes are able to achieve more throughput.

4.4. Concluding Remarks

In this chapter, the configuration settings used to run the different simulations for the evaluation of the PI rate controller in Ad-hoc networks in static scenarios were presented and described. The PI rate congestion control mechanism was compared with TCP New Reno to finally show the results of these simulations as well as an analysis and evaluation of its behavior. From the results, we observed that the PI rate congestion control in general is able to obtain better average throughput than TCP New Reno by means of controlling the source node transmission rate and therefore dealing better with the average queue size at the intermediate nodes. This improvement in average throughput is also observed whenever the number of hops and simultaneous flows is increased. In general, the improvement in throughput performance of the PI rate controller in comparison to the throughput achieved by TCP New Reno, for the static scenarios tested, is within the range of 5.6% to 23.05% varying from scenario to scenario.

The behavior of the PI rate congestion control algorithm in mobile scenarios will be tested on Chapter 5.

Chapter 5
Mobile Scenarios

Having evaluated the static cases, we shall now evaluate the PI congestion control algorithm under mobile scenarios. The performance measures are the same as those used in the previous section, but emphasizing on the improvement of network throughout.

5.1. Simulation Environment

Like the static scenarios, all the simulations were performed using Qualnet [Qual05] with the models described in Chapter 3, but with nodes mobility added. The simulations are run with and without the PI congestion control algorithm and again compared with TCP New Reno [FlHe04]. The generic FTP application with a packet size of 512 bytes is used.

Same as the static scenarios, all network systems are simulated for a time of 250 seconds. The result values are the average of 3 different simulation runs using 3 different seeds. There are about 50000 samples per seed. This enabled us to obtain a 95% confidence interval for each particular scenario. Again the upper and lower confidence interval bounds for each scenario vary according to the scenario.

5.2. Simulation Configuration Settings

This section briefly presents the values in Qualnet that were used for each of the different communication layers. The packet size used is set to 512 bytes. The detailed values are shown in Table 5.1. The configuration settings are mainly the same as those for the static scenarios of Chapter 4. Modifications are made to include mobility and the difference is listed in Table 5.1 below.

5.3. Mobile Scenarios

The mobile scenarios have been described in detail in Section 3.4.1. The aim of these scenarios is to observe how the PI congestion control algorithm performs in order to cope with congestion control arriving from cross traffic and node mobility. We increase the mobility speed of nodes in order to test more complex scenarios for realistic wireless Ad-hoc networks.

Table 5.1: Simulation Configuration Settings

********** General Configuration **********		
Parameter	Value	Observations
TERRAIN-DIMENSIONS	(700, 700) randomly placed and uniformly distributed topology	The size of the physical terrain in which the nodes are being simulated.
DUMMY-NUMBER-OF-NODES	- 5 nodes (Scenario 7) - 50 nodes (Scenarios 8 to 12)	The number of nodes being simulated.
NODE-PLACEMENT	UNIFORM	The node placement strategy.
MOBILITY	RANDOM-WAYPOINT FILE-BASED	• This mobility model randomly selects destinations and speeds for each node. • The mobility pattern is defined in a time-ordered file containing the destination coordinates at a particular time.
MOBILITY-WP-PAUSE	30s	Specifies the amount of time the nodes will pause after they reach their selected destinations.
MOBILITY-WP-MIN-SPEED	0.5, 1, 5, 10, 15 [m/s]	Used only for Random-Waypoint Model.
MOBILITY-WP-MAX-SPEED	1, 5, 10, 15, 20 [m/s]	Used only for Random-Waypoint Model
MOBILITY-POSITION-GRANULARITY	0.5 [m]	It determines the resolution at which node positions are updated. Smaller figures mean node positions are updated more often at smaller increments.

Section 5.3.1 examines a simple scenario depicted in Figure 3.11 with a source node moving back and forth with respect to a destination. Sections 5.3.2 to 5.3.6 examine five more complicated scenarios consisting of 50 nodes moving randomly with different mobility speeds according to the Random Way Point mobility model. For these five scenarios, we consider the topology depicted in Figure 3.12. The distance between nodes varies according to the distribution of the nodes and the speed configured. Ten simultaneous FTP flows will continuously send packets of 512 bytes size. The moving speed of the nodes is the parameter that will be changing in each scenario.

Our measurement and observation are provided for each mobile scenario. The result values for each of these scenarios are the average of 3 different simulation runs using 3 different seeds. This enables us to obtain a 95% confidence for each of these scenarios.

5.3.1. Scenario 7: Mobile Source Node Moving Towards Destination Node (5 nodes, 1 flow, 3m/s)

The speed with whom the nodes move in this scenario is 3 m/s. We start with a simple scenario depicted in Figure 3.11 and consist of one source node moving back and forward towards a destination with a speed of 3 m/s while constantly establishing an FTP session.

Table 5.2: Simulation Results of the Scenario of a Mobile Source Moving Towards a Destination (5nodes, 1flow, 3m/s)

Average throughput [kbps]		372.65	397.43
Aggregated throughput [Kbps]		372.65	397.43
Fairness Index (Section 3.6)		N/A	N/A
Average RTT [ms]		91.22	86.13
Average Queue Size [bytes]	[Node1]	6290.01	5188.57
	[Node2]	455.53	386.39
	[Node3]	97.74	90.99
	[Node4]	741.20	620.22
	[Node5]	31.05	41.11
Total packets dropped at network layer		0	0
Number of Packet drops due to retransmission limit	[Node1]	51	47
	[Node2]	0	0
	[Node3]	0	0
	[Node4]	9	7
	[Node5]	77	56

Table 5.2 shows that the average aggregated throughput of the PI rate controller (397.43 Kbps) is better than that of TCP New Reno (372.65 Kbps). This improvement in throughput performance represents a 6.65% increase over that of TCP New Reno with a confidence interval of ±5.99 Kbps for the value of aggregated throughput for the PI rate algorithm. Table 5.2 additionally shows that there are no packet drops on network layer and that the average number of packet drops at MAC layer is greater for TCP New Reno than for the PI rate controller.

5.3.2. Scenario 8: 50 Mobile Nodes (10 flows, 1m/s)

The moving speed of the nodes for this scenario is 1 m/s. Table 5.3 shows that the average aggregated throughput of the PI rate controller (1720.14 Kbps) is better than that of TCP New Reno (1458.9 Kbps). This improvement in throughput performance represents a 17.9% increase over that of TCP New Reno with a 95% confidence interval of ±91.91 Kbps.

Table 5.3: Simulation Results of the Scenario of 10 Flows Over 50 Mobile Nodes Randomly Located and Uniformly Distributed (1m/s)

Average throughput [kbps]	4	299.53	50.0
	7	163.25	107.84
	16	51.65	70.82
	22	154.27	182.92
	23	120.84	407.69
	30	252.36	64.3
	31	71.16	57.2
	32	103.93	582.33
	39	91.28	146.54
	49	150.58	50.51
Aggregated throughput [Kbps]		1458.90	1720.14
Average RTT [ms]		548.07	437.18
Total packet drops		22	0
Packet drops due to retransmission limit		321	232

The flows show a different distribution of traffic throughput, probably due to the different path length each source node takes to reach its respective destination as well as the different amount of interference from other nodes along this path.

Table 5.3 additionally shows that the packet drops on MAC layer for the case of the PI rate controller are less than those using TCP New Reno. Moreover, the PI rate controller has no packet drops while TCP New Reno experiences some. The effect of MAC layer contention in the wireless channel as well as the interference among nodes also increases (in terms of packet drops due to retransmission limit) as more nodes are contending to get the best available throughput. Nevertheless, the effect is less for the case of using the PI rate controller.

5.3.3. Scenario 9: 50 Mobile Nodes (10 flows, 5m/s)

The moving speed of the nodes in this scenario is 5 m/s. Table 5.4 shows that the average aggregated throughput of the PI rate controller (1986.02 Kbps) is better than that of TCP New

Reno (1682.59 Kbps). This improvement in throughput performance represents a 18.03% increase over that of TCP New Reno with a confidence interval of ±12.51 Kbps for 95% of confidence.

Table 5.4: Simulation Results of the Scenario of 10 Flows Over 50 Mobile Nodes Randomly Located and Uniformly Distributed (5 m/s)

Average throughput [kbps]	4	446.36	440.38
	7	209.56	188.71
	16	62.53	85.52
	22	108.55	167.32
	23	156.15	187.52
	30	239.78	283.77
	31	116.86	102.51
	32	75.75	41.04
	39	117.07	297.30
	49	149.94	191.94
Aggregated throughput [Kbps]		1682.59	1986.02
Average RTT [ms]		615.54	273.28
Total packet drops		52	2
Packet drops due to retransmission limit		454	130

Table 5.4 also shows that the number of packet drops has increased in both the network and MAC layers when the moving speed increases. However, the number is greater for TCP New Reno than for the PI rate controller. The traffic within the network is affected by the MAC layer contention (in terms of packet drops due to retransmission limit) in the wireless channel because the moving nodes have created more interference among each other. Depending on the mobility patterns the moving speed at a particular moment can additionally create regions of high contention when many nodes are contending to get the best available throughput within their transmission range. This negative effect is however less for the PI rate controller, due to the continuous per node update of the window size along each of the different paths.

5.3.4. Scenario 10: 50 Mobile Nodes (10 flows, 10m/s)

For this scenario the moving speed of the network nodes is increased to 10 m/s. Table 5.5 shows that the average aggregated throughput of the PI rate controller (1157.181 Kbps) is better than that of TCP New Reno (885.601 Kbps). This improvement in throughput performance represents a 23.46% increase over that of TCP New Reno for a 95% confidence

and an interval of ±75.87 Kbps. We can observe that the mobility of the nodes affect the traffic throughput of the flows. It is not uniformly distributed but its variation seems to be due to the increase in speed that makes the nodes to interfere among each other and therefore create different paths.

The increasing trend of packet drops on both network and MAC layers continue for both TCP New Reno and the PI rate controller. This is a direct effect of the node mobility. However, we can highlight that still the effect on the PI rate controller is less. Thus, its control of the source transmission rate is effectively helping to avoid congestion and indirectly reducing contention.

Table 5.5: Simulation Results of the Scenario of 10 Flows Over 50 Mobile Nodes Randomly Located and Uniformly Distributed (10m/s)

Average throughput [kbps]	4	4.721	47.257
	7	3.419	496.373
	16	14.261	0.121
	22	10.979	35.794
	23	.025	224.474
	30	21.447	91.033
	31	33.129	30.454
	32	160.419	17.425
	39	.409	67.064
	49	616.792	147.186
Aggregated throughput [Kbps]		865.601	1157.181
Average RTT [ms]		184.06	185.1
Total packet drops		135	2
Packet drops due to retransmission limit		5187	1684

5.3.5. Scenario 11: 50 Mobile Nodes (10 flows, 15m/s)

The speed of the nodes in this scenario is 15 m/s. Table 5.6 shows that the average aggregated throughput of the PI rate controller (987.591 Kbps) is better than that of TCP New Reno (953.75 Kbps). This improvement in throughput performance represents a 3.54% increase over that of TCP New Reno at a confidence interval of ±117.9 Kbps.

The increasing mobility of the nodes keeps on decreasing the flows' traffic throughput because whenever the nodes move close to each other the interference reduces the ability to transmit simultaneously and causes the transmission to experience larger delays. The PI

adaptive control seems to help on this effect because the increasing trend of packet drops on both network and MAC layers is less than that of TCP New Reno.

Table 5.6: Simulation Results of the Scenario of 10 Flows Over 50 Mobile Nodes Randomly Located and Uniformly Distributed (15m/s)

Average throughput [kbps]	4	69.407	4.820
	7	271.972	2.193
	16	1.833	60.448
	22	73.386	64.991
	23	38.951	11.794
	30	131.882	37.381
	31	36.563	16.017
	32	84.589	186.098
	39	6.186	92.498
	49	238.986	511.351
Aggregated throughput [Kbps]		953.755	987.591
Average RTT [ms]		206.0	142.5
Total packet drops		571	59
Packet drops due to retransmission limit		3783	1551

5.3.6. Scenario 12: 50 Mobile Nodes (10 flows, 20m/s)

This scenario is the same as the ones before but here the nodes move with a speed of 20 m/s.

Table 5.7: Simulation Results of the Scenario of 10 Flows Over 50 Mobile Nodes Randomly Located and Uniformly Distributed (20m/s)

Average throughput [kbps]	4	7.82	34.84
	7	6.52	11.79
	16	5.59	18.96
	22	.122	384.40
	23	7.39	101.44
	30	9.29	67.47
	31	102.30	27.91
	32	227.15	73.67
	39	17.61	0.96
	49	314.44	144.73
Aggregated throughput [Kbps]		698.23	866.17
Average RTT [ms]		154.5	82.6
Total packet drops		318	50
Packet drops due to retransmission limit		4755	1103

Table 5.7 shows that the average aggregated throughput of the PI rate controller (866.17 Kbps) is better than that of TCP New Reno (698.23 Kbps). The throughput performance in general deteriorates with the node mobility speed. However, the PI rate controller is able to better cope with congestion in Ad-hoc networks than TCP New Reno because it shows a better throughput under the same scenario conditions. The throughput difference between them represents a 24.05% increase of the PI rate controller over that of TCP New Reno under 95% confidence level and within an interval of ±160.4 Kbps.

Again we can notice the effect of mobility in the performance. The nodes are moving faster, therefore the average RTT in some cases results to be shorter whenever the nodes are closer and become easily reachable without the need of passing through many different multiple hops. This causes the flows traffic throughput to vary as the route length changes. These route length changes have an effect on the nodes. It makes some of them to receive and/or forward more packets, creating spots of high contention on the wireless channel as well as of congestion.

Table 5.8: Examples of RTT Average Values for Scenario 9 Using the PI Rate Control Algorithm

Average throughput [kbps]	22	384.40	34.47
	39	0.96	396

As mentioned above, the values of RTT for each of the 10 sessions, due to different paths, route length, contention and mobility, influence the throughput behavior. By analyzing the simulation results, we observed in some cases that the flows with greater throughput have shorter round trip delays. To show this behavior we chose two of the ten different sessions of this scenario to provide an example of this observation as depicted in Table 5.8.

5.4. Concluding Remarks

In this chapter, the configuration settings used to run the different simulations for the evaluation of the PI rate controller in Ad-hoc networks in mobile scenarios were presented and described. The PI rate congestion control mechanism was compared with TCP New Reno to show the results of these simulations as well as an analysis and evaluation of its

behavior. From the results, we observed that the PI rate congestion control in general is able to obtain better overall aggregated average throughput than TCP New Reno by means of controlling the source node transmission rate along the paths and therefore dealing better with the average queue size at the intermediate nodes, as well as with the average RTT.

For most of the cases where the speed of the nodes is increased, it is observed an improvement in average throughput; the range of improvement is from 3.54% to 24.05 % depending on the scenario. The moving speed definitely has an effect on the performance. This will be studied in the next chapter.

Chapter 6
Performance Comparison

Having done all the measurements, we present in this chapter the performance of the PI congestion control algorithm under different node speeds, and compare it with the performance of TCP. We also compare the adaptive PI rate controller performance when using different ad-hoc routing protocols AODV and DSR. The intent is to show the effect of the interaction of routing in the network layer with other communication layers such as the MAC and the transport layers.

6.1. Comparison Under Different Node Mobility

In this section the performance of TCP New Reno and the PI congestion control algorithm are compared under different node speeds. The topology consists of 50 mobile nodes randomly located and uniformly distributed over an area of 700m x 700m as depicted in Figure 3.12.

6.1.1. Throughput

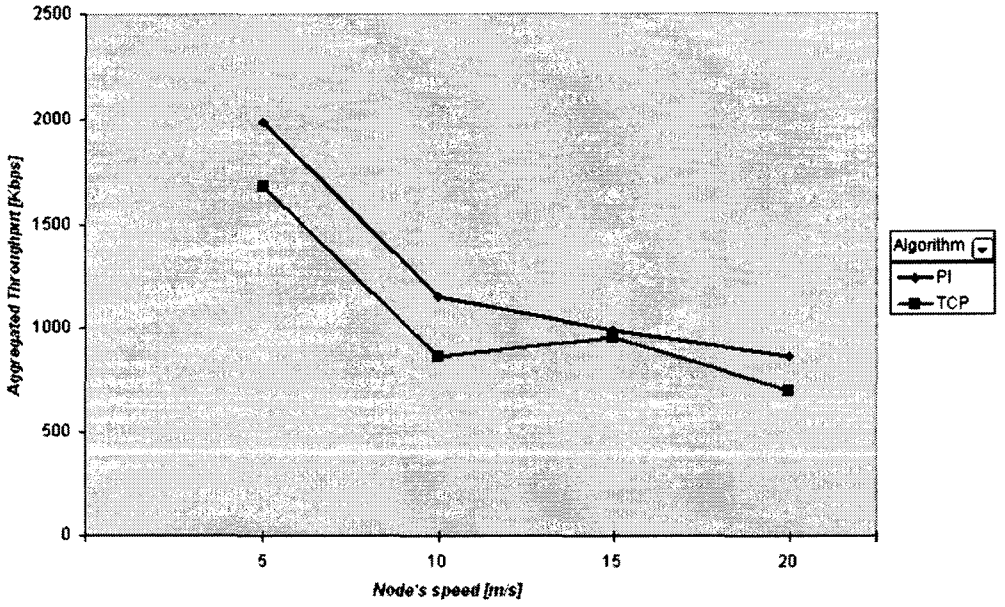

Figure 6.1: Average Throughput Performance of TCP and the Adaptive PI rate Controller under Different Node Speeds.

Figure 6.1 compares the average throughput performance of TCP New Reno and the Adaptive PI rate controller under different node speeds. We can see that in general the average aggregated throughput for both TCP New Reno and the PI rate controller decreases with node mobility. The throughput of the PI rate controller decreases from 1986 Kbps and slows down to 866 Kbps at about 20 m/s. The TCP throughput has similar trend but a lower throughput (~300 Kbps less). For most of the moving speeds the throughput of the PI rate controller is better than that of TCP New Reno. These observations suggest that the PI adaptive control of the sources transmission rates is providing better performance and resource utilization in the network.

For certain mobility speeds the throughput could also increase when the speed is increased. This could happen when the timing of TCP and MAC retransmissions, from certain node positions in the network, results in the re-establishment of packet flows at a higher mobility speed. The throughput increase could also happen when the rate of link failures is low despite the nodes are moving fast. For example, when two nodes are moving together, the link between them will not break, regardless of their speed. Another factor could be that the source and destination nodes are at most two or three hops away. The throughput becomes less whenever the number of hops is increased.

This probably is the case of TCP New Reno, which has an interesting behavior at 15 m/s. The throughput performance is slightly better than that of a lower speed (10m/s). This could be due to the drawbacks TCP shows whenever there is burstiness in the transmission. Burstiness generally creates more contention in the wireless channel especially under high load conditions, which directly reduces the throughput in the transmission. Therefore, if at the speed of 10m/s there were more events of burstiness in the transmission, even if the node speed was lower than that of 15 m/s, it could be possible to have more effective throughput at a higher node speed.

6.1.2. RTT

Figure 6.2 shows the evolution of the average RTT under different levels of mobility. We can observe that for most cases the PI rate controller has a lower average RTT than TCP New Reno. This is due to the fact that by adaptively controlling the number of packets the source nodes and every intermediate node along the path could send, the PI rate controller avoids

creating congestion in particular nodes along the path. In such way, by not having congested nodes, contention in the channel is reduced and packets are able to reach their destinations effectively. This has an effect on the total delay the packet experience along the paths as the less congested nodes the fewer packet drops. The RTT also tends to be shorter as the speed of nodes increase, allowing them to reach their destination nodes faster.

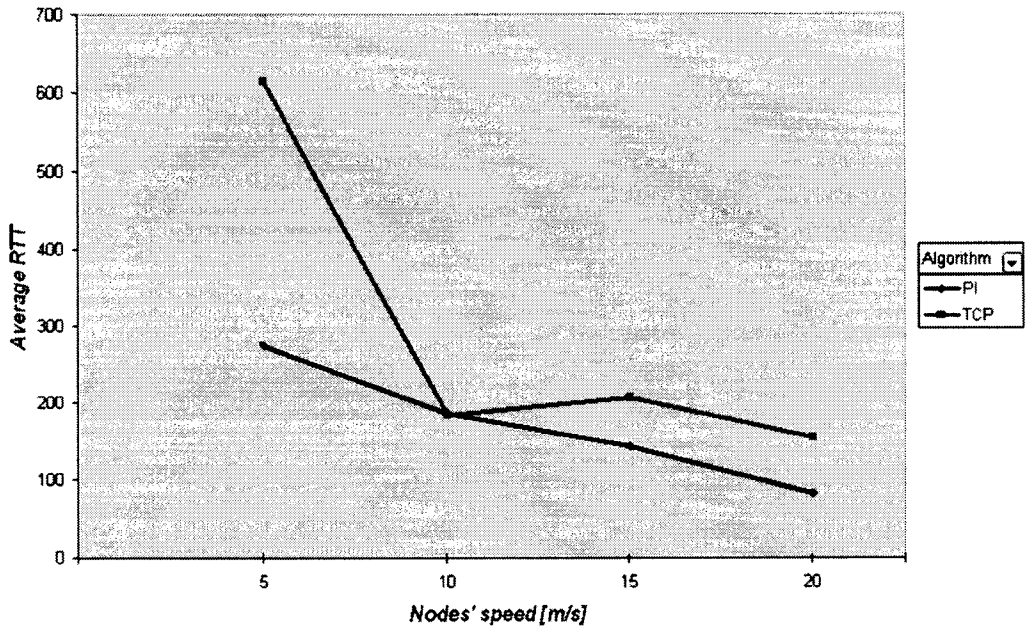

Figure 6.2: Average RTT Performance of TCP and the Adaptive PI Rate Controller under Different Node Speeds.

Again for the case of TCP New Reno there is an interesting behavior. The RTT drops from about 600 ms at 5 m/s to about 200 ms at 10 m/s. This is due to the increase in mobility to twice the speed. However, the interesting observation is the RTT becomes longer (210ms) at 15 m/s. This is supported by the observation in Figure 6.1. Usually the round trip delay is affected by the propagation, the processing and the queuing delays. As we saw in Figure 6.1, the network can provide a higher effective traffic throughput at 15m/s than at 10m/s. This means that there were more packets in transit within the network for that particular mobility pattern. Therefore, the queues in each node were longer, causing the different delays to increase causing to have a low effective throughput. Another factor could be that the source and destination nodes were less than two or three hops away (Recall that the nodes'

transmission range is ~250 m. Therefore, the nodes' coverage while moving, overlaps within the 700x700 m network area under study), thus having a higher throughput. For the case of the PI rate controller, the average RTT decreases according to the mobility increase. This shows that the control of the transmission rate and the window size update at each node helps to avoid having bottleneck nodes with high congestion that could increase the RTT on different routes.

6.1.3. Network Layer Packet Drops

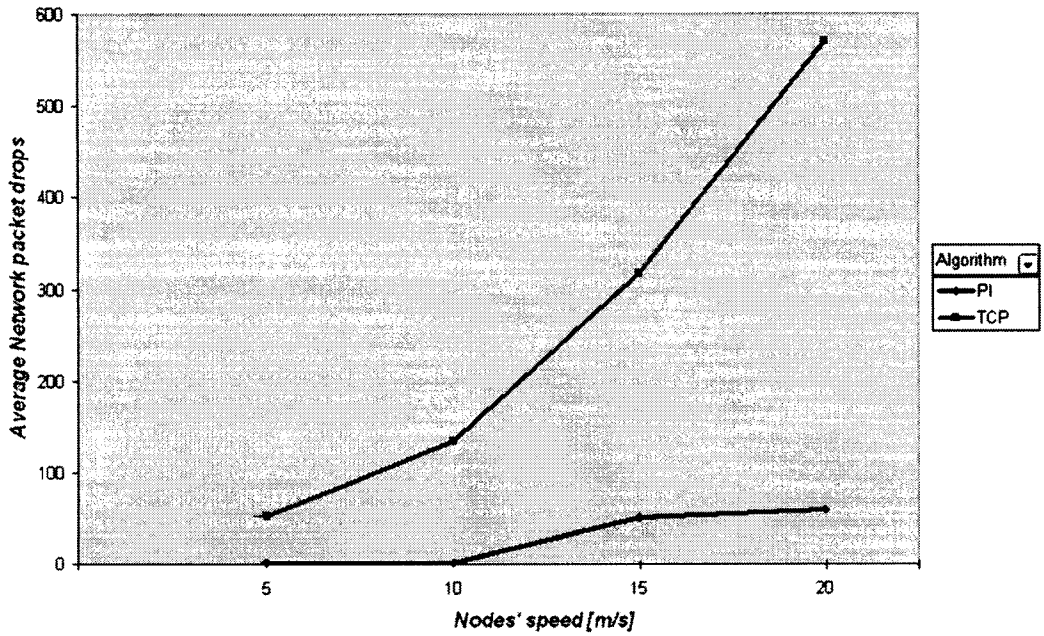

Figure 6.3: Average Number of Network Layer Packet Drops of TCP and the Adaptive PI Rate Controller under Different Node Speeds.

Figure 6.3 compares the evolution of the packet drops at network layer of TCP and the PI rate controller under different levels of mobility. The PI rate controller starts with no packet drops and slightly increases this number when the mobility speed keeps increasing. However, this does not reach more than 100 network layer packet drops. TCP in contrast suffers from packet drops always. It starts with a small number of packet drops (~50) and this number appears to have an exponential increase as a function of node mobility. For the PI rate controller the average number of packet drops is negligible at low mobility (5m/s to 10 m/s), then increases but appears to level off beyond 20 m/s. This is due to the fact that updating the advertised

window size of each node based on the update of its instantaneous queue size, adaptively and efficiently control the number of packets the nodes are able to handle without congestion and therefore without drop of packets.

For both, TCP New Reno and the PI rate controller, the higher the mobility the more packets were dropped. This is due to the fact that whenever the nodes are moving faster the network topology also changes, making it more difficult for the network layer to update the routes the packets can take to their destinations. When the speed is increased route failures happen more quickly, resulting in packet losses, and frequent route discoveries.

6.1.4. MAC packet drops

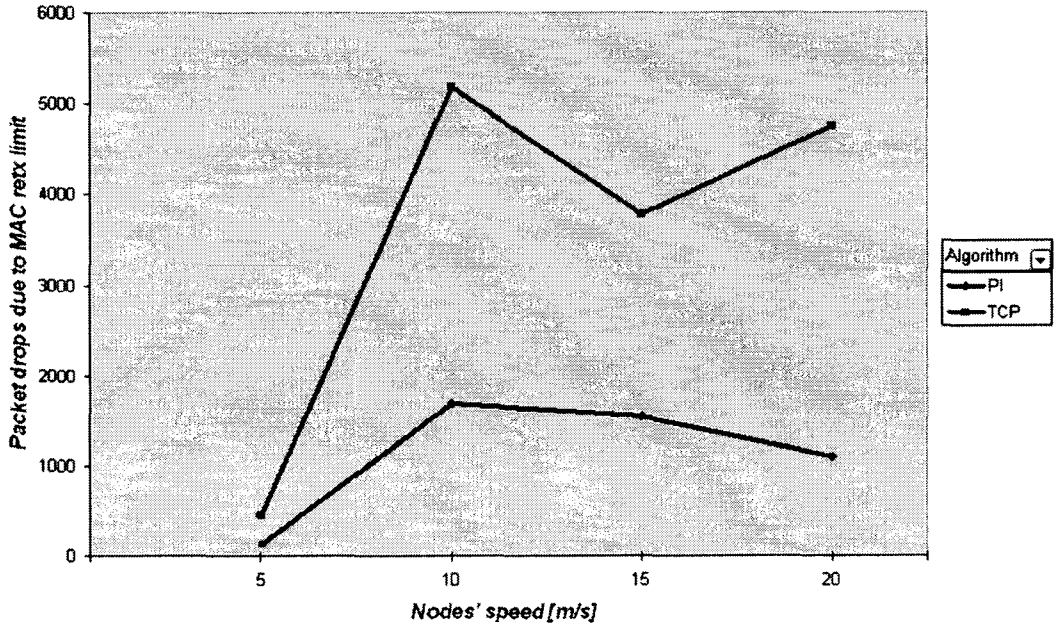

Figure 6.4: Average Number of Packet Drops due to MAC Retransmission of TCP and the Adaptive PI Rate Controller under Different Node Speeds.

Figure 6.4 compares the evolution of the average number of packet drops at MAC layer for TCP and the PI rate controller under different levels of mobility. The PI rate controller starts with almost no packet drops and increases as the mobility speed increases to 10 m/s. From this point, though seemingly constant (~150 packet drops), the packet drop does show a slight decrease.

TCP in contrast suffers from packet drops increase and decrease at different mobilities. TCP starts with a sharp increase in packet drops due to MAC retransmission when the mobility is increased from 5m/s to 10m/s. The packet drops decreases to about 4000 packets at 15 m/s but back up to 5000 packets again at 20m/s. The temporary decrease can be partly attributed to the retransmission limit while other fluctuations may be caused by the node positions in the network because certain areas may have more dispersed nodes on the average and therefore less contention and MAC retransmissions. On the other hand, if there are many nodes within their transmission ranges at certain moments during the simulation, contention and the number of packet drops increase causing a need for more MAC retransmissions.

In general, the PI adaptive rate controller suffers fewer packet drops at the MAC layer than TCP. We observe that there is less contention in the wireless channel for the case when the PI rate controller is used than for the case of TCP New Reno. However, for both TCP New Reno and the PI rate controller, the higher the mobility, the more contention and therefore, more packets are dropped.

6.2. Comparison Under Different Ad-hoc Routing Protocols

Note that the design and implementation of our controller does not need any particular underlying Ad-hoc routing protocol. It is a separate algorithm that interacts among the transport and network layers. Therefore, it gives the flexibility for integration with any routing agent in wireless Ad-hoc networks. The remaining question is which routing protocol integration would produce a better performance.

In order to study and compare the effects of different Ad-hoc routing protocols, we have chosen the AODV protocol for comparison with DSR (used throughout our research work so far). As a reference, we firstly provide in Figure 6.5 a comparison between the performance of AODV using TCP and our adaptive PI rate controller before investigating the effect of a different routing protocol (the AODV) on the PI-rate controller.

Figure 6.5 compares the throughput evolution of TCP and the PI rate controller under different levels of mobility. The PI rate controller starts with a throughput of about 3100 Kbps

at 5m/s and drops to ~2200 Kbps at 20m/s. There is a dip in throughput (to ~2300 Kbps) at 10m/s

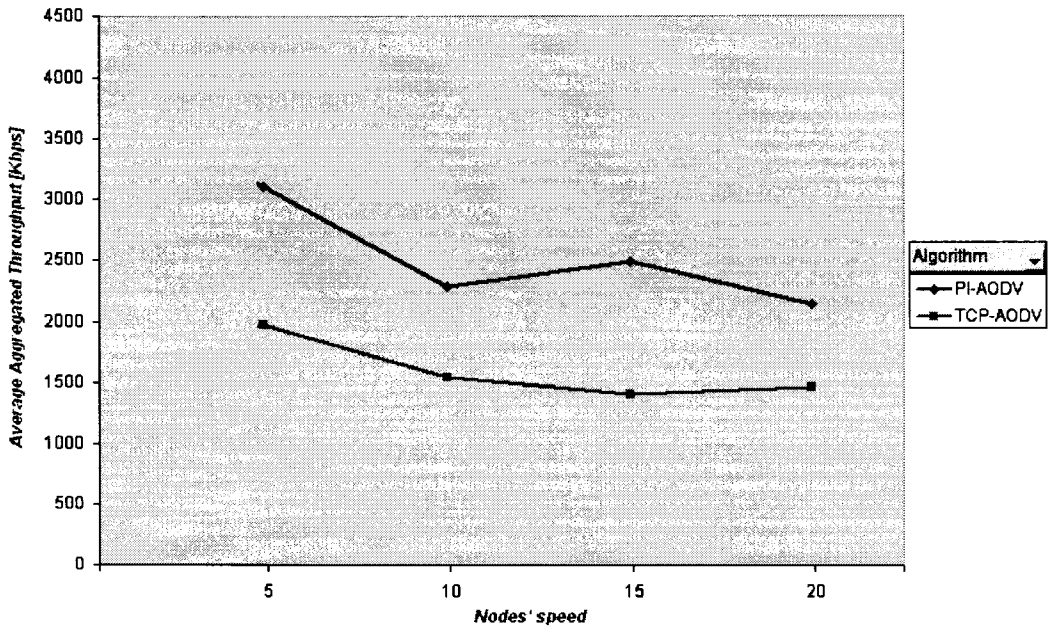

Figure 6.5: Average Aggregated Throughput Performance of TCP and the Adaptive PI Rate Controller using AODV and Under Different Node Speeds.

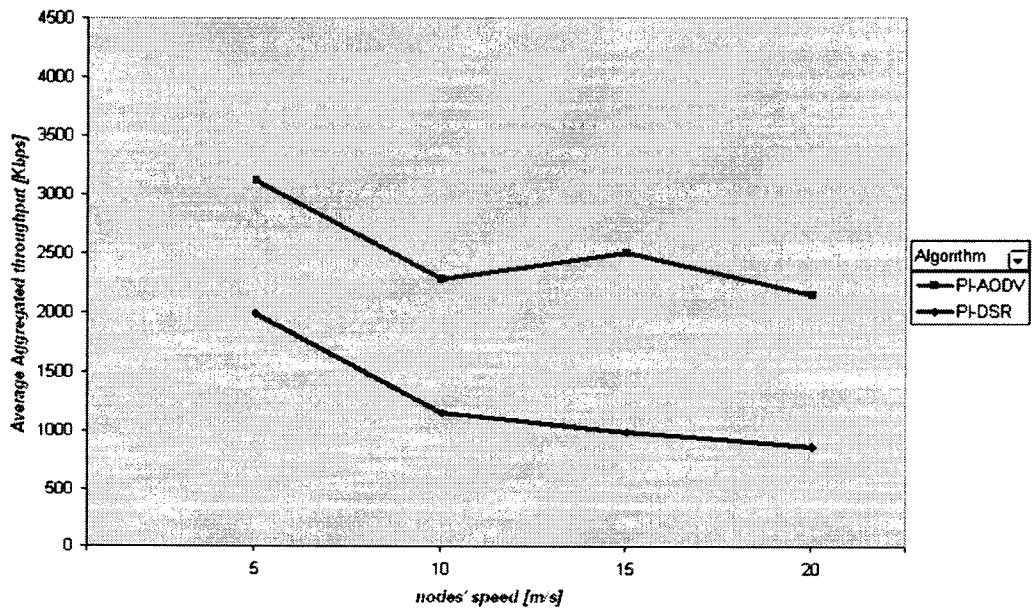

Figure 6.6: Average Aggregated throughput performance of the Adaptive PI Rate Controller under Different Routing Protocols.

probably due to the existence of a more dynamic topology causing more route failures. TCP in general shows a more constant behavior. It always decreases with node mobility but the throughput is always less than PI-rate using AODV.

6.2.1. Throughput

Figure 6.6 compares the performance of the PI rate controller using AODV and DSR respectively. DSR with the PI rate controller has a throughput of about 2000 Kbps at 5m/s, but decreases to almost half (~1000Kbps) at 20 m/s. The reduced performance is due to packet loss at high mobility. This can be explained because source routing is used by DSR. In high mobility scenarios, it is more difficult for DSR to be able to adjust to the dynamics of the topologies and routes, especially because the intermediate nodes do not update routing information.

AODV starts with a throughput of about 3100 Kbps at 5m/s and decreases to about 2100 Kbps at 20m/sec. The better performance can be explained by the fact that AODV has a quicker response time (in comparison to DSR) to link breakage in active routes. Despite the fact that AODV has periodic updates of routing information, AODV establishes route table entries dynamically at intermediate nodes and is able to find and update routes faster and more accurately at any node. Unfortunately, we also observe a dip in the aggregate throughput (decreasing up to ~2250Kbps) at 10 m/s. This can probably be due to the situation that even though the response time to link breakage is better than DSR (especially useful in high mobility), the network update at each intermediate node adds contention in the wireless channel which in some cases affects the effective throughput in the network.

So overall, the throughput performance of the adaptive PI rate controller is better for AODV than for DSR (with a difference of ~1000 Kbps). The performance difference between DSR and AODV becomes more obvious in Figure 6.7 and Figure 6.8 when the mobility becomes higher.

6.2.2. Packet Drops

Figure 6.7 compares the evolution of the average number of packet drops at the MAC layer for the PI rate controller using DSR and AODV under different levels of mobility. DSR shows a

sharp increase of packet drops (~1700 packets) at 10 m/s due to MAC retransmission. Then it decreases to ~1100 packets at 20 m/s. The increase is due to the increasing mobility and the inability of DSR to response to route updates (again because there is no participation of the intermediate nodes in the re-route establishment). The reduction in packet drops can be explained by the fact that whenever the routes are broken (1) there is a delay for new routes to be established, (2) a period in which the contending nodes reduce its transmission and therefore (3) the contention in the channel until a new route is established again. AODV slightly increase the number of packet drops from about 50 packets at 5 m/s to almost 500 packets at 20 m/s.

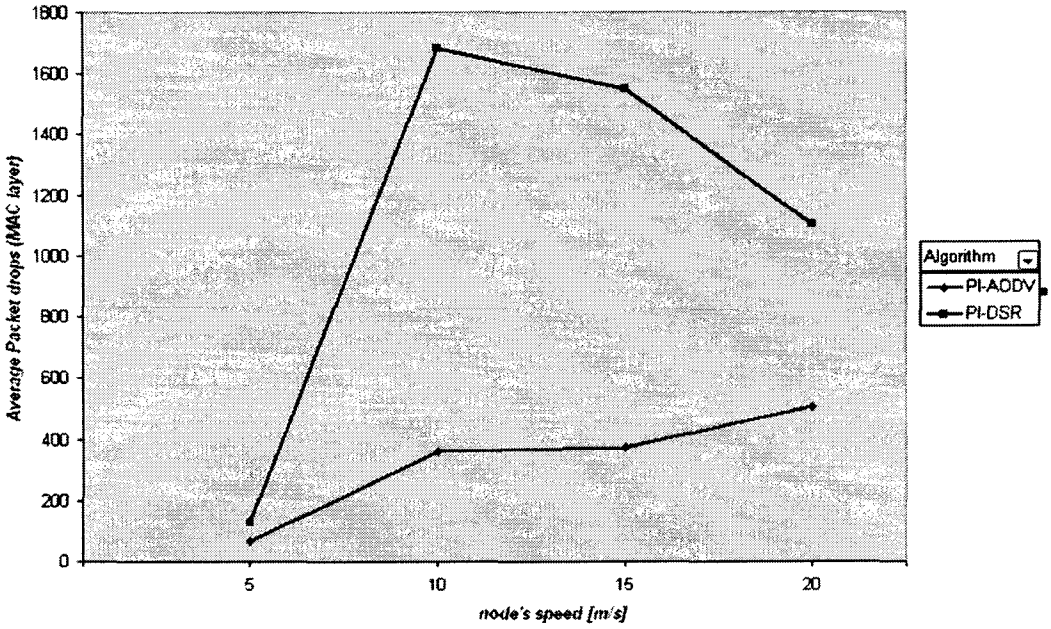

Figure 6.7: Average Packet Drops performance at the MAC layer of the Adaptive PI Rate Controller under Different Routing Protocols.

Both AODV and DSR start with almost no packet drops and both suffer an increase of them while the mobility speed increases. From Figure 6.7 one sees that DSR has more packet drops than AODV at the MAC layer at high mobility. This difference increases from less than 50 at 5m/s to ~1300 packet drops at 10m/s before decreasing to ~600 packet drops at 20m/s. This difference in the number of packet drops is due to the fact that DSR uses source routing. Apart from the source node, no intermediate nodes are involved in the route update Therefore, in high mobility scenarios, there are more route failures caused by the quick topology changes

in the network that source nodes are unable to effectively update in time. In these scenarios, packets are dropped, and retransmission and routes update are frequently needed. All these cause an increase in MAC contention which later will also result in packet drops. On the other hand, AODV keeps periodic updates of routing information as route table entries at each intermediate node. This is a disadvantage in simple static scenarios because more MAC contention results. However, it becomes an advantage in high mobility scenarios because it can result in less network transmissions (to update the routes whenever a route or a link fails) as all intermediate nodes are constantly updating the network condition, thus reducing the effect of contention in the MAC layer.

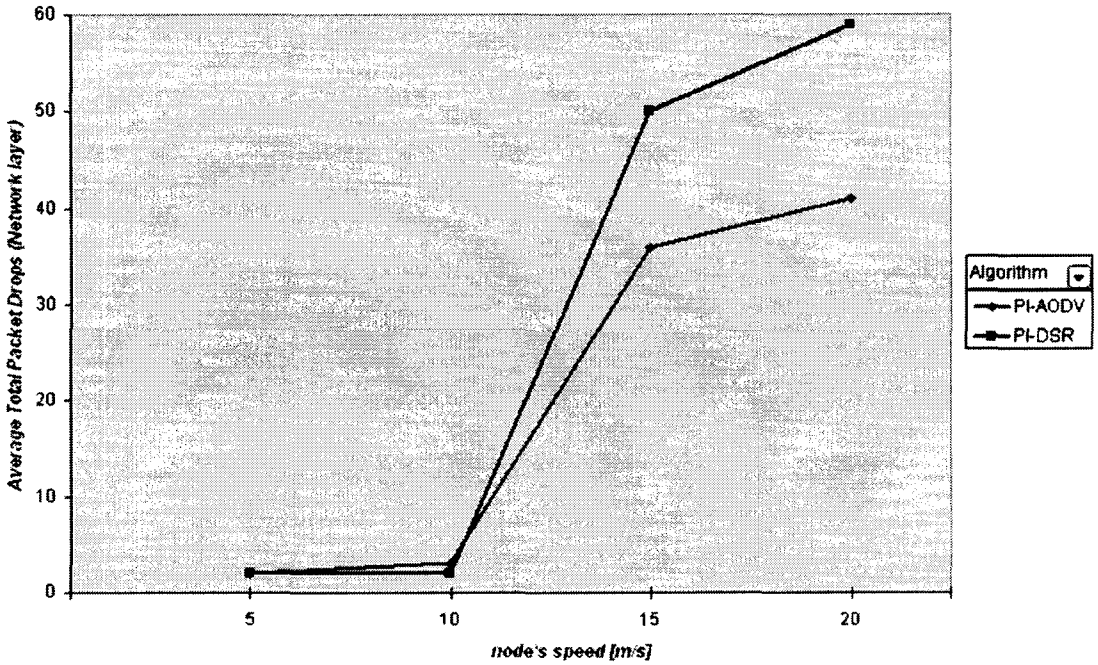

Figure 6.8: Average Packet Drops performance at the Network layer of the Adaptive PI Rate Controller under Different Routing Protocols.

Figure 6.8 compares the effect of using DSR and AODV on the average number of packet drops at the Network layer under different levels of mobility. The DSR packets drop is low (<5 packets) at low mobility of 5m/s and 10 m/s. This increases sharply beyond 10 m/s. The performance trend of AODV is about the same but with less packet drops. The difference is small but increases to ~20 packet drops when mobility increases. The higher packet drops in DSR can again be attributed to the shortcomings of DSR in responding to link failures as

discussed in previous sections. AODV in contrast is more effective in reacting to route failures and avoiding packet drops as each intermediate node participates in the network status update making it faster and more accurate.

6.3. Concluding Remarks

Our simulations show that the selection of a particular Ad-hoc routing protocol is an important factor to consider as it influences the overall network throughput performance. This influence arises from the mobility of the networks nodes (both as sources and routers), giving routing a very important role to play when considering congestion control algorithms. As seen, the difference in throughput performance between two different Ad-hoc routing protocols is mainly due to the characteristics of each Ad-hoc routing protocol.

On the other hand, one also observes that the throughput performance of the adaptive PI rate controller is better for AODV than for DSR (with a difference of ~1000 Kbps). This is in addition to the reference of Figure 6.5 where PI is shown to be better than TCP New Reno when using AODV. So we can comfortably say that the PI rate control algorithm always has a positive effect on the throughput performance which is independent of the Ad-hoc routing protocol used. This is due to the fact that the adaptive PI rate algorithm design does not relies on a particular routing protocol, making it easily deployable for any Ad-hoc routing protocol.

Chapter 7
Conclusions

This thesis has concentrated mainly on the implementation and evaluation of a PI rate control scheme in wireless Ad-hoc networks. The PI rate controller was proposed in order to regulate the traffic in the network. This controller is based on the classical feedback control theory and requires the interaction between the network layer and the transport layer and from both intermediate nodes and end-nodes.

We have provided an extension and a design method to implement the PI Rate controller at each node which can act as a source node, a destination node as well as an intermediate node. This is particularly important because all intermediate nodes can perform as routers, and some of them could act as bottleneck routers anytime. By controlling the source node transmission rate based on the intermediate node's information along the path, network overload is reduced which helps improving the overall system performance of TCP.

The adaptive PI rate controller is TCP compatible in the sense that the basic TCP window mechanism is used. It also leads to improvements in end-to-end TCP throughput. This scheme offers a simple way to provide flow congestion control through a mechanism that controls the transmission rate of source nodes. By controlling the transmission rate this scheme offers a way to improve the throughput performance of TCP. Moreover, the self-tuning capabilities of the PI rate control algorithm offer the ability to react to the fluctuations of the network.

In order to evaluate the performance of this control scheme and its behavior in wireless Ad-hoc networks simulation models were developed and verified. A large number of simulations in both static and mobile scenarios were conducted. First we verified the PI rate controller behavior in a simple static scenario with one intermediate node bottleneck. We compared its behavior to TCP New Reno in terms of: average throughput, instantaneous queue size, window size, and average queue size. We observed that the PI rate controller outperformed TCP New Reno. The PI Rate controller behavior was also investigated in terms of throughput improvement under several different static and mobile scenarios. In all cases we found that the

PI Rate control scheme outperforms to that of TCP New Reno. Traffic flows are more evenly routed throughout the network because there is no "core" router in Ad-hoc networks resulting in lower rate and higher throughput. Even with the additional processing overhead, the PI-rate controller is feasible for Ad-hoc networks as it is simple to implement without requiring complex modifications. It requires simple cross layer communication as it is between adjacent layers. It does not require huge modifications to many communication layers and its protocols, which ease its deployability and inter-operability with other networks. Because of its simplicity, the PI rate control algorithm offers the possibility to be extended by means of the use of feedback information from other layers.

We also compare the adaptive PI rate controller performance when using different ad-hoc routing protocols AODV and DSR to show the effect of the interaction of routing in the network layer with other communication layers such as the MAC and the transport layers. The PI rate controller gives the flexibility for integration with any routing agent in wireless Ad-hoc networks because it is a separate algorithm that interacts among the transport and network layers and its design and implementation does not need any particular underlying Ad-hoc routing protocol.

Our simulations show that the adaptive PI rate controller had a positive influence on the network throughput no matter the routing protocol selected. However, the selection of a particular Ad-hoc routing protocol is an important factor to consider as it influences the network performance. The final difference in throughput performance between two different Ad-hoc routing protocols is mainly due to the characteristics of each Ad-hoc routing protocol.

7.1. Future Work

For future work, one can study the interaction of Ad-hoc routing protocols with the adaptive PI rate controller. There are also issues related to the implementation of the PI Rate control scheme using feedback information (such as application frames dropping, non-cooperative nodes, routing failure messages, MAC layer retransmissions, physical layer energy consumption) from cross layers. All these may be important for a unified solution that would take into account the interaction among other different communication layers.

References

[AfMa96] Afek Y., Mansour Y., Ostfeld Z., "Phantom: A Simple and Effective Flow Control Scheme", in *Proceedings of ACM SIGCOMM Computer Networks Journal,* Vol. 32, No. 3, 1996, pp. 277-305. http://citeseer.ist.psu.edu/afek96phantom.html

[AlJi03] Altman E. and Jimenez T., "Novel Delayed ACK Techniques for Improving TCP Performance in Multi-hop Wireless Networks", in *Proceedings Personal Wireless Communications (PWC 2003)*, Venice, Italy, Sep. 2003, pp. 237-253.

[AnPa03] Anastasi G. and Passarella A., "Towards a Novel Transport Protocol for Ad-hoc Networks," in *Proceedings of the Eighth International Conference on Personal Wireless Communications (PWC 2003),* September 2003, pp. 23-25.

[AnPa03] Anantharaman V., Park S.-J., Sundaresan K., and Sivakumar R., "TCP Performance over Mobile Ad-hoc Networks: A Quantitative Study", in *Wireless Communications and Mobile Computing Journal (WCMC 2004), Special Issue on Performance Evaluation of Wireless Networks,* Vol. 4, 2004, pp 203-222.

[BaKr97] Bakshi B., Krishna P., Vaidya N. H., and Pradhan D. K., "Improving Performance of TCP over Wireless Networks," in *Proceedings of 17th International Conference on Distributed Computing Systems (ICDCS),* Baltimore, MD, May 1997, pp. 365-373.

[BoDa03] Boukerche A., Das S.K., "Congestion control performance of R-DSDV protocol in multi-hop wireless ad hoc networks", in *Wireless Networks.,* Vol. 9, No. 3., May 2003, pp. 261-270.

[BoHe03] Bohacek S., Hespanha J., Lee J., Lim C., and Obraczka K., "TCP-PR: TCP for Persistent Packet Reordering," in *Proceedings of the IEEE 23rd International Conference on Distributed Computing Systems (ICDS 2003),* May 2003, pp. 222-231.

[CaBo02] Camp T., Boleng J., and Davies V., "A survey of mobility models for Ad-hoc network research", in *Proceedings and Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications (WCMC '02)*, 2002, Vol. 2, No. 5, pp. 483–502.

[Chia04] Chiang M., "To Layer or Not to Layer: Balancing Transport and Physical Layers in Wireless Multi-hop Networks: jointly optimal congestion control and power control", in *Proceedings IEEE Infocom04,* March 2004 and in *IEEE Journal on Selected Areas in Communications,* Vol. 23, No. 1, January 2005, pp. 104-116.

[ChNa04] Chen K., Nahrstedt K., Vaidya N., "The Utility of Explicit Rate-Based Flow Control in Mobile Ad-hoc Networks", in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2004),* Atlanta, Georgia, Vol. 3, March, 2004, pp. 1921-1926.

[ChNa05] Chen K., Nahrstedt K., "Limitations of Equation-based Congestion Control in Mobile Ad-hoc Networks", in *Proceedings International Workshop on Wireless Ad-hoc Networking (WWAN 2005)* in conjunction with the *24th International Conference on Distributed Computing Systems (ICDCS-2005)*, March, 2005, pp. 756-761.

[ChRa98] Chandran K., Raghunathan S., Venkatesan S., and Prakash R., "A Feedback Based Scheme for Improving TCP Performance in Ad-hoc Wireless Networks," in *Proceedings of International Conference on Distributed Computing Systems,* Amsterdam, May 1998, pp. 472–479.

[ChRa01] Chandran K., Raghunathan S., Venkatesam S. and Prakash R., "A Feedback-based Scheme for Improving TCP Performance in Ad-hoc Wireless Networks", in *Proceedings IEEE personal Communication Magazine*, Vol. 8, No. 1, February 2001, pp. 34-39.

[ChXu03] Chen K., Xue Y., and Nahrstedt K., "On Setting TCP's Congestion Window Limit in Mobile Ad-hoc Networks," in *Proceedings IEEE International Conference on Communications (ICC 2003),* Alaska, pp. 1080-1084. http://citeseer.ist.psu.edu/chen03setting.html

[DyBo01] Dyer T. and Boppana R. V., "A Comparison of TCP Performance Over Three Routing Protocols for Mobile Ad-hoc Networks", in *Proceedings of the 2nd ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc 2001),* Long Beach, CA, USA, Oct 2001, pp. 56-66.

[FlHe04] Floyd S., Henderson T., and Gurtov A., "The New Reno Modification to TCP's Fast Recovery Algorithm", *RFC 3782* (Proposed Standard), Apr. 2004.

[FuLu05] Fu Z., Luo H., Zerfos P., Lu S., Zhang L., and Gerla M., "The Impact of Multi-hop Wireless Channel on TCP Performance," in *IEEE Transactions on Mobile Computing*, Vol. 4, No. 2, March-April 2005, pp. 209-221.

[FuZe03] Fu Z., Zerfos P., Luo H., Lu S., Zhang L. and Gerla M., "The Impact of Multi-hop Wireless Channel on TCP Throughput and Loss", in *Proceedings IEEE Infocom03 22nd Annual Joint Conference of the IEEE Computer and Communications Societies,* San Francisco, CA, USA, Vol. 3, Apr. 2003, pp. 1744-1753.

[GeTa99] Gerla M., Tang K., and Bagrodia R., "TCP Performance in Wireless Multi-hop Networks," in *Proceedings of 2nd IEEE Workshop on Mobile Computer Systems and Applications IEEE WMCSA*, New Orleans, Feb 1999, p. 41.

[GuYa02] Guo S., Yang O., W.W., "Performance of Backup Source Routing (BSR) in Mobile Ad-hoc Networks", in *Proceedings of IEEE Wireless Networking Conference 2002,* Orlando, FL, USA, Vol. 1, pp. 440-444.

[HaFl03] Hanley, M., Floyd, S., Padhye, J., Widmer, J.: "TCP Friendly Rate Control (TFRC): Protocol Specification", *RFC 3448,* Proposed Standard, January 2003.

[HeBa05] Heimlicher S., Baumann R., May M. and Plattner B., "SAFT—Store and Forward Transport in Mobile Ad-hoc Networks," *Communication Systems Group, ETH Zurich, Tech. Rep. 239,* July 2005.

[HeBa06] Heimlicher S., Baumann R., May M. and Plattner B., "SaFT: Reliable Transport in Mobile Networks", *MASS '06,* Vancouver, CA October 9-12 2006.

[HoVa99] Holland G. and Vaidya N. H., "Analysis of TCP Performance over Mobile Ad-hoc Networks", in Proceedings of IEEE/ACM Mobicom99, Seattle, WA, August 1999, pp. 219–230.

[HoVa99] Holland G. and Vaidya N. H., "Impact of Routing and Link Layers on TCP Performance in Mobile Ad-hoc Networks," in *Proceedings of IEEE Wireless Communications and Networking Conference WCNC 1999,* New Orleans, LA, USA, Vol. 3, September 1999, pp. 1323-1327.

[HoXi05] Hongqiang Z., Xiang Ch., Yuguang F., "Rate-based Transport Control for Mobile Ad-hoc Networks", in *Wireless Communications and Networking Conference, IEEE 2005,* Vol. 4, No. 13-17, March 2005, pp. 2264 - 2269.

[HoYa06] Hong, Y., Yang, O.W.W., "Design of an Adaptive PI Rate Controller for Streaming Media Traffic Based on Gain and Phase Margins", in *IEE Communications Proceedings,* Vol. 153, No. 1-2, February 2006, pp. 5 - 14.

[IETF04] Johnson B., Maltz A., Hu Y., "The Dynamic Source Routing Protocol for Mobile Ad-hoc Networks (DSR)", *Internet Draft,* July 2004. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt (last accessed September 2006).

[IETF99] Floyd S., Henderson T., "The New Reno Modification to TCP's Fast Recovery Algorithm", *Internet Draft,* April 1999. http://www.ietf.org/rfc/rfc2582.txt (last accessed September 2006).

[JaCh84] Jain R., Chiu D., Hawe W., "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems", *DEC Technical Report DEC-TR-301*, 1984.

[JoMa96] Johnson D.B., and Maltz D., "Dynamic Source Routing in Ad-Hoc Wireless Networks," *Chapter 5 in the book Mobile Computing*, edited by Tomas Imielinski and Hank Korth, (Kluwer Academic Publishers), 1996, chapter 5, pp. 153-181.

[KaHa02] Katabi D., Handley M., and Rohrs C., "Congestion control for High Bandwidth-delay Product Networks", in *Proceedings ACM SIGCOMM of the 2002 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications,* August 2002, pp. 89-102.

[KaKa00] Karandikar, S., Kalyanaraman, S., Bagal, P., and Packer, B., "TCP Rate Control", in *Proceedings ACM Computer Communications*, Rev., 2000, Vol. 30, No. 1, pp. 45–58.

[KaVa03] Karthikeyan S., Vaidyanathan A., "ATP: A Reliable Transport Protocol for Ad-hoc Networks", in *Proceedings of the ACM Symposium on Mobile Ad-hoc Networking and Computing Mobihoc '03*, June 1-3, 2003, pp. 64-75.

[KhSh04] Kherani A. and Shorey R., "Throughput Analysis of TCP in Multi-hop Wireless Networks with IEEE 802.11 MAC", in *Proceedings of IEEE*

Wireless Communications and Networking Conference (IWCNC'04), Atlanta, GA, Vol. 1, March 2004, pp. 237-242.

[LeMi06] Lee, U., Midkiff, S.F., "Channel distribution fairness in multi-channel wireless ad-hoc networks using a channel distribution index", in *25th IEEE International Performance, Computing and Communications Conference (IPCCC 2006),* Vol. 10, No. 12, April 2006, pp. 111-118 .

[LiSi01] Liu J. and Singh S., "ATCP: TCP for Mobile Ad-hoc Networks," in *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 7, July 2001, pp. 1300-1315.

[LiSt05] Lijun Ch., Steven H. and Doyle C., "Joint Congestion Control and Media Access Control Design for Wireless Ad-hoc Networks", in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, Miami, FL, March 2005, pp. 2212-2222

[LiXu03] Lim H., Xu K., and Gerla M., "TCP Performance over Multi-path Routing in Mobile Ad-hoc Networks," in *Proceedings of the IEEE International Conference on Communications (ICC 2003),* Anchorage, Alaska, Vol. 2, May 2003, pp. 1064-1068.

[LoWy05] Low, S., and Wydrowski, B., "Understanding XCP: Equilibrium and Fairness", in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2005)*, Vol. 2, March 2005, pp. 1025-1036.

[MoSi00] Monks J., Sinha P., and Bharghavan V., "Limitations of TCP-ELFN for Ad-hoc Networks," in *Proceedings of The 7th International Workshop on Mobile and Multimedia Communication (MOMUC 2000),* Marina del Rey, CA, Oct. 2000, http://timely.crhc.uiuc.edu/publications.html (last accessed October 2006).

[NaHe05] Nahm K., Helmy A., Kuo J., "TCP over Multi-hop 802.11 Networks: Issues and Performance Enhancement", in *Proceedings of the 6th ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc '05),* May 2005, pp. 277-287.

[PeBh94] Perkins C., Bhagwat P., "Highly dynamic destination-sequenced distance vector (DSDV) for mobile computers," in *ACM SIGCOMM '94*, Aug. 1994, pp. 234–244.

[PeRo99] Perkins Ch., and Royer E., "Ad-hoc On-Demand Distance Vector (AODV) Routing," *Proc. 2nd IEEE Wksp. Mobile Comp. Sys. And Apps.*, New Orleans, LA , Feb. 1999, pp. 90-100.

[Qual05] Qualnet documentation, available at: http://www.scalable-networks.com/ (last accessed August 2006).

[Rapa95] Rappaport, T. S., "Wireless Communications: Principles & Practice," *Prentice Hall*, 1995.

[RaKl05] ElRakabawy S., Klemm A., Lindemann Ch., "Transport 2: TCP with Adaptive Pacing for Multi-hop Wireless Networks", *in Proceedings of the 6th ACM*

International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc '05), May 2005, pp. 288-299.

[RoTo99] Royer E., Toh Ch., "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, April 1999.

[ShRa03] Shakkottai S., Rappaport T., and Karlsson P., "Cross layer design for wireless networks", IEEE Communications Magazine. April 2003.

[Stev97] Stevens W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001* (Proposed Standard), Jan. 1997.

[SuAn03] Sundaresan K., Anantharaman V., Hsieh H., and Sivakumar R., "ATP: A Reliable Transport Protocol for Ad-hoc Networks", *in Proceedings of the 4th ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc '03),* Annapolis, MA, 2003, pp. 64-75.

[TaMa03] Takai M., Martin J., Bagrodia R., "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *IEEE Journal On Selected Areas In Communications,* Vol. 21, No. 3, April, 2003.

[WaZh02] Wang F. and Zhang Y., "Improving TCP Performance over Mobile Ad-hoc Networks with Out-of-Order Detection and Response", in *Proceedings of the 3rd ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc '02),* Lausanne, Switzerland, 2002, pp. 217-225.

[XuGe03] Xu K., Gerla M., Qi L., and Shu Y., "Enhancing TCP Fairness in Ad-hoc Wireless Networks using Neighborhood RED", in *Proceedings of 9th Annual International Conference on Mobile Computing and Networking ACM (Mobicom '03),* San Diego, CA, USA, September 2003, pp. 16-28.

[YiSh04] Yi Y. and Shakkottai S., "Hop-by-hop Congestion Control over a Wireless Multi-hop Network", in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom '04),* Vol. 4, March 2004, pp. 2548-2558.

[Yu04] Yu X., "Improving TCP Performance over Mobile Ad-hoc Networks by Exploiting Cross-Layer Information Awareness", in *Proceedings of the 10th ACM Annual International Conference on Mobile Computing and Networking (Mobicom '04),* Philadelphia, PA, USA, 2004, pp. 231-244.

Appendix A
TCP Reno

TCP Reno is a TCP version that includes the following mechanisms: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery. TCP Reno reacts to heavy congestion by detecting a loss through timeouts and not via duplicate acknowledgement reception [Stev97]. Whenever the sender is still receiving ACKs from the receiver, the sender should not go back into the slow start phase. The sender can keep on sending packets since the flow still exists. But, the sender should reduce the number of packets sent in order to reduce the resources utilization.

TCP Reno introduces a mechanism called Fast Recovery [Stev97] which should be activated after a Fast Retransmit. Fast Retransmit will cause the sender to retransmit packets after receiving "n" duplicated ACKs. This is implemented by initially halving the size of the congestion window size and storing this value in the ssthresh parameter. Then the congestion window value is set to the ssthresh value plus three times the Maximum Segment Size (three refers to the number of duplicated ACKs required to trigger a fast retransmit). This applies up to half the number of segments when the Fast Recovery mechanisms will take place. The size of the congestion window at this time is W but, the shrinking means that all the packets that are within the window have actually already be sent out. Therefore, unless the window is forced to include new packets, there is no way new packets could be sent out.

The only way to shift the window is to inflate the size of the congestion window size upon the receipt of each duplicated ACK by one segment. After W/2 duplicated ACKs have been received, the window will be ready to include new packets. After all the duplicated ACKs have been used up, the next ACK should be that from the retransmission and thus it should ACK all packets. Upon the receipt of this normal ACK, the sender should exit the Fast Recovery phase and set its congestion window size to the value of the ssthresh parameter (W/2). The W value is still the size of the congestion window size before the Fast Recovery mechanism started. Due to the receipt of duplicated ACKs, all of the packets except for the last one have already been transmitted. Thus, only one packet needs to be sent. With the Fast Recovery mechanism, TCP Reno is forced to send fewer packets to reduce the network utilization until it knows it could

send more. In such way, only when a packet is timed out a slow start is used. This is implemented by setting the ssthresh parameter to half the current congestion window size value and then setting it to 1 segment. In such way, the tcp connection will slow start up to the ssthresh value and then will execute the congestion avoidance [Stev97].

Appendix B
TCP New Reno [FlHe04].

TCP New Reno modifies the Fast Retransmit and Fast Recovery mechanisms of TCP [FlHe04]. These modifications are intended to fix the following Reno problems and are completely implemented in the sender side:

- Fast Retransmit mechanism assumes that only one segment is lost, whenever there is a bursty transmission of packets and thus bursty packet loss, resulting in loss of ACK clocking and timeouts if more than one segment is lost.
- The problem above usually happens when invoking Fast Retransmit and Fast Recovery. Because TCP Reno invokes them several times, it leads to multiplicative decreases of the congestion window size and the ssthresh parameters; affecting throughput of the connection.
- Another problem is ACK starvation which happens due to the ambiguity of duplicate ACKs and due to the dynamics of the congestion window size. The sender reduces the congestion window size whenever it enters into Fast Retransmit; it receives duplicate ACKs that inflate the congestion window size so that it sends new packets until it fills its sending window. It receives a non duplicate ACK and exits Fast Recovery. However, because of past multiple losses, this ACK will be followed by 3 duplicate ACKs showing that another segment was lost. Therefore it will enter again into the Fast Retransmit phase after reducing the values of the congestion window size and the ssthresh.
- Successive Fast Retransmit events cause the left edge of the sending window to advance and the amount of data in flight (sent but not yet acknowledged) to become eventually more than the congestion window. Because there are no more ACKs to be received then the sender stops and recovers from this dead-lock only through a timeout which causes a slow start.

TCP New Reno is the same as TCP Reno but with modifications during the Fast Recovery phase. It makes use of partial ACKs in such way that whenever there are multiple packet drops, the ACKs for the retransmitted packet will acknowledge some, but not all the segments send before the Fast Retransmit. In TCP Reno, the first partial ACK will make the sender to go out of

the fast recovery phase resulting in the stalling of the TCP connection because of the timeouts requirement whenever there are multiple losses in a window. In New Reno, a partial ACK will be a lost packet indication, and then the sender will retransmit the first unacknowledged packet. Unlike TCP Reno, partial ACKs do not take TCP New Reno out of the Fast Recovery phase. It will just retransmit one packet per round trip time until all the lost packets are retransmitted. It avoids the need for multiple fast retransmits within a single window of data. The drawback is that it may take many round trip times to recover from a packet loss which influence the fact that there must be enough new data to keep the ACK clock running. TCP Reno exits the Fast Recovery phase whenever the first ACK advancing the window arrives at the sender. TCP New Reno, only exits fast-recovery after all segments in the last window have been acknowledged [FlHe04].

Appendix C
Adaptive PI Rate Controller

C.1. Traffic Control Mechanism [HoYa06]

The PI rate controller is located in every intermediate node and can calculate a desirable source transmission rate. This transmission rate is based on the instantaneous queue length of the buffer in the intermediate node. The intermediate node advertises the transmission rate as a window size to the source through the IP and ACK packets so that the source can regulate its current transmission rate $q_i(t)$.

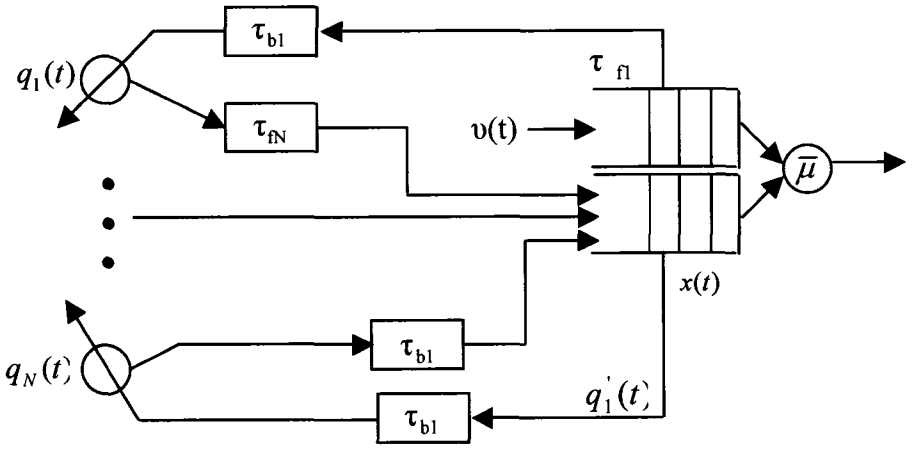

Figure C.1: Intermediate Node

Figure C.1 depicts an intermediate node with N best-effort traffic flows (each with a controlled source) and one guaranteed service traffic flow (uncontrolled) with input rate $\upsilon(t)$. The intermediate node has a finite buffer space K to store the incoming packets and an output link to serve them at a constant data rate μ. The buffer in the intermediate node is organized into a logical queue whose service is assumed first-input first-output (FIFO).

There are two kinds of time delays, both considering the propagation, queuing and processing delays. The sum of these two delays $\tau_i = \tau_{fi} + \tau_{bi}$ defines the varying round trip time τ_i of the controlled source node i: (1) the forward time delay τ_{fi} from the controlled source

node i to the destination node; (2) the feedback time delay τ_{bi} from both the intermediate node to the destination node and from the destination node to the controlled source node i.

From the model of the intermediate node in Figure C.1, the change of the queue size in the buffer of the intermediate node is given by the sum of the input rates of both the multiple best-effort traffic and the guaranteed service traffic minus the service rate. Thus the instantaneous queue size can be written as:

$$\dot{x}(t) = \sum_{i=1}^{N} q_i(t - \tau_{fi}) + \upsilon(t) - \overline{\mu}$$

The term $q_i'(t-\tau_{fi})$ represents the rate at which source node i sends packets into the intermediate node. The term $v(t)$ - μ is the available bandwidth in the intermediate node to serve the TCP data traffic. Based on the instantaneous queue length $x(t)$ of the buffer, the advertised transmission rate $q'_i(t)$ for the controlled source node i can be calculated by the following:

$$q_i'(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau$$

$$= K_P(x_0 - x(t)) + K_I \int_0^t (x_0 - x(\tau))d\tau$$

where $q'_i(t)$ represents the advertised source transmission rate, $e(t)$ the queue deviation, and K_P and K_I are the proportional and the integral gain of the PI rate controller, respectively. The advertised source node transmission rate is calculated based on the instantaneous queue length of the buffer. The advertised transmission rate becomes the source rate after a feedback time delay of τ_{bi}.

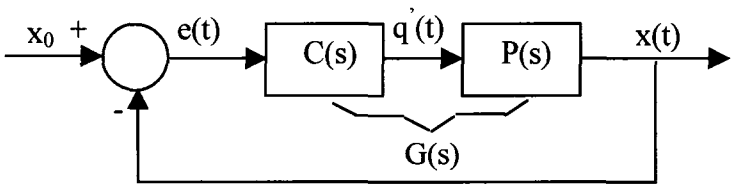

Figure C.2: Control System

The PI rate controller design is based on its gain and phase margins. Figure C.2 shows its queue management control system. $P(s)$ represents the intermediate node with N controlled

source nodes and *C(s)* the PI rate controller. It has been determined [HoYa06] that a gain margin in the range of $2<A_m<4$, and a phase margin in the range of $30^\circ < \varphi_m < 60^\circ$ for the controller transfer function must be used to guarantee system stability so that our adaptive PI rate controller can properly regulate the source node transmission rate. In addition, the number of controlled source nodes must be estimated and the round trip time must be extracted from the header of the IP packet. The values of the proportional and integral gain must be calculated.

Whenever the number of active controlled source nodes changes or the average round trip time becomes longer than the value to which it was designed, the values of the proportional and integral gains will need to be recalculated. Finally, the advertised window size can be obtained from the advertised source transmission rate via the following calculation [KaKa00], [KaHa02], and [LoWy05]: $win_i'(t) = \tau_i \cdot q_i'(t)$

The advertised window size is restricted between 1 and the maximum window size allowed for the source nodes.

Appendix D
Qualnet Tools and Models [Qual05].

Qualnet provides a comprehensive environment for designing protocols, creating and animating experiments, and analyzing the results of those experiments. It is composed of a set of tools with components for network modeling and simulation. The modular design of Qualnet allows for the modification of protocols at different communication layers, cross layer communication and built-in measurements definition at each of the communication layers. Qualnet six main tools and their respective functionalities can be summarized as the following:

1. Scenario Designer: It includes a graph interface that can be used by the user to conveniently create and design scenarios, to include devices, to configure interfaces and network components, and to include the main settings for various different applications.
2. Animator: It is used to execute and animate experiments created in the scenario designer. It can show the details of each layer when simulation is running.
3. Analyzer: A graphical statistical analyzing tool for displaying simulation results in graphs.
4. Protocol Designer: This is a graphical protocol designing tool to design new network protocols and to provide a simplified mechanism for incorporating protocols into the simulator. Qualnet offers the ability to either modify its existing sample protocol models or to create new ones. The sample models mainly offer client-server applications, application-routing, transport, network-routing, network-queuing and MAC layer protocols. Some of these are intensively used in this research for the PI rate control algorithm implementation.
5. Packet Tracer: This is a graphical packet tracing tool that allows analyzing the content of the packets generated during the simulation. This tracer is a standalone tool that reads packet data collected from the Qualnet Simulator. The purpose is to allow one to follow the life cycle of a packet as it proceeds up and down a protocol stack, and across a network.
6. Simulator: It allows simulating large, heterogeneous networks and the distributed applications that execute on those networks. Some of the features it provides include a set of wired and wireless network protocol and device models, useful for simulating diverse types

of networks. It is designed as a parallel simulator that allows the execution of simulations through a command line developer, making it faster as processors are added.

The simulation in Qualnet works as a collection of network nodes, each with its own protocol stack parameters and statistics. It mainly makes use of messages as the unit defining the interaction between protocols and between nodes. There could be two types of messages: (1) messages for packets sent and used for communication between nodes, and (2) messages sent according to set timers which allow protocols to schedule events.

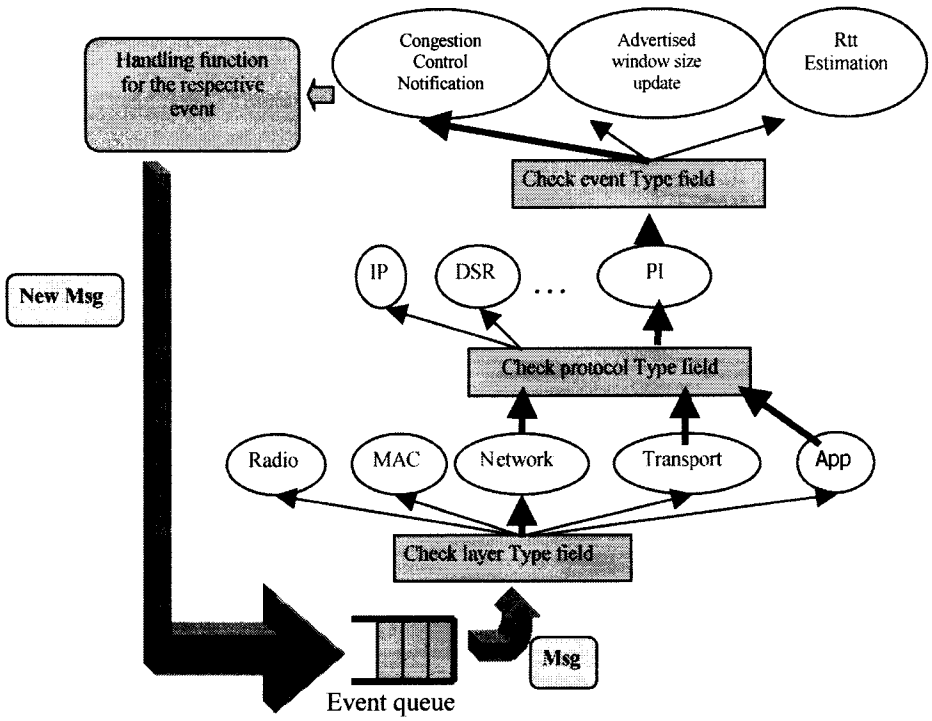

Figure D.1: Qualnet's Message Processing

Figure D1 shows the message processing in Qualnet. It shows that whenever a new message is received, an event counter is initialized and sent to an event queue where the message will be scheduled according to the "layer type" field value of the respective message. Depending on this value, specific functions will be executed according to the type of protocol the message is dealing with. Moreover, the protocol particular tasks will then be conducted according to the type of event established in the message. After the functions which the message was sent for are

executed, a function that handles the end of each event will flush the table events accordingly. This process is conducted in every communication layer and according to the protocols used in each of them.

Qualnet uses a layered architecture similar to that of the TCP/IP protocol stack. Within that architecture, data moves between adjacent layers. Adjacent layers in the protocol stack communicate by means of programming and via APIs. Generally, layer communication occurs only between adjacent layers.

The protocols implementation in Qualnet uses a stack model. Qualnet is an event-driven simulator where neighboring layers communicate through APIs. The interface between layers is event-based. To pass data to, or request a service from an adjacent layer, the system schedules an event at that layer. The layer code, therefore, is implemented as an event handler, that receives an event data structure. This data structure is called a message that contains the type of event and the associated data. The event handler then parses the data and handles it appropriately, possibly scheduling further events at the next layer.

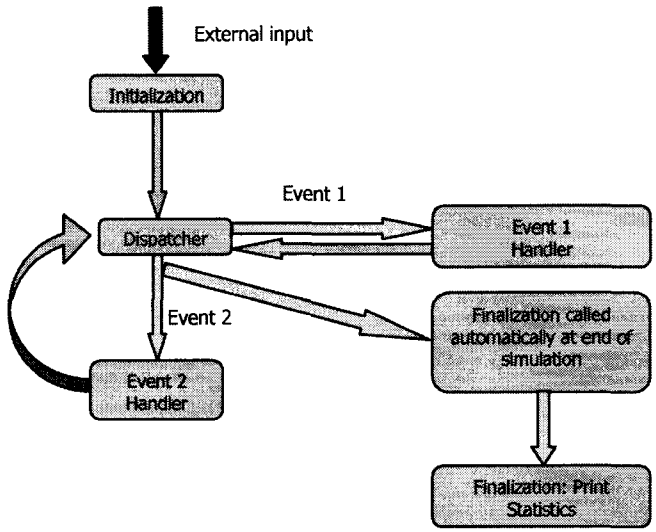

Figure D.2: Qualnet's Event Handler Process

The event handler process is shown in Figure D.2. At the lowest level, the process domain is structured as a Finite State Machine (FSM). The process in Figure D.2 occurs in the following order: Every protocol begins with an Initialization function. The initialization

function reads external input to configure the state of the protocol. The protocol then passes control to an event dispatcher. Whenever an event comes in to any communication layer (e.g. a packet arrival, a congestion notification, an RTT estimation update, a window size update, an end of session), Qualnet Simulator determines to which protocol it should be directed (e.g. queuing protocol, routing protocol (DSR), transport protocol (TCP New Reno or PI), application protocol (FTP)), and calls the event dispatcher for that protocol. The event dispatcher determines what type of event it is (e.g. congestion control notification, advertised window size update, RTT estimation, etc.), and calls an appropriate event handler (e.g. corresponding module, function or code) to process it. At the end of the simulation, a finalization function is called for every protocol (at each node), to print out the collected statistics. This process must be implemented by every protocol to interface to the Simulator. After building the protocol model functionality in order to incorporate the new protocol into Qualnet, it is needed to convert the Finite State Diagram into header and body files. Each state and the relationship between states can be programmed with C or C++ language. Therefore the source code for each model must be developed. During the protocol development process, Qualnet protocol APIs are also used. These are of two types: (1) General APIs which contains functions and routines available to all the protocol layers (sending message, setting timers, etc.) and (2) Layer APIs which contain functions and routines available to a specific layer only.