İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

IMPROVING TCP PERFORMANCE IN IEEE 802.11 BASED WIRELESS NETWORKS WITH ADAPTIVE RTS RETRANSMISSION SCHEME

M.Sc. Thesis by

Koray ATALAY, B.Sc.

Department : Computer Engineering

Programme: Computer Engineering

Supervisor : Assoc. Prof. Dr. Sema F. OKTUĞ

JUNE 2006

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

IMPROVING TCP PERFORMANCE IN IEEE 802.11 BASED WIRELESS NETWORKS WITH ADAPTIVE RTS RETRANSMISSION SCHEME

M.Sc. Thesis by
Koray ATALAY, B.Sc.
(504031519)

Date of submission : 8 May 2006

Date of defence examination: 19 June 2006

Supervisor (Chairman): Assoc. Prof. Dr. Sema F. OKTUĞ

Members of the Examining Committee Assist. Prof. Dr. Feza BUZLUCA (İTÜ.)

Assist. Prof. Dr. Tuna TUĞCU (BÜ.)

JUNE 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

IEEE 802.11 TABANLI KABLOSUZ AĞLARDA ADAPTİF RTS TEKRARI İLE TCP PERFORMANSININ İYİLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ
Müh. Koray ATALAY

(504031519)

Tezin Enstitüye Verildiği Tarih : 8 Mayıs 2006
Tezin Savunulduğu Tarih : 19 Haziran 2006

Tez Danışmanı :	**Doç.Dr. Sema F. OKTUĞ**
Diğer Jüri Üyeleri	**Yrd.Doç.Dr. Feza BUZLUCA (İTÜ.)**
	Yrd.Doç.Dr. Tuna TUĞCU (BÜ.)

HAZİRAN 2006

FOREWORD

At first place, I would like to thank my advisor Assoc. Prof. Sema F. Oktuğ, for her advices and understandings during this work. I also would like to thank my colleagues Murat Ege, Eda Ormancı, Gökhan Tuncel and Mehmet Kayaoğlu for their support in the office works during this work. I also want to thank my managers Mustafa Errize and Salim Çallı at Beko Electronics.

I hope this work will be useful for the person who pays time for reading.

May 2006

Koray ATALAY

CONTENTS

ABBREVIATIONS

DFWMAC	: Distributed Foundation Wireless Media Access Control
MAC	: Medium Access Control
DCF	: Distributed Coordination Function
PCF	: Point Coordination Function
LAN	: Local Area Network
CSMA/CA	: Carrier Sense Multiple Access / Collusion Avoidance
DIFS	: Distributed Inter Frame Space
IFS	: Inter frame Space
RTS	: Ready to Send
CTS	: Clear to Send
NAV	: Network Allocation Vector
TCP	: Transport Control Protocol
NS	: Network Simulation
AODV	: AdHoc On Demand Vector Routing
ELFN	: Explicit Link Failure Notification
DSR	: Dynamic Source Routing
TCP-F	: TCP-Feedback
RFN	: Route Failure Notification
RRN	: Route Reestablishment Notification
IDD	: Inter Packet Delay Difference
STT	: Short-Term Throughput
POR	: Packet Out of Order Delivery Ratio
PLR	: Packet Loss Ratio
ATP	: Adaptive Transport Protocol
RED	: Random Early Detection
LRED	: Link Random Early Detection
COL	: Collusion
DUB	: Duplicate
ERR	: Error
STA	: State invalid
BSY	: Busy
RET	: Retry count exceed
TUOT	: Time Out
IFQ	: Interface Queue
ARP	: Address Resolution Protocol
CBK	: Callback
TTL	: Time to Live
NRTE	: No Route Exists

LIST OF TABLES Page Number

LIST OF FIGURES Page Number

IEEE 802.11 TABANLI KABLOSUZ AĞLARDA ADAPTİF RTS TEKRARI İLE TCP PERFORMANSININ İYİLEŞTİRİLMESİ

ÖZET

Yapılan çalışma, özellikle AdHoc ve Mesh ağlarda, birden fazla düğüm üzerinden geçen güvenli veri aktarımını incelemektedir. Kablosuz ağların kapsama alanlarının genişlemesi ancak ağa yeni düğümlerin eklenmesi ile mümkündür. Yeni düğümler eklendikce birden fazla düğüm üzerinde kurulan bağlantılar kaçınılmaz olmaktadır. Ayrıca kurulan her bağlantının kabul edilebilir bir performans sunması gerekmektedir. Aksi halde Ad-Hoc ve Mesh ağların çok kısıtlı kapsama alanları olacaktır. IEEE 802.11 Ad-Hoc ağ uygulamalarda yaygın olarak kullanıldığından, yapılan çalışmada temel alınmıştır. Kablolu ağlarada kullanılan ve başarısını kanıtlamış TCP protokolu ile 802.11 tabanlı kablosuz ağlarda güvenli veri taşımanın performansı incelenmiştir.

Yapılan incelemeler sonucunda, IEEE 802.11 kapsamında tanımlanan RTS tekrar parametresinin, veri taşıma performansı üzerindeki etkili olduğu görülmüştür. Takiben, NS2 ortamında yapılan benzetilerde, RTS tekrar parametresi arttırılması ile veri gönderim başarımının yükseldiği gösterilmiştir. Son olarak adaptif bir yöntem ile bu parametrenin ortam şartlarına göre değer alması için yeni bir algoritma önerilmiştir. Önerilen algoritmanın performansı NS2 benzetim ortamında ölçülmüş ve sonuçları verilmiştir. Elde edilen simulasyon sonuçlarında, önerilen yöntemin birden fazla düğüm üzerinden geçen TCP bağlantıları için performansı olumlu yönde etkilediği gösterilmiştir.

IMPROVING TCP PERFORMANCE IN IEEE 802.11 BASED WIRELESS NETWORKS WITH ADAPTIVE RTS RETRANSMISSION SCHEME

SUMMARY

This work focuses on ways to improve the reliable data transfer over multi-hop wireless connections. It is important to analyze the multi-hop data transfer in wireless networks because the coverage range of wireless networks increases with the addition of new nodes and performance of these connections should be at an acceptable level. Otherwise, Ad-Hoc and Mesh networks will have limited coverage range. Since IEEE 802.11 is commonly used in MAC layer of Ad-Hoc applications, the 802.11 MAC layer is investigated throughout the work. In addition, the performance of TCP, an important reliable data transfer protocol that works well for wired networks, is analyzed over IEEE 802.11 based multi-hop connections.

During this work, it is shown that the RTS retry value defined in 802.11 has effect on data transmission performance in simulated topologies. Following to that, affect of increasing RTS retry limit on goodput, delay and jitter is investigated with NS2 simulations. This simulation results showed an improvement on goodput and jitter. At the end, a new adaptive algorithm for adjusting RTS retry limit value according to network conditions is proposed. This new scheme is implemented in NS2 and obtained simulation results are presented. These simulation results showed that proposed adaptive algorithm has better performance on multi-hop TCP connections over 802.11 based wireless nodes.

1. INTRODUCTION

Several types of wireless networks have evolved during past years. Sensor Networks, Ad-Hoc Networks, Wireless Local Area Networks, Cellular Networks, Sensor-Actuator Networks and most recently Mesh Networks are the types of wireless networks. Even though the basic element of each wireless network is a wireless node that is capable to communicate through a wireless link, special care should be taken in all layers for different wireless network types. Therefore, there are several open areas for research, which make wireless networks a very large, popular and active field for researchers.

Cellular and Wireless Local Area Networks have a fixed infrastructure formed by Base Stations. Base stations are connected to each other through wired links and these stations are responsible from performing all the routing functions. Wireless nodes must be located in the coverage area of the base stations in order to communicate with base station or other wireless nodes through a base station one hop away.

In Sensor Networks, nodes are equipped with sensors in addition to their wireless network interfaces. In Sensor Networks, there is no fixed infrastructure and large amounts of nodes are deployed into the monitored environment. Data sensed by the nodes is collected in special nodes, which have ability to connect or direct connection to the main storage point where all data is stored. Sensor nodes must perform routing functions to transfer sensed data to the special nodes over other sensor nodes. Moreover, sensor nodes should be low cost and need to use their power efficiently to survive for a long time.

Ad-Hoc Networks have no fixed infrastructure. Each node must perform routing functions in addition to their host functions. Nodes can establish connections to the nodes that are not in their direct communication range. Coverage range of the Ad-Hoc networks extends with the addition of new nodes. Ad-Hoc networks are very suitable for areas where there is no fixed wired or wireless infrastructure. Since there is no need for wired connections, deployment of nodes to extend the coverage area is

very easy, quick and cost efficient compared to Wireless Local Area and Cellular networks. With these features, Ad-Hoc Networks are very feasible to be used in battlefields for communication between troops or in rescue areas where fixed infrastructure has been destroyed.

Sensor-Actuator network is a combination of Sensor Network and Ad-Hoc Network. There are sensor nodes, which are responsible to sense the environment, and actor nodes, which are responsible to take actions according to the collected data. Sensor nodes carry all the properties of nodes in Sensor Networks. Actor nodes have better processing capabilities, higher transmission powers and long battery life.

A MESH network is defined as a combination of all other types of wireless networks. Mesh concept aims to connect all different types of wireless networks. In a MESH Network, there is a wireless backbone, which extends the coverage area of the network. Nodes in the backbone are generally at fixed positions, have multiple network interfaces, are able to perform routing functions and are capable to do protocol conversions for the different types of wireless networks.

Even though each wireless network type has different characteristics, they all face the common problems of wireless communication. Since the environmental conditions can change from place to place and from time to time, there are no fixed conditions at all times between nodes. Hence, it is not possible to get rid of random channel errors due to environmental conditions. Also, all nodes use the same medium to communicate with each other and this results in shared communication channel usage, which also decreases and limits the available bandwidth for each node. Nodes can also be mobile which might cause frequent and dynamic changes in the routes. Routers must overcome these route changes. For these reasons, each layer of the communication model must be tailored well to overcome both application type and common problems of wireless communication.

This work focuses on ways to improve the reliable data transfer over multi-hop wireless connections. It is important to analyze the multi-hop data transfer in wireless networks because the coverage range of wireless networks increases with the addition of new nodes and performance of multi-hop connections should be at an acceptable level. Otherwise, the concepts of Ad-Hoc and Mesh networks will have no meaning. Since IEEE 802.11 is commonly used in MAC layer of Ad-Hoc

applications, the 802.11 MAC layer is investigated throughout the work. In addition, the performance of TCP, an important reliable data transfer protocol that works well for wired networks, is analyzed over IEEE 802.11 based multi-hop connections.

1.1. IEEE 802.11DFWMAC

IEEE 802.11 DFWMAC (Distributed Foundation Wireless Media Access Control) is a MAC standard for wireless networks. This standard defines DCF (Distributed Coordination Function) and PCF (Point Coordination Function) modes for different applications. This work only considers the DCF mode of 802.11. 802.11 DCF is basically designed for Wireless LANs (Local Area Networks) where the AP (Access Point) stands one hop away from the source node and it is performing well in Wireless LANs. IEEE 802.11 is also well suited for Ad-Hoc network applications where there is no AP [1].

IEEE 802.11 is a member of CSMA/CA (Carrier Sense Multiple Access / Collusion Avoidance) family of MAC protocols. When a packet arrives at the MAC layer, the MAC layer senses signals on the media. This process is called Carrier Sensing. If the media is free meaning that there is no packet transmission on the media, the node starts the transmission procedure. After sensing free media, the node waits a time called the DIFS (Distributed Inter Frame Space) duration. After the DIFS duration, if media is still free, the node starts to transmit the MAC frame, which has 47 bytes frame header followed by actual data. When the destination node receives a data packet directed to it, it sends a response to the source node with a MAC Layer ACK message to inform the successful reception of the data packet. This ACK message is 39 Bytes long. If there is a problem and the destination node cannot get the packet correctly, it does not send the ACK packet to source node. If source node does not receive an ACK message for a predefined duration after the transmission of the packet, it retransmits the packet. If source does not receive an ACK message after N retransmissions, it stops retransmission and drops the packet. N is defined as 4 in IEEE 802.11 standard [13].

It is also possible for two nodes to start transmission at the same time after DIFS duration to the same destination node. In this case, collusion occurs in the media and the destination node cannot receive the packets correctly. As a result, the destination

node does not reply with an ACK message to the source node and since source node has not received the ACK message, it retransmits the packet.

Another possibility is that media might not be free when a new packet is arrived at the MAC layer. In this case, MAC layer calculates a random number called Back-off within a pre-defined interval called the contention window. After staying idle during DIFS duration, the node continues to stay idle for another time calculated as back-off number times IFS (Inter frame Space) duration. After this waiting duration, if media is still busy, it doubles its contention window size and picks a new back-off number within the contention window. Contention window is limited by the specification and the node stops doubling the window size when window reaches the predefined limit. If the media is free after all, the node waits for the DIFS duration and starts to transmit the data packet. After successful transmission, contention window size is reset to initial value.

In this scheme, there is a high probability of collusion. To decrease the chances for collusion, RTS/CTS (Ready to Send / Clear to Receive) handshake mechanism, also called CA (Collusion Avoidance), is defined. In this model, the source node sends RTS message before transmitting a packet and waits for CTS message from the destination node. RTS is a 40 Byte message and includes the destination and source MAC numbers and the duration of total transmission. Destination node replies to the RTS message with a 39 bytes long CTS message. This CTS message carries the same information as the RTS message. With the RTS message, the source node informs its neighbors that it has gained access to media and is ready to start transmission. On the other hand, with the CTS message, the destination node informs its neighbors that it is ready to receive packet. This CTS message is more important than the RTS message since without the CTS message, nodes around the destination node may not be able to sense messages coming from the source node and may start transmission at the same time and thus cause a collusion in the destination node. After successful RTS/CTS handshake, the source node starts transmission and the destination node responds with an ACK message after receiving the packet [13].

Another improvement made possible by RTS/CTS handshake is Virtual Carrier Sensing. If a node receives RTS or CTS message from media, it creates an entry into its NAV (Network Allocation Vector) table. This table holds information about source and destination node MAC numbers and durations of transmissions. A node

checks its NAV table before sensing the carrier to decide whether medium is free. If there is an entry in this table, it skips physical carrier sensing and waits until the media becomes free.

Following sections describe the behavior of 802.11 in the case of hidden and exposed terminal problems.

1.1.1. Hidden Terminal Problem

A Hidden node is defined as a node outside the transmission range of a source node but inside the transmission range of destination node when there is a transmission between the source and destination nodes. Since this hidden node does not hear the messages generated by the source node, it can start its transmission while the other is still continuing. In this way, it can cause collusion at the destination node that prevents destination node from receiving the packets of the ongoing transmission correctly.

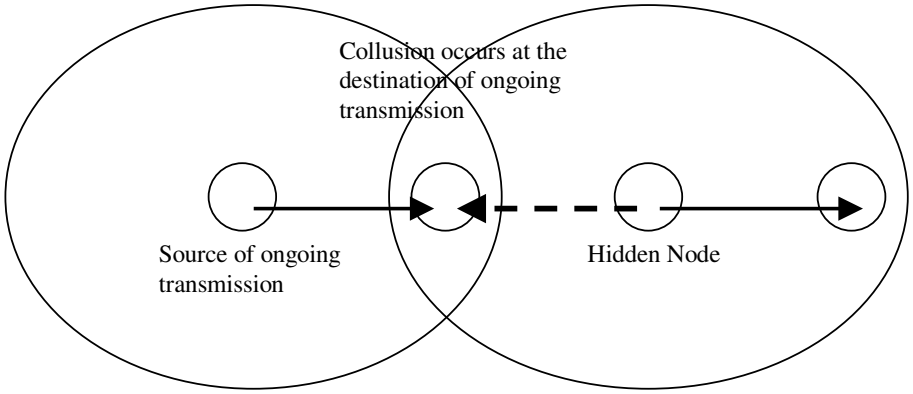


Figure 1.1: Hidden Terminal problem

IEEE 802.11 RTS/CTS mechanism eliminates the hidden terminal problem. However there is still a low probability that both source and hidden nodes can send RTS messages at the same time. This can cause a collusion of RTS packets at the destination node. Another possibility is that the hidden node may not able to receive CTS messages of destination node due to a collusion caused by another node. In that case, the hidden node may start its transmission and can cause another collusion. Hidden node problem most commonly causes data packet loss at the MAC layer because RTS and CTS messages are very short and probability of collusion is very low. In IEEE 802.11 standard, a source node retries to send a data packet only four times and drops it.

1.1.2. Exposed Terminal Problem

An exposed node is a node inside the transmission range of the source node of an ongoing transmission. Since this node is able to receive RTS messages of the source node, it may not be able to respond with a CTS message to any node requesting to send a packet during the ongoing transmission. As a result, the requesting node needs to repeat RTS messages to exposed node until the end of ongoing transmission.

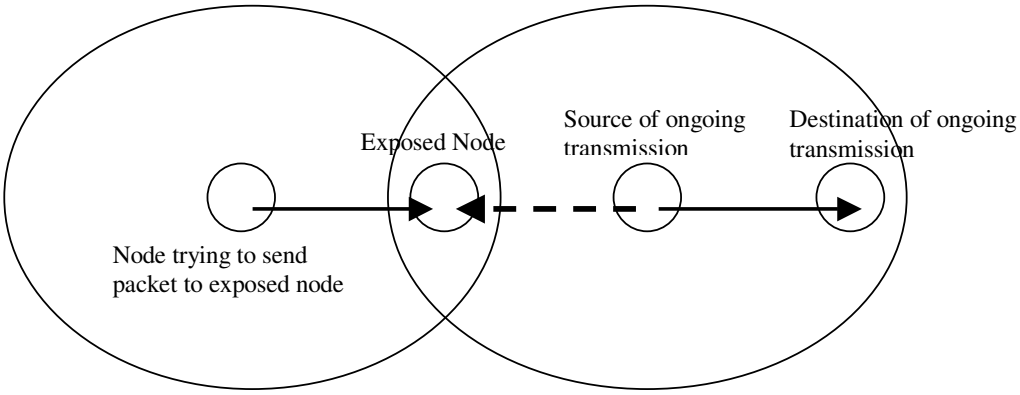


Figure 1.2: Exposed Terminal problem

It is clear that, even with RTS/CTS handshake, IEEE 802.11 MAC protocol is not able to overcome the exposed terminal problem, which does not occur in WLAN applications but is quite possible for Ad-Hoc applications. Exposed terminal problem can also cause packet loss in the MAC layer. According to IEEE 802.11, if a node is not able to receive a CTS message reply of RTS message for seven times, it drops the data packet and informs the upper layers that destination node is not reachable.

1.2. TCP Algorithm

TCP is the de-facto standard for reliable data transfer for the Internet today. It is designed for wired network applications and its parameters are well tailored for fixed wired networks. Following the initial definition of TCP protocol, new variants of the protocol have been proposed each constituting of several modifications of the original scheme.

Basically, TCP is an end-to-end reliable data transfer protocol. A source node initiates the communication and starts to send data packets. For each packet it waits for an ACK packet to confirm the reception of the packet at the destination node. If

the source node does not receive an ACK package, it retransmits the related data packet.

TCP is not only an algorithm for reliable data transfer. It is also able to probe network capacity and tries to use all available capacity at maximum level. To implement such behavior, Congestion Window is defined in TCP protocol as the number of packets that TCP a sender can inject into network without waiting for ACK packets.

Before starting data transmission, a TCP source negotiates with the destination node about the maximum window size. After this handshake, the TCP source starts to probe network and initializes its congestion window size to one and waits for an ACK message of transmitted packet. After receiving an ACK message, the TCP source doubles its congestion window size and hence sends two packets. The window size is doubled each time when all ACK messages for all packets inside the window are received which is a very aggressive behavior to probe network availability. After the congestion window size reaches its limit value, the TCP source stops doubling the window size. After this point, the TCP source increases the window size by one after each reception of ACK messages. This increase is bounded by the receiver or source window size. Window size is determined during the negotiation between the source and destination in the first initialization phase.

The previously described one is the best scenario where the bandwidth between source and destination is not the limiting factor for transmission and no congestion occurred on the network. However wired networks suffer from packet drops due to buffer overflows, which is called congestion. In the case of congestion, intermediate nodes drop packets on the flight. This causes packet losses and source cannot receive ACK messages for lost packets. These packets must be resent to realize reliable data transfer. This is the most realistic scenario experienced in wired networks. TCP is well tailored to handle this situation. The reason for packet loss is congestion, which means overloaded bandwidth. Hence, source should reduce its transmission rate to prevent packet loss. When a packet is lost, TCP senses the existence of congestion in the network and reduces the transmission rate by decreasing congestion window size to its half. Increase and decrease on congestion window continues until all data transmission is completed.

1.3. Multi-Hop Performance of TCP in Wireless Networks

In this section, the performance of TCP over 802.11-based multi-hop connections is measured via NS2 [11] simulations and the focus of this work is presented.

String topology is selected to analyze the performance because of its simplicity. In this simulation, node movement and random channel errors are also disabled to isolate problems faced due to random errors and node movement. Moreover, AODV (Ad-hoc On Demand Vector) routing algorithm is selected, because AODV only generates packets when there is a demand for a route [10]. By using AODV, the extra traffic generated by routing algorithm after a route is established between source and destination is eliminated. With this configuration, the pure performance of TCP and 802.11 over multi-hop connection is measured.

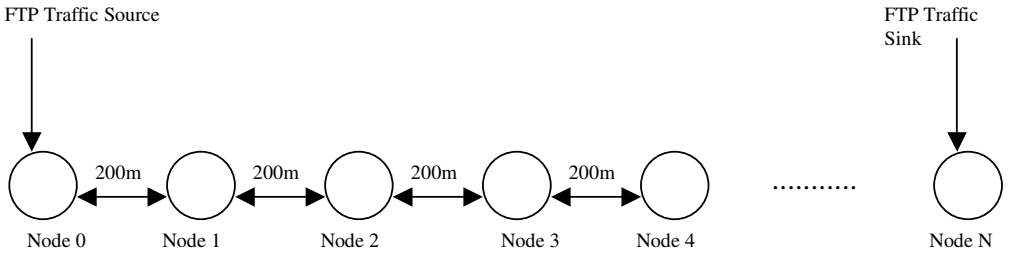


Figure 1.3: Topology used in NS2 simulations.

As seen in Figure 1.3, nodes are placed 200m apart from each other. According to this placement, each node is only able to communicate with its neighbor nodes. In the simulations, IEEE 802.11, which is the most common MAC layer model for Ad-Hoc Networks, is used. Bit rate of 802.11 is left 1Mbps as the default of NS2. In this simulation, the source node is the first node of the string and the destination node is the last one. Only one FTP transfer from source to destination is generated for 500 seconds and TCP packet size is left 1000 bytes as the default of NS2. The goodput is calculated by subtracting duplicated TCP packets from the total received TCP packets.

According to the simulation results shown in Figure 1.4, goodput decreases as the number of nodes increases, but after the number of nodes in the network reaches 10, the goodput stabilizes and changes slightly. Similar results are also obtained in [12] and [15].

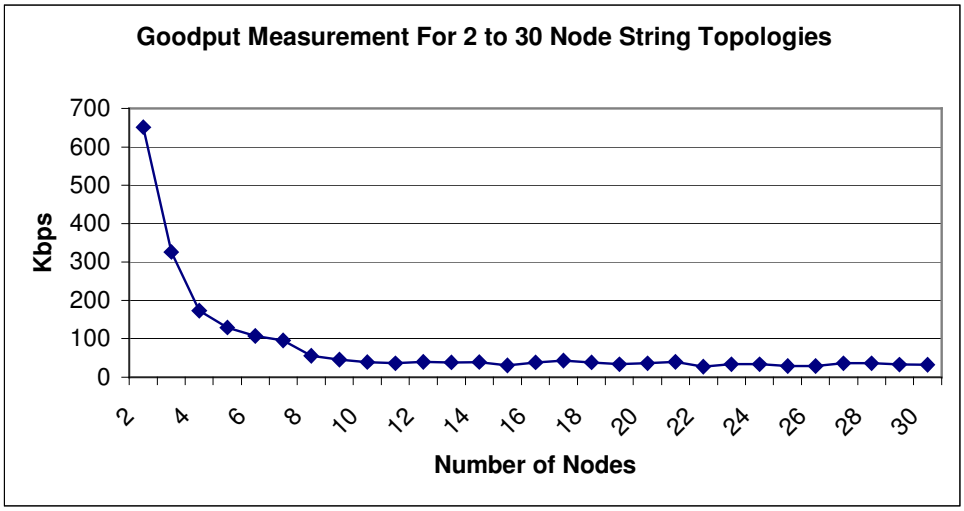


Figure 1.4: Goodput measurement for 2 to 30 node string topologies in terms of Kbps.

Even though a 1Mbps link is available for transmission, for 2-node topology the goodput is around 650Kbps. This decrease is expected due to the overheads of RTS/CTS, Mac ACK and TCP ACK packets. On the other hand, goodput drops almost to its half in 3-node topology. Since there are only 3 nodes and the middle node can only communicate with the first or last node at the same time due to the shared channel, only one packet can be transmitted at a given time. This is the reason why goodput drops below the 1/2 of the 2-node case.

The reason for goodput drop for 4-node topology is the actual result of RTS/CTS handshake. In the case that the first node is transferring data to the second node, the second node sends a CTS message to the first one. Third node knows the ongoing transmission and does not send a packet or responses to the RTS message of fourth node. Again only one transmission can be active at a given time. This drops goodput below the 1/3 of 2-node topology.

To explain the goodput decrease in 5-node case, it is necessary to state the signal transmission range of NS2 wireless card model. In NS2 wireless card model, there are two parameters regarding the range of transmitted signals. First one is the Transmission Range of a node. Transmission Range is defined as the maximum range that a received packet has enough power to be able to be decoded. In NS2 wireless model, transmission range of a node is 250 m. Second one is the Interference range, which is defined as the maximum distance that propagated signal causes noise in ongoing transmissions. Interference range in NS2 wireless model is

550 m. Since the nodes are placed 200 m apart, transmission of a node affects all nodes within 550 m range of source and destination nodes. This means all nodes, reachable in two hops from source and destination nodes will sense ongoing transmissions and will wait until the end of transmission. This is another factor that limits the channel usage for the nodes that are in two-hop distance away from communicating nodes.

5-Node case is special. As explained above, even though node-3 does not receive CTS messages, it will sense the carrier of ongoing transmission taking place between previous nodes. Node-3 will delay its transmission if it has packet to send node-4. Also node-3 will not be able to decode RTS messages of node-4 due to collusion. This actually reduces goodput to the below of 1/4 of 2-node case. Figure 1.5 provides a graphical explanation of transmission ranges.

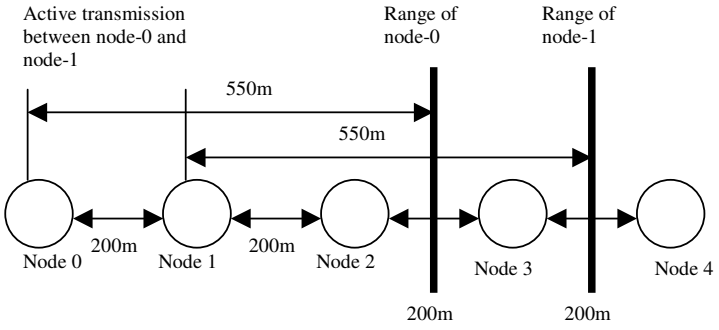


Figure 1.5: Effect of Interfrance ranges on 5-node topology.

Topologies that have more than five nodes should have close goodput values to the one of the 5-node case. This is expected because, in these topologies nodes can reuse transmission channel spatially. However, as seen from the simulation results, goodput decreases below the 5-node case as the number of nodes increases.

This simulation result proves that even if there is no packet loss due to random channel errors or route breaks due to node movement, performance of packet transmission decreases as the number of nodes increases. This work focuses on analyzing the reasons of this problem. Next section includes the previous work done on TCP performance over wireless networks followed by a detailed analysis of the problem and a proposal for improving the performance.

2. PREVIOUS WORK

Performance of reliable data transfer is investigated during the past years. This section provides a summary of the work done to improve TCP performance over wireless networks.

2.1. TCP-ELFN

TCP-ELFN (Explicit Link Failure Notification) algorithm is a variant of standard TCP algorithm used in wired networks. This algorithm is based on the feedbacks received from link-layer routing algorithm. The algorithm is based on a modification of the DSR (Dynamic Source Routing) algorithm to inform source node transport layer with Host unreachable ICMP like message [4].

In the case of link breakage, corresponding node generates and sends Host Unreachable message to the source node. When the source receives this message, it enters a standby mode. In this mode, the source freezes its timers and window size and probes the network periodically using dummy messages. If the source receives any reply packages from the destination, it exits standby and continues to work using stored timer and window size values.

ELFN enhancement improves the performance of standard TCP in the case of random errors experienced during the transmission. However, this algorithm does not handle the problems related to the burst transmission behavior of standard TCP. In case of route changes, commonly seen in wireless networks when the nodes are mobile, the algorithm treats the new route in a similar way to an old route by restoring previous timer and window sizes values. However, the new route may have better or worse conditions.

2.2. TCP – Feedback

TCP-F (TCP-Feedback) algorithm is another variant of standard TCP. Operation of the algorithm is similar to TCP-ELFN. This algorithm uses RFN (Route Failure

Notification) and RRN (Route Reestablishment Notification) messages generated by the nodes on the route [3]. Nodes on the path generate RFN messages to the source node when there is no available path to the destination and they record the link failure in their internal databases. When the source receives this message, it enters standby mode similar to the one in TCP-ELFN and freezes its timers and window sizes. TCP-F algorithm does not generate dummy packets to probe link status, but source node starts timers to exit standby mode.

If intermediate nodes find an available path to the destination, they generate RRN messages to the source. When a source receives an RRN message, it restores its timers and window size values and resumes sending packets. If no available path is found until the source timer is expired, the source restarts its timers and executes congestion control algorithm.

This algorithm does not use probe packets to detect availability of the destination. This is an improvement over TCP-ELFN because source node does not cause dummy traffic in the network while it is in standby. However it does not handle burst traffic nature of TCP and conditions of an entirely new route.

2.3. Multi-Metric End-to-End Measurements

This algorithm is based on several measurements made on transmission path to detect the state of the active path [14]. By using these measurements the source decides how to response to changing conditions using standard TCP. States needed to be detected in wireless networks are congestion, random error, route change and disconnection states. In the case of congestion, transmission rate should be reduced. If a channel error or route change occurs, source should maintain its transmission rate. When the disconnection is experienced, source should freeze its current state and probe the network until the new route is established.

There are four measurements to detect the route state. First one is the Inter-packet Delay Difference (IDD). It is obtained by calculating delay differences between consecutive packets. IDD is used to detect congestion in the network. Random errors and source packet sending behavior does not affect IDD value. However IDD value becomes imprecise in the case of mobility induced out of order packet delivery. Second measurement is Short-Term Throughput (STT). STT is less sensitive to out

of order packet delivery and very imprecise in the case of burst channel errors, network disconnection and source transmission rate behavior. However STT parameter can be used to identify congestion in the network. Third measurement is Packet Out-of-order delivery Ratio (POR). POR is used to identify route changes. Last measurement is Packet Loss Ratio (PLR). PLR is measured by calculating missing packets in the receiving window and is used to identify channel errors.

The state of network is determined by combining the four measurement techniques. Destination node makes the measurements and informs the source by setting reserved fields in TCP-ACK packets. Under normal conditions, transport algorithm works as standard TCP and the destination makes the measurements. When the destination notices a problem signaled by any of four types of measurements, it determines the network state and informs the source node. The source takes necessary actions when it receives the state information.

This algorithm identifies the network state and treats each state differently. However making measurements in destination and informing the source node by ACK messages can cause problems when the ACK messages are lost in the network. Since this algorithm depends on standard TCP, it does not handle burst traffic nature of TCP and does not adapt to new route conditions.

2.4. ATP (Adaptive Transport Protocol)

ATP is proposed as an entirely new transport protocol [2]. It depends on rate base data transmission. ATP uses quick start during connection initiation and route switching. During the route establishment, intermediate node informs the source about the available bandwidth and the source adjusts its initial transmission rate according to this information. During the transmission, intermediate nodes prepare messages to inform source node about the status of the connection. In the case of congestion, the source node adjusts its transmission rate.

Since ATP is rate-based protocol, it does not use retransmission timers. Moreover, it does not wait for ACK packets to insert new packets in to network. In the case of packet loss, the destination node informs the source by sending negative ACK packets. This also reduces backward acknowledge traffic generated by the destination node. These features of ATP handle the burst nature of standard TCP, and

moreover ATP adjusts its transmission rate according to the conditions of new route. On the other hand, there is a need for transport layer protocol change if there is a connection between wireless network and wired network.

2.5. LRED Algorithm

This work specially focuses on multi-hop performance of 802.11 and TCP protocol. Authors showed that aggressive behavior of TCP results instability in TCP window due to the packet drops experienced in overloaded nodes. In this work, they have shown an optimum TCP window size where network resources are used at optimum level.

Authors [5] proposed a packet drop algorithm in link layer of intermediate nodes to stabilize TCP congestion window around optimum window size. This algorithm works like Random Early Detection (RED) algorithm used in wired networks. Each intermediate node calculates the moving average of packet retransmissions. If this value is less than a predefined threshold, packets are passed without any marking and dropping process. Otherwise packets are marked or dropped according to probability calculated from average retransmission value.

Dropping of TCP packets prevents the growth of congestion window and stabilizes it around an optimum size before the network starts to suffer from overload. Since this algorithm operates in the link layer of intermediate wireless networks and does not change standard TCP behavior, it might be suitable for connections that have nodes in wireless and wired networks.

2.6. Dynamic Adaptive Acknowledge Strategy

The main idea of dynamic adaptive acknowledge strategy is to delay ACK packets and merge them to minimize the number of ACK packets in transmit [7]. Normally, TCP destination sends ACK packets for each received TCP packet and each received ACK message in the source triggers the growth of congestion window and release of new TCP packets in to network. Since the TCP data packets and ACK packets are carried in the same medium, reverse ACK packets cause additional backward traffic. So reducing the number of ACK packets will decrease the backward traffic and regulate the source congestion window size.

This algorithm measures the inter delay of received TCP packets and adaptively calculates maximum inter delay timeout value. If TCP packets are received within the maximum inter delay time, one ACK message is sent after receiving four TCP packets. If an out of order TCP packet or a TCP packet filling the gap in receiver buffer is received, the destination does not delay any ACK packets and sends the appropriate ACK packets immediately. If the destination cannot receive TCP packets during calculated timeout duration, it creates ACK packets without waiting for four TCP packets.

2.7. Packet Merging

Authors of [6] aim to improve the TCP performance of wireless networks by combining TCP data and ACK packets' transmissions in intermediate nodes on connection path. ACK packets are very small when compared to data packets and transmitting ACK packets in the backward traffic direction wastes the transmission channel. Since TCP traffic generates forward data transmission and backward ACK transmission on the same transmission channel, merging the transmission of TCP data and ACK packets will make backward traffic transparent form the forward data transmission and this will improve the performance in wireless connections.

In each wireless node, there are two separate queues for TCP data and ACK packets. When a node wants to send a packet, it takes one packet from data queue and one packet from ACK queue and propagates two messages in order when it gets the control of channel. Both nodes in data transfer direction and ACK transfer direction receives data and ACK packets. According to the received packet headers, node receiving both packets drops the packet that is not designated to it. A node receiving the TCP data packet sends MAC layer ACK first and other node waits for a while to not cause congestion in packet source node.

2.8. Adaptive RTS/CTS Control Mechanism for IEEE 802.11

IEEE 802.11 has two options for data transmission. In the first one, source propagates MAC data when the transmission channel is available and waits for MAC ACK message. The second option is to propagate MAC data after two-way handshake with RTS/CTS mechanism. RTS/CTS handshake is proposed to decrease

Hidden/Exposed terminal related packet collusions. However, destination node may not be able to send CTS message even when it is not busy. In that case, the source tries RTS/CTS handshake for several times and drops the related packet. In the case of light traffic, RTS/CTS mechanism will work well, but when the network is overloaded, disabling RTS/CTS mechanism will yield better performance.

Authors of [8] proposed a dynamic decision algorithm for enabling and disabling the RTS/CTS handshake. In their algorithm, each node counts the number of waiting CTS timeout events. If this value is smaller than a predefined threshold, the node sends packets with RTS/CTS handshake. Otherwise, the node sends packet without RTS/CTS handshake. They have also shown that dynamic switching between two modes has better performance with TCP traffic than using only one method.

2.9. Adaptive Increase of RTS Retransmission in IEEE 802.11

In IEEE 802.11 MAC layer with RTS/CTS handshake method, a source node tries to send each RTS message seven times. If the source cannot get response after seventh try, it drops the related packet. This generally happens when the nodes are overloaded with the generated traffic.

Authors of [9] proposed to adaptively increase the RTS retransmission limit according to the previous retransmission values. In their algorithm, if the previous retransmission value is increasing compared to one before, RTS retransmission limit is increased by a constant. Otherwise RTS retransmission value is decreased by the same amount. Also they have shown that increasing RTS retransmission value above 20 has no significant effect on performance. For this reason they have limited the increase of RTS retry limit above 20. On the other hand, they limit the decrease of RTS retransmission below the standard value seven. According to their results, IEEE 802.11 performs better than keeping RTS retransmission limit at seven.

3. DETAILED INVESTIGATION OF RTS RETRY LIMIT

The decrease on goodput shown in figure 1.4 is investigated through out this section. In the next section, MAC and router layers are analyzed and it is shown that the reason of the problem is RTS retry limit parameter defined in IEEE 802.11 standard. Following to that, effect of changing RTS retry limit on goodput, delay and jitter metrics will be shown for several length string topologies with NS2 simulation results.

3.1. Analysis of MAC and Router Layer

In this section, MAC and router layers are analyzed at node base using NS2 trace files. In these analyses, only 12-node string topology is considered and one FTP connection is created between source and destination nodes. Simulation duration is 3000 seconds and simulations are run several times. Similar results are obtained for each simulation run. Throughout this section, only the results of one simulation are given.

3.1.1. Explanation of Packet Drops in NS2 Wireless Model

There are several reasons of packet drops in NS2 wireless model. Table 3.1 is given to explain the reason of drop actions in NS2.

Table 3.1: Explanation of packet drops in NS2

	Explanation of Packet Drop
MAC LAYER	
COL	Collusion
DUB	Duplicate
ERR	MAC Error
STA	MAC State invalid
BSY	MAC Busy
RET	Retry Count Exceed
ROUTER LAYER	
LOOP	Loop on Route

TOUT	Time Out
IFQ	Interface Queue Full
ARP	ARP Queue Full
OUT	Outside Subnet
CBK	Call Back
TTL	Time to Live
NRTE	No Route Exceed
GENERAL	
END	Simulation End

3.1.2. Node Base Analyze of MAC Layer

Table 3.2 is given to show the type and the number of packet drops experienced during the simulation in MAC Layer.

Table 3.2: Number of packet drops at MAC layer of each node in 12-node topology.

Node Number	COL	DUB	ERR	STA	BSY
0	39	0	0	0	2
1	11085	0	0	0	3
2	16669	0	0	0	4
3	13504	0	0	0	5
4	13488	0	0	0	12
5	13548	0	0	0	9
6	11763	0	0	0	23
7	10981	0	0	0	30
8	14947	0	0	0	11
9	6890	0	0	0	10
10	2695	0	0	0	3
11	32	0	0	0	0
Total	115641	0	0	0	112

As seen in the Table 3.2, the main reason of drops in MAC layer is the collusion experienced by nodes. On the other hand, there are no drops due to DUB, ERR and STA. Packet drops due to BSY is also experienced by the nodes but the occurrence of this case is very rare when compared to collusions.

It is seen that collusions occurs rarely in source and destination nodes because they have only one neighbor. On the other hand, middle nodes experience collusions most

frequently. Table 3.3 is provided to show the type and the number of packets that are dropped due to collusions. As seen in table, most of the collusions are experienced on RTS packets. Number of collusions on TCP data packets and AODV data packets follows the RTS packets.

Table 3.3: Types and numbers of packets dropped due to the collusion in each node of 12-node topology.

Node Number	**RTS Packets**	**CTS Packets**	**TCP Data Packets**	**TCP ACK Packets**	**AODV Packets**	**MAC ACK Packets**	**Total**
0	39	0	0	0	0	0	39
1	10843	27	95	0	120	0	11085
2	16367	39	146	1	116	0	16669
3	13098	69	119	79	139	0	13504
4	13121	72	117	72	106	0	13488
5	13169	68	118	77	116	0	13548
6	11412	77	104	74	96	0	11763
7	10639	67	112	77	86	0	10981
8	14408	106	252	74	107	0	14947
9	6616	6	0	149	119	0	6890
10	2662	1	0	8	24	0	2695
11	32	0	0	0	0	0	32
Total	112406	532	1063	611	1029	0	115641

Node-0 and Node-11 experience collusions only while receiving RTS packets. For Node-0, this can happen only if Node-1 starts transmitting RTS message just after the start of Node-2 transmission. Since the interference range of a node is 550m, it will effect the reception of the 400 m away Node-0. This is the only scenario that can be the case for the RTS message drops in Node-0 and Node-11. Also, the number of drops is very low when compared to total drops. This confirms that the probability of this case is very low. From this explanation, reason for the drop is due hidden terminal problem. In this example, Node-2 and Node-9 are the hidden terminals for Node-0 and Node-11 respectively.

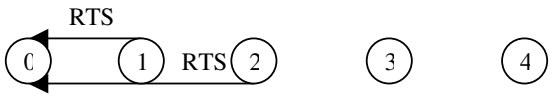


Figure 3.1: Explanation of RTS collusion at node-0.

Node-1 can experience collusions in several ways. The first case is when Node-0 and Node-2 start transmitting RTS packets at the same time. Secondly, Node-0 can send RTS to Node-1 just after Node-3 RTS transmission to Node-2. Additionally, if Node-2 sends CTS packet at the same time as the Node-0 RTS packet, they can collide at Node-1. According to the Node-0 collusion counts, such timing's probability is very low and is not enough to explain the total collusions in Node-1. Another collusion in Node-1 may happen if Node-0 sends an RTS message while Node-3 is transferring data packets to Node-4. Since Node-0 is not able to detect the Node-3's transmission, it can initiate the RTS transmission. On the other hand, Node-1 is inside the interference range of Node-3's transmissions and will not able to receive the RTS packets of Node-1 due to collusion. This collusion will exist until Node-3 completes the transmission. Since the transmission of data packets takes much longer time than the transmissions of other MAC packets, the chances for RTS collusions are higher for this case. This is the main reason why the number of RTS collusions increases tremendously. Moreover this case is the exposed terminal case that occurs in this network configuration.

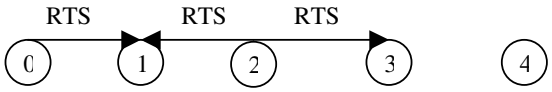


Figure 3.2: Explanation of RTS collusion at node-1.

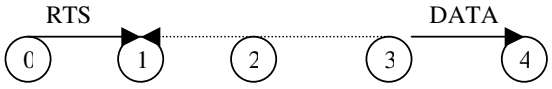


Figure 3.3: Explanation of RTS collusion at node-1.

In addition to the exposed terminal problem, middle nodes experience another collusion case due to the reverse traffic. As seen in the Figure 3.4, Node-3 will not hear the data transmission between Node-0 and Node-1. At any time, it can initiate RTS transmission to Node-2 and this transmission will actually cause RTS collusion in Node-2. This is clearly the hidden terminal problem, which exits due to the

interference ranges. For these reasons, it is expected to have more collusion in RTS messages in middle nodes.

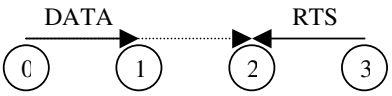


Figure 3.4: Explanation of RTS collusion at node-2.

As seen from the table 3.3, there is no data packet collusion in Node-0 and Node-11. This is as expected because of RTS/CTS handshake. On the other hand, even if RTS/CTS handshake is used in middle nodes, there is still collusion of data packets. As seen from the figure 3.5, this can happen only if Node-3 initiates RTS message to its neighbors. In that case, packets will cause collusion in Node-1 and Node-1 will not be able receive the data packet correctly. This problem is again the hidden terminal problem. Moreover, explained case is exactly the same as the case given in figure 3.4. This means that data collusion may be the reason of RTS collusion in other nodes. In this example, data collusion at Node-1 is actually RTS collusion at Node-2.

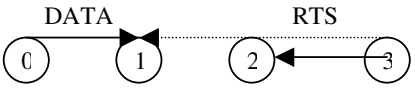


Figure 3.5: Explanation of data collusion at node-1.

There is not any other possibility for data collusions in simulated topology. Preceding nodes of source might be thought of as a reason for data collusion. But, this is not possible since they can detect the carrier of source transmission. Nodes that are far away to detect the source node messages will be also far away to cause interference at the destination.

As a result, one can conclude that RTS collusions can occur due to exact timing of nodes but the probability is very low. Another case of RTS collusion is due to the hidden terminal problem and hidden terminal problem is the only reason of data packet drops. From the results of the table 3.3, it can be observed that data packet drops occur very rarely as compared to RTS drops. For this reason, one could say that exposed terminal problem causes the most of the RTS packet drops.

In the case of RTS collusions, the destination node cannot send back CTS message to the source node. In that case, the source node repeats the RTS message. As stated

before, in default IEEE802.11 MAC standard, RTS message is repeated seven times. Also if data collusion happens, the source node cannot receive a MAC ACK message from the destination node. The source node tries to send message again and according to the standard data messages are repeated four times. If retry limits are exceed, packet are dropped in MAC layer due to Retry Count Exceed (RET). Table 3.4 is given to show the number of packet drops in each node due to RET.

Table 3.4: Types and numbers of packets drop due to RET in each node of 12-node topology.

Node Number	**TCP DATA RET**	**TCP ACK RET**	**AODV RET**	**TOTAL DATA RET**	**RTS RET**	**Total**
0	182	0	0	182	182	364
1	86	0	0	86	86	172
2	57	0	3	60	60	120
3	51	44	4	99	99	198
4	30	51	1	82	82	164
5	32	31	0	63	63	126
6	9	45	0	54	54	108
7	15	34	0	49	49	98
8	0	32	0	32	32	64
9	0	34	0	34	34	68
10	0	39	0	39	39	78
11	0	14	0	14	14	28
Total	462	324	8	794	794	1588

As seen in the table 3.4, the total number of data RETs for each node is equal to the RTS RET of each node. When this is analyzed in NS2 trace file, it is seen that after each RTS RET there exists a data RET. To be sure, some code is added into the NS2 802.11 files to find the packet drop reason. The result shows that 794 data packets are dropped due to the RTS RET and no data packets are dropped due to data RET. This also shows that four times retry of data packets is enough to recover collusions in the simulation scenario but retry limit for RTS messages is not enough to cope with RTS collusions.

From these results, one can conclude that the actual reason for packet drops in MAC layer is the retry count exceed due to the RTS collusions. Also, as explained above, most of the RTS collusions are due to the exposed terminal case. It must be stated

that, MAC layer does not discharges the packets physically. In the case of retry count exceed, MAC layer stops trying to send packet and informs the upper layers.

3.1.3. Node Base Analysis of Router Layer Packet Drops

Packets can be dropped in Router Layer due to several reasons. Table 3.5 is given below to show the number of packet drops in each node of 12-node string topology with their reasons.

Table 3.5: Reason and number of packet drops in each node of 12-node topology.

Node Number	LOOP	TOUT	IFQ	ARP	OUT	TTL	CBK	NRTE
0	0	0	0	0	0	0	402	0
1	0	0	0	0	0	0	127	105
2	0	0	0	0	0	0	75	41
3	0	0	0	0	0	1	88	236
4	0	0	0	0	0	0	47	279
5	0	0	0	0	0	0	40	156
6	0	0	0	0	0	0	85	57
7	0	0	0	0	0	0	48	121
8	0	0	0	0	0	1	42	39
9	0	0	0	0	0	0	39	48
10	0	0	0	0	0	2	44	67
11	0	0	0	0	0	0	26	0
Total	0	0	0	0	0	4	1063	1149

Table 3.5 shows that there is no packet drop due to the LOOP, TOUT, IFQ, ARP and OUT, but very few drops due to the TTL. According to the results, The TCP ACK packet experiences one of the TTL drop and AODV packets experiences the rest three drops. On the other hand, packets are mostly dropped due to the Callback (CBK) and No Route Exceed (NRTE). As stated before, MAC layer drops a packet if it is not able to receive CTS message after 7^{th} try of RTS. When this happens, MAC layer triggers router layer since RTS RET means that it is not possible to reach the node where packet should be sent, which also means there is a break on previously established route to destination. In that case, router layer may drop all packets in its queue. This is called CBK in NS2 simulation trace file. Additionally, route layer stops sending packets and starts route establishment procedure. Even a node starts

route reestablishment procedure, some of the nodes around it can still send packets to it directed to broken link. When a node receives such packet, it may drop the packet due the no route exists which is called NRTE in NS2 trace files.

Table 3.6: Type and reason of packet drops in each node of 12-node topology.

Node Number	TCP CBK	ACK CBK	AODV CBK	TOTAL CBK	TCP NRTE	ACK NRTE	TOTAL NRTE	TOTAL
0	402	0	0	402	0	0	0	402
1	127	0	0	127	105	0	105	232
2	75	0	0	75	41	0	41	116
3	87	0	1	88	38	198	236	324
4	47	0	0	47	18	261	279	326
5	40	0	0	40	7	149	156	196
6	0	85	0	85	44	13	57	142
7	0	48	0	48	88	33	121	169
8	0	42	0	42	0	39	39	81
9	0	39	0	39	0	48	48	87
10	0	44	0	44	0	67	67	111
11	0	26	0	26	0	0	0	26
Total	778	284	1	1063	341	808	1149	2212

Table 3.6 shows the type and number of packets dropped in router layer of each node. It is seen that most of TCP Data packets are dropped in the source node due to the CBK. This is as expected because, TCP generally generates burst traffic and this increases the number of packets in router layer queue. TCP packet drops due to the CBK decreases toward the middle nodes. After Node-5 there is no TCP Data packet drop due to CBK. The reason for that is the local route repair behavior of AODV algorithm. According to the AODV model used in NS2, if node is more close to the destination than the source, it does not drop the packets with CBK and buffers the packets in its queue. After that it starts to establish new route locally. This explains why the nodes that are after Node-5 do not execute CBK for the packets in the queue. This is also similar for the reverse TCP ACK traffic. Nodes before Node-6 never drop TCP ACK packets due to the CBK.

However, TCP Data packet drops due NRTE still exits until Node-8. Reverse ACK traffic also experiences similar behavior as the TCP data traffic down to Node-2. For TCP data traffic, nodes other than Node-6 and Node-7 drop the received TCP

packets as a result of route failure. However, for Node-6 and Node-7 the situation is different. Since these nodes are closer to the destination, instead of dropping packets after route failure, they buffer them until the local route establishment of AODV algorithm. However, these nodes may not able to establish new routes to the destination because AODV messages may not reach to the neighbor nodes due to the traffic. This will result in a failure in the AODV route reestablishment and the packets in queue will be dropped with NRTE. However, the simulation results show that nodes closer to the packet destination are able to establish new routes to the destination in the case of route error.

Table 3.7: Number of sent, received and lost packets in 12-node string topology.

	Number of Packets
TCP Data Packets Created	16862
TCP Data Packets Received	15742
TCP ACK Packets Created	15742
TCP ACK Packets Received	14649
TCP Data Packet Loss	1119
TCP ACK Packet Loss	1093
Total Lost	2213

Results in Table 3.7 are obtained from the agent level traces and number of packet losses is calculated by subtracting the number of packets received from the number of packets sent in source and destination nodes. Table 3.8 is provided to compare the agent level packet drops with the router level drops due to the NRTE and CBK. It is seen that agent level calculated TCP data packet loss is exactly same to the TCP data packet drops due to the CBK and NRTE. For TCP ACK packets, there is one missing packet. Using the result of the previous analysis, one could say that this packet was dropped due to the TTL.

Table 3.8: Packet lost in agent level and router level in 12-node string topology.

	According to the Agent Outputs	**Total of CBK and NRTE drops**
TCP Data Packet Loss	1119	1119
TCP ACK Packet Loss	1093	1092
Total Loss	2212	2211

3.1.4. Node Base Analysis of TCP Packets

Figure 3.6 is provided to show the incoming and outgoing TCP packets in each node of simulation topology. According to the figure, source agent creates around 5.6 TCP packets per second, but receives around 4.8 TCP ACK packets per second. There are 0.8 packets per second difference between sent and received packets. The source node must retransmit this difference in each second. As a result, source node only inserts 4.0 new packets per second in to the network.

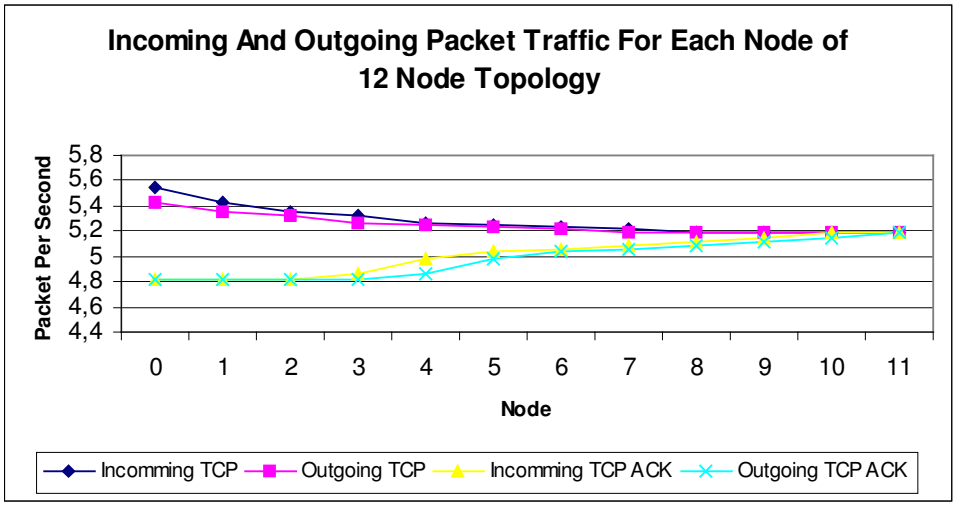


Figure 3.6: Incoming and outgoing packets in each node of 12-Node string topology

Figure 3.7 is provided to find out at which node most of the packet drops occur. Most of the TCP data packets are dropped in the source node before entering the network. This is the result of aggressive packet insertion behavior of TCP algorithm. Most of TCP ACK packets are dropped in Node-4 and nodes around it. Since Node-4 is closer to the destination, in the case of route failure, it will queue all incoming packets until AODV reestablishes the local route to the destination. However, nodes around source will be busy due to the source traffic at this time and they will not be able reply route requests of Node-4. In that case, AODV stops trying to find route after several retries and Node-4 drops all the packets in its queue. This case is similar for the nodes around Node-4.

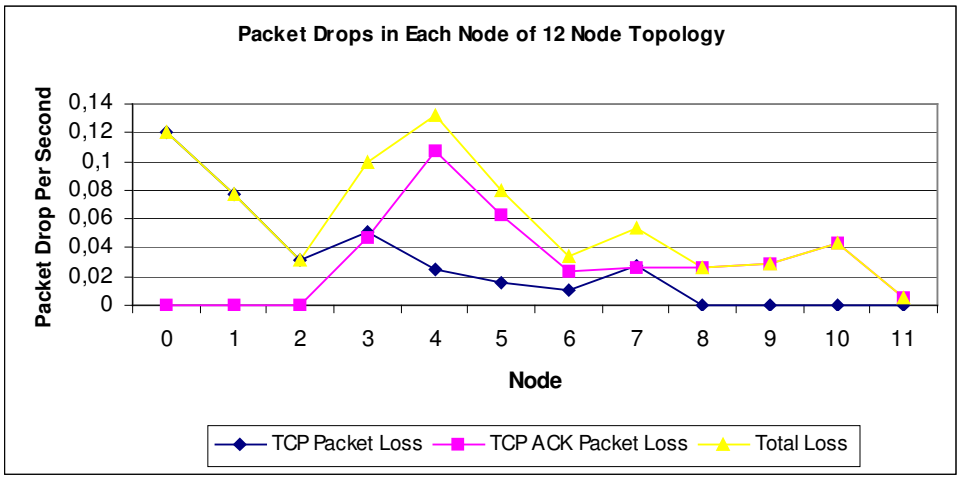


Figure 3.7: Packet drops in each node of 12-Node string topology

All these simulation results explain the reason of performance decrease in simulated topology. Due to the exposed terminal problem, RTS RET error exists in MAC layer. Following that, even nodes are stationary, router layer starts to assume that the designated node is not in its transmission range. As a result of this, router layer may drop all packets in its queue and triggers route reestablishment procedure. Moreover, until the route is reestablished, all packets coming from other nodes to the same destination may be dropped due to no route to destination. Moreover, once can say that local route reestablishment of AODV algorithm generally ends unsuccessfully and results with drop of the packets buffered in node.

3.2. Effect of RTS Retry Parameter

In the previous section, it was shown that the basic reason for packet drops is the exposed terminal problem. Exposed terminal problem causes RTS retry count exceeds and this triggers route errors. In the case of route errors, a node drops all the packets in its queue or starts local route recovery according to the distance to destination node. In IEEE 802.11 standard, retransmission of RTS message is limited to seven and a packet is dropped if CTS is not received to any of retransmitted RTS. In this section, the effect of RTS retry limit on packet drops is investigated.

12-node string topology is used to show the effect of RTS Retry Limit on packet drops. Only one FTP connection is created between the source and destination node during the simulations. Each simulation is run for 3000 seconds and RTS retry limit is changed from 10 to 28. Even if simulations are run several times, similar results

are obtained for same configuration. To be consistent, only one simulation result is provided during this section for each configuration.

In table 3.9, total number of collusions, RET drops, data packets and collusions per packet are provided. Total packets entry corresponds to the sum of TCP data and TCP ACK packets.

Table 3.9: COL and RET values for different RTS retry limits in 12-Node string topology.

	RTS-7	**RTS-10**	**RTS-13**	**RTS-16**	**RTS-19**	**RTS-22**	**RTS-25**	**RTS-28**
Total COL	115641	243787	341839	397781	409405	411917	412467	410702
Total RET	794	405	187	44	7	0	0	0
Total Packets	32604	53959	69945	78494	80409	81015	80916	80847
COL Per Packet	3,546	4,518	4,887	5,067	5,091	5,084	5,097	5,079

From table 3.9, once can observe that number of collusions increases as the RTS retry limit value increases. This is as expected because, there are more packets inserted in to network and nodes try for more number of times to send a packet. It will be meaningful to analyze the COL per packet value. As seen in Table 3.9, COL per packet increases but this increase is not as sharp as the total collusions. The reason is that total number of packets increases as the RTS value changes

Another important observation is that, increasing RTS Retry Limit above 19 does not cause dramatic changes on any of measured values. This means that RTS Retry Limit 19 is enough for this simulation configuration to prevent severe packet drops due to the retry count exceed.

Figure 3.8 shows the packet drops in router layer due to the NRTE and CBK. It is seen that as RTS retry limit increases, the number of CBK and NRTE drops decreases. This is expected because; the reason for packet drops in router layer was due to the exceeding of RTS retry limit. For RTS Retry Limit above 19, there are no packet drops and RTS retry limits 22, 25 and 28 yield similar results.

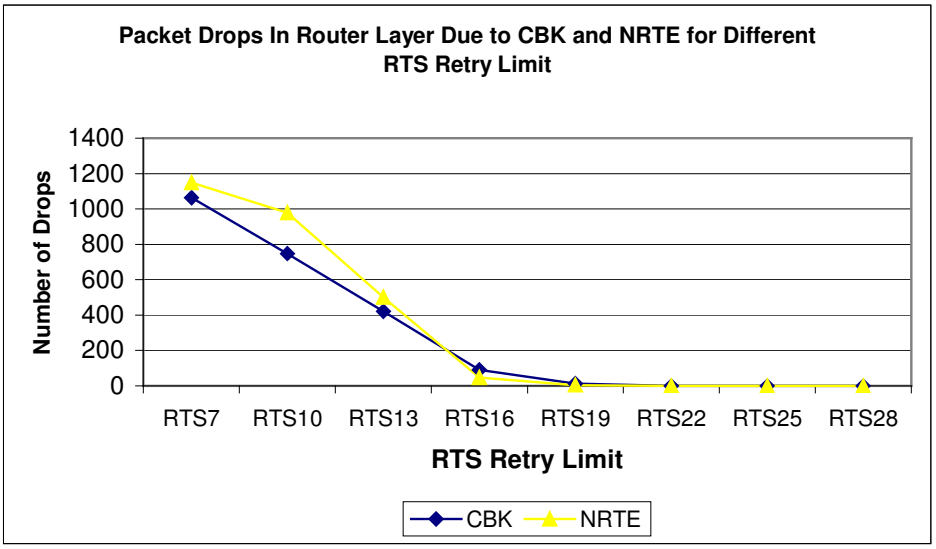


Figure 3.8: CBK and NRTE values for different RTS retry limits in 12-node string topology.

Until this point, it is shown that exposed terminal problem is the main cause of the performance problem in our simulation configuration. The MAC layer exceeds the RTS retry count limit and triggers the router layer as a result of exposed terminal problem. With above simulations, it is verified that increasing RTS retry limit decreases the number of RTS retry exceeds in MAC layer and this also decreases packet drops in router layer due route breaks. Following sections provides simulation results to show the effect of increasing RTS retry limit on goodput, Delay and Jitter.

3.2.1. Goodput Measurement for Different RTS Retry Limit

In this section, goodput is measurement for different RTS retry limit values for 8, 12, 16 and 20-Node string topologies. Only one FTP connection is created between source and destination nodes and simulation is run for 3000 seconds. Each simulation configuration is run several times. Provided results in this section are the average result of each simulation configuration. Goodput is measured in the destination node by subtracting the number of retransmitted packets from the total number of received packets.

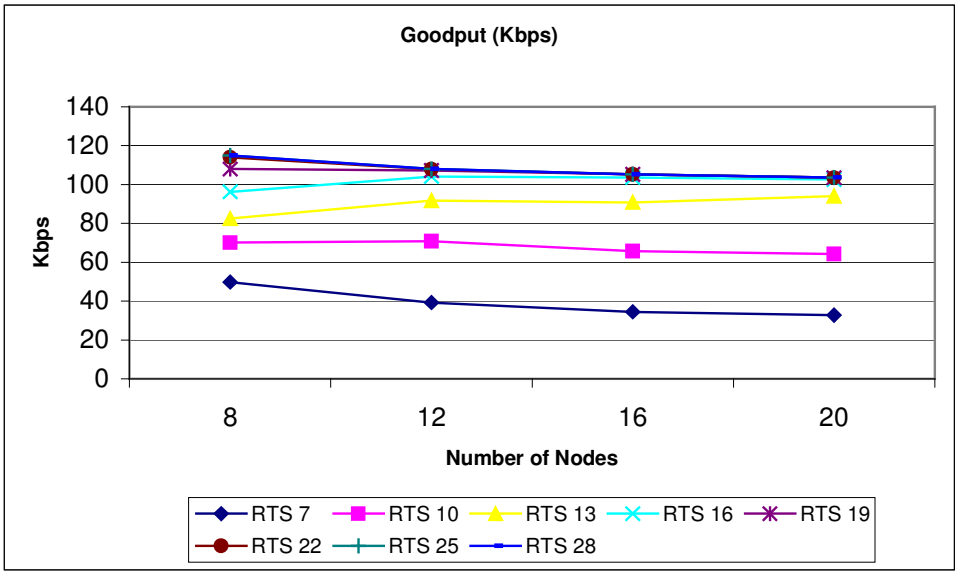


Figure 3.9: Goodput measurements for different RTS trial limits in 8, 12, 16 and 20-node topologies.

Figure 3.9 is the result of goodput measurements. From the figure, it is seen that goodput increases by increasing RTS retry limit value. On the other hand, increasing RTS retry limit above 19 causes no major change on goodput. This simulation result is as expected since the effect of RTS retry limit on the number of packets inserted in to network was shown before.

Packet loss rate is also measured during these simulations. Figure 3.10 provides packet lost rate for each simulation configuration. As the RTS retry limit increases, packet lost rate for each topology decreases. Similarly, increasing the RTS retry limit above 19 does not cause major changes.

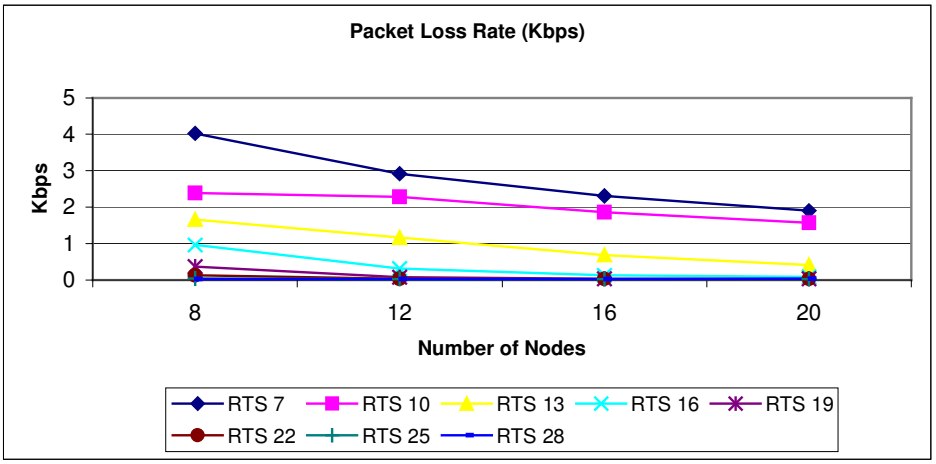


Figure 3.10: Packet loss rate measurement for different RTS trial limits in 8, 12, 16 and 20 node string topologies.

It has been shown that as the number of nodes increases the packet drop rate decreases for all RTS retry value. One of the reasons for that might be the decrease in the number of packets inserted in to the network.

3.2.2. Delay Measurement for Different RTS Retry Limit

In this section, the delay is measured for different RTS retry limit values in 8, 12, 16 and 20 Node Topologies. Simulations run for goodput measurement are used in delay measurements. Delay is measured for the packets, which did not drop during their path from source to destination. For dropped or duplicated packets, delay is measured only for the latest packet that reaches the destination.

Figure 3.11 provides the average delay experienced in each configuration. It is seen that average delay increases as RTS retry limit increases. This is as expected because increasing RTS retry limits causes packets to remain in MAC layer for a longer time due to the increased number of retries. Also, one can see that increasing RTS retry limit above 19 does not have a significant effect on the average delay. Moreover, one can say that as the number of nodes increases, delay experienced by the packets increases. This is the actual result of the increasing the number of nodes.

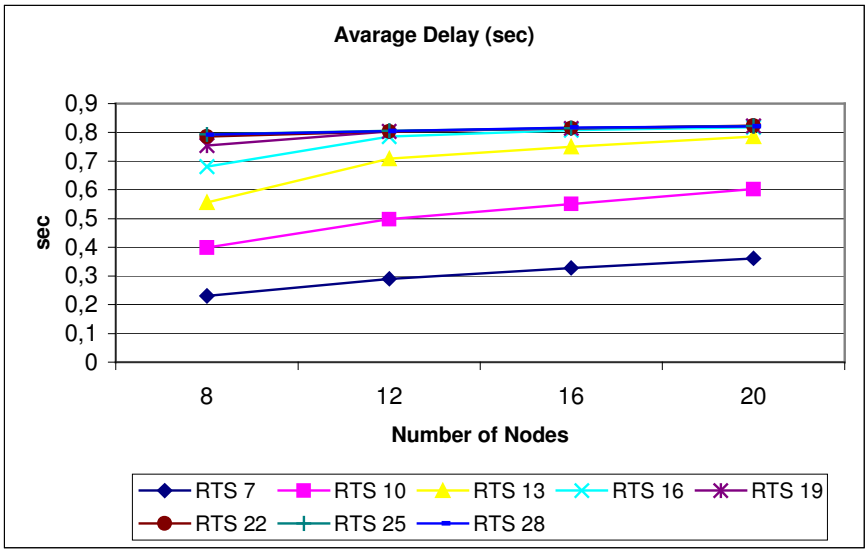


Figure 3.11: Average delay measurements for different RTS retry limits in 8, 12, 16 and 20 nodes string topology.

Figure 3.12 is provided to compare minimum delay experienced in each configuration. As seen in the figure, minimum delays are very close to each other. It

is also seen in the figure that as the number of nodes increases, minimum delays increases. Minimum delays are given as the minimum of each simulation runs.

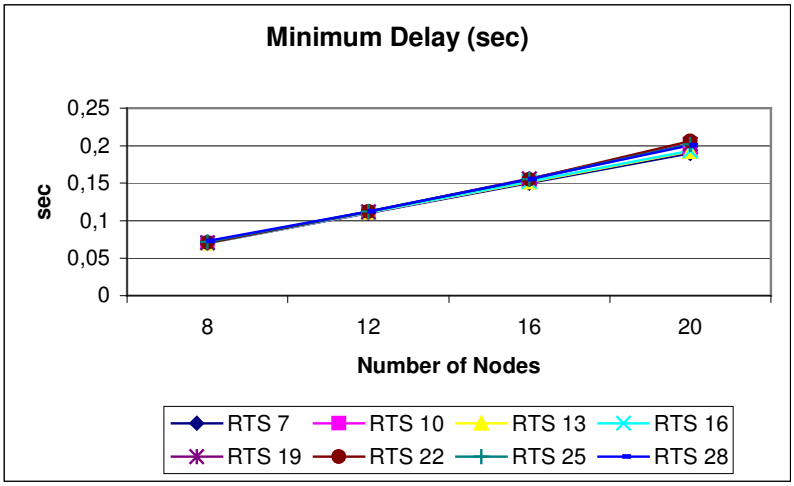


Figure 3.12: Minimum delay measurements for different RTS retry limits in 8, 12, 16 and 20 nodes string topology.

Figure 3.13 is provided to show maximum delay experienced in each simulation configuration. Maximum delay is selected as the maximum delay seen in all repeat of same configuration. As stated before, AODV algorithm tries to find route locally if the node is closer to the destination. In that case the node does not drop packets in its buffer and tries to establish route. This route establishment may take long time and if route is reestablished at the end, packets in the queue are sent to the destination. It was also shown that nodes close to destination experiences route errors when RTS retry limit is not enough to gain access to channel. This explains why the highest maximum delay is experienced when RTS retry limit is seven.

Secondly, it should be expected to see increase in the maximum delay as the node numbers increases. However, RTS retry limit 10, 13, 16, 19 and 22 on 12, 16 and 20 Node topologies results causes a conflict. The explanation can be setting RTS Retry Limit to these values is not enough to prevent packet drops and route reestablishment in these configurations. This also shows that there can be route failures even if RTS retry limit is set to 19 and 22. On the other hand, setting RTS Retry Limit to 25 and 28 totally prevents route failures, thus they have very good maximum delay performance when compared to other configurations.

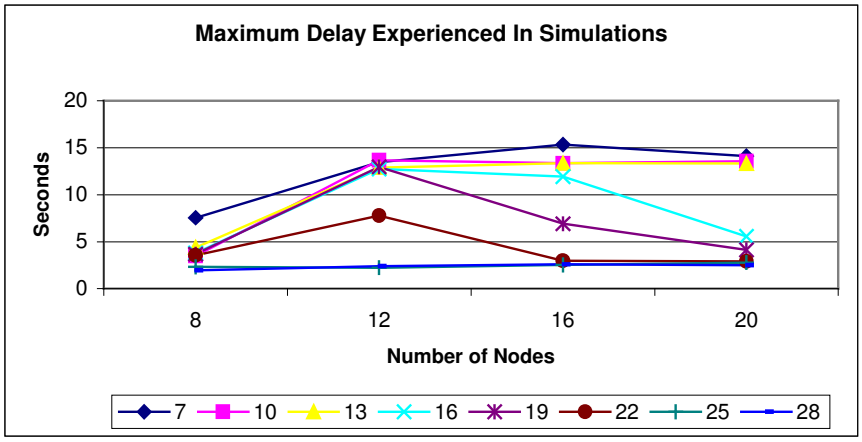


Figure 3.13: Maximum delay measurements for different RTS retry limits in 8, 12, 16 and 20 nodes string topology.

Figures 3.14 to 3.17 are provided to show the delay distribution on percentage of packets for 8, 12, 16 and 20 node topologies. In each figure, it is seen that as the RTS retry limit increases, peak point of distribution curve moves and this movement means increase on average delay. Similar observation can be done as the number of nodes increases. One can say again that RTS-19, RTS-22, RTS-25 and RTS-28 have very close delay distribution.

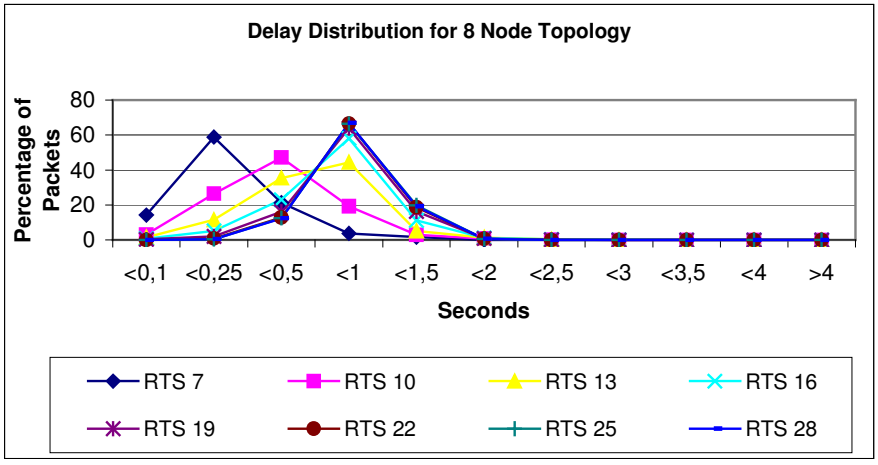


Figure 3.14: Delay distribution for different RTS retry limits in 8-node string topology.

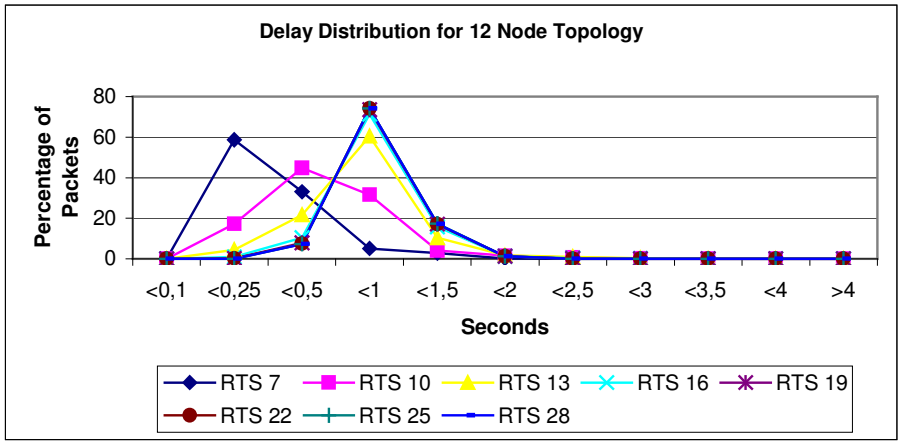


Figure 3.15: Delay distribution for different RTS retry limits in 12-node string topology.

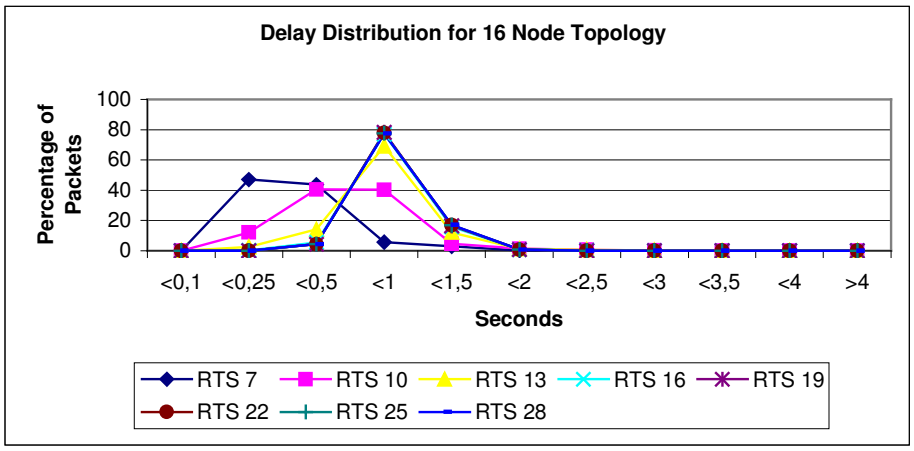


Figure 3.16: Delay distribution for different RTS retry limits in 16-node string topology.

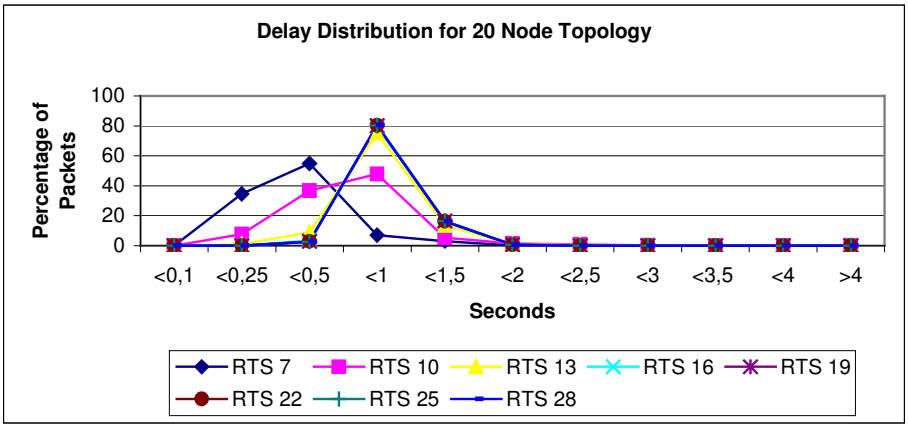


Figure 3.17: Delay distribution for different RTS retry limits in 20-node string topology.

3.2.3. Jitter Measurement for Different RTS Retry Limit

In this section, measurements of jitter for 8, 12, 16 and 20 Node topologies are provided. Simulations run for goodput measurement are also used in jitter measurements. Jitter is calculated as the variance of arrival time difference of packets at destination node. In this measurement, packet order is not considered and duplicated packets are taken into account.

Figure 3.18 provides the jitter measurement for each configuration. It is seen that jitter decreases as the RTS Retry limit increases. As stated before, increasing RTS retry limit, reduces the number of packet drops and number of route breaks. This also decreases the packet inter arrival time in the destination. It can be said again that increasing RTS retry limit above 19 does not make major changes on jitter.

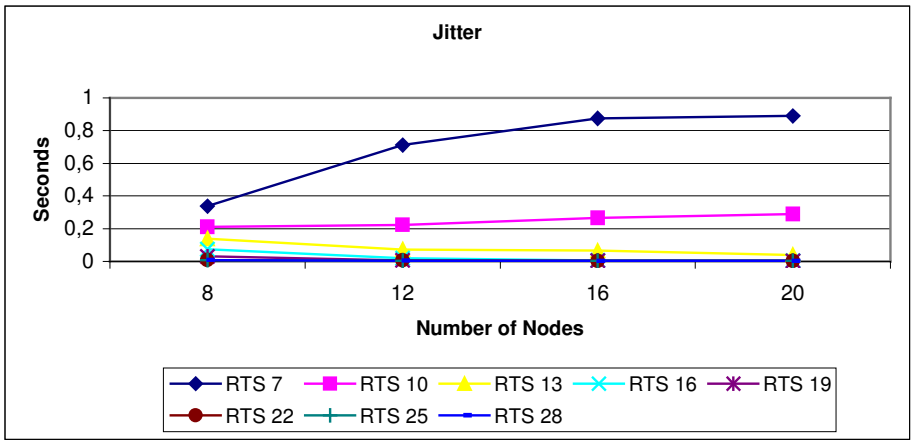


Figure 3.18: Jitter measurement for 8, 12, 16 and 20 node topologies with different RTS retry limits.

From these simulation results it is shown that increasing RTS retry limit above seven has better effect on goodput and jitter but makes average delay worse. One can say to set RTS retry limit above seven, however constantly setting RTS retry value to large values may reduce the performance in other conditions which are not considered yet.

4. ADAPTIVE ALGORITHM FOR ADJUSTING RTS RETRY LIMIT

In previous section, it was shown that increasing RTS Retry Limit results better performance in terms of goodput and Jitter. In this section, an adaptive algorithm to adjust RTS retry limit according to the network load is provided with NS2 simulation results.

4.1. Definition of Adaptive Algorithm

As shown before, packets are dropped when the nodes exceeds RTS retry limit. This happens when source node generates burst traffic that overloads the nodes. The main idea is to detect burst traffic and increase RTS retry limit before nodes exceeds the RTS retry limit. This increase on RTS retry limit will prevent the packet drops and link breaks when the network is overloaded and as a result it will improve the performance.

A node can sense the incoming traffic by listening CTS messages that are not directed to it self. Figure 4.1 is given to explain the situation. As seen in the figure, node-C is trying to send data message to node-D. When node-D sends CTS reply to node-C, this message is also received by node-E. This informs node-E that there is a communication in its neighbor node. So, frequency of CTS messages that node-E hears from node-D may signal the load of incoming traffic. In the case of loaded traffic, node-E should increase its RTS retry limit to gain access to channel and to not face retry count exceed problem after it receives packets from node-D. According to this observation, it is meaningful to measure inter arrival time of CTS messages in each node and adjust RTS retry value according to measured time.

Algorithm provided in figure 4.2 is added in to NS2 source. Algorithm first checks the destination of CTS message. If CTS message is not directed to related node, algorithm starts to calculate RTS retry limit.

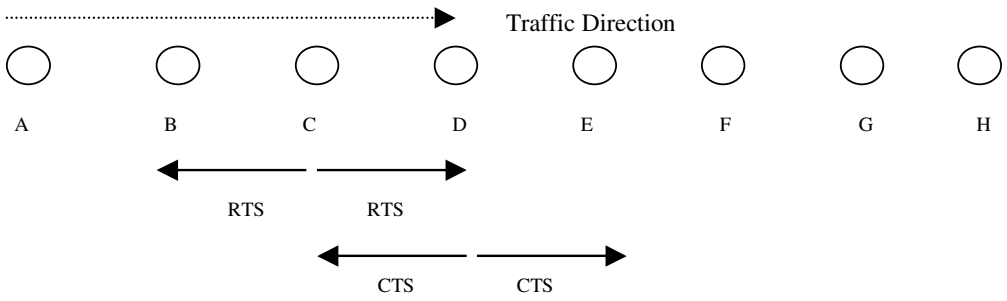


Figure 4.1: Traffic direction and CTS messages.

Algorithm calculates the inter arrival time difference between two consecutive CTS packets. If this value is greater than Max_Diff_Time, it decreases the value of RTS retry limit by subtracting RTS_DEC_VALUE, otherwise RTS retry limit is increased by adding RTS_INC_VALUE. Max_Diff_Time parameter is the threshold value that is used to decide whether there is burst traffic is coming or not. In previous sections, it was shown that increasing RTS retry limit above 19 does not make major changes, for this reason RTS retry value is limited with the MAX_ALLOVED_RTS_TRIAL parameter. Also decrease on RTS retry value is limited by the MIN_ALLOVED_RTS_TRIAL parameter.

```
if(packet_type == CTS && destination != index)
{
   delta = now - previous_time;
   if(delta < Max_Diff_Time)
   {
      rts_value = rts_value + RTS_INC_VALUE;
   }
   else
   {
      rts_value = rts_value + RTS_DEC_VALUE;
   }
   previous_time = now;
   if(rts_value > MAX_ALLOVED_RTS_TRIAL)
      rts_value=MAX_ALLOVED_RTS_TRIAL;
   if(rts_value < MIN_ALLOVED_RTS_TRIAL)
      rts_value=MIN_ALLOVED_RTS_TRIAL;
}
```

Figure 4.2: Pseudo code of implemented adaptive algorithm for adjusting RTS trial limit.

In all simulations, MAX_ALLOVED_RTS_TRIAL value is set to 25 and MIN_ALLOVED_RTS_TRIAL is set to 7 which is the default value of IEEE 802.11 standard. For Max_Diff_Time, RTS_INC_VALUE and RTS_DEC_VALUE

parameters, several simulations are done and results are provided in following sections.

4.1.1. Selection of Max_Diff_Time Parameter

Max_Diff_Time parameter is very important on the performance of proposed algorithm because it is used to decide whether to increase or decrease the RTS retry limit according to the packet traffic in network.

As stated before, only one of four nodes can send a packet due to the interference ranges of transmitted signals. According to the simulation configuration used through this work, each packet transmission between two nodes takes almost 0.01 seconds. So it takes 0.04 seconds for a packet to move 4 hop away. For this reason, if a node receives packets above 0.04 seconds interval, it can be said that network is not loaded. To see the correctness of this assumption, several simulations are done using UDP traffic. In these simulations, CBR traffic is generated on 12-Node string topology. Bit Rate and Packet size are set to 1Mbps and 1000Byte respectively. To simulate the load in network, interval time between consecutive packets is changed from 0.01 to 0.1 seconds. Results are provided in figure 4.3.

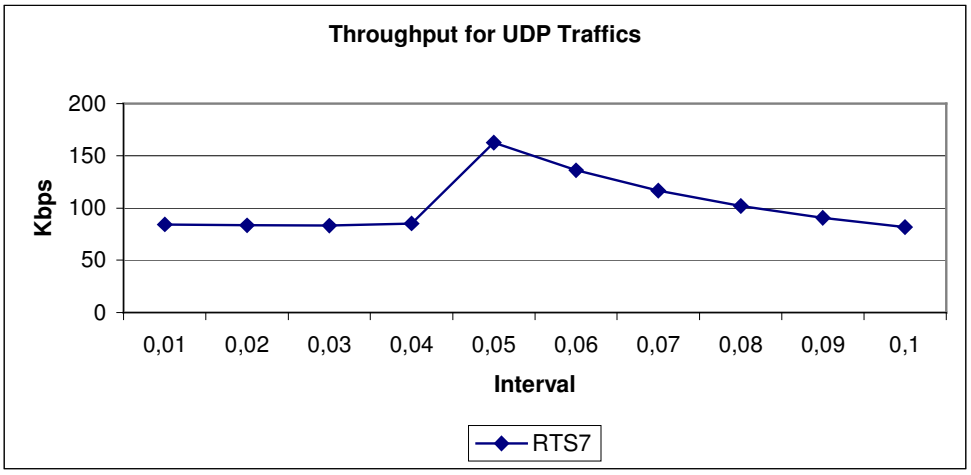


Figure 4.3: Throughput of UDP traffics in 12-node string topology.

This simulation result shows the correctness of the calculations. It is seen that maximum throughput is achieved when the interval is 0.05 seconds. Actually, maximum value should be lying between 0.04 and 0.05. Since there is a delay in each node due to packet processing, packet interval of 0.04 has the same throughput

as lower interval values. For this reason, Max_Diff_Time is set to 0.05 in all simulations of adaptive algorithm.

4.1.2. Selection of RTS_INC_VALUE and RTS_DEC_VALUE

RTS_INC_VALUE and RTS_DEC_VALUE are used to increase and decrease the RTS retry limit. In this section, effect of these parameters on goodput is investigated in 8, 12, 16 and 20 node topologies. In these simulations, only one FTP source is used and simulations are run 3000 seconds. For each topology RTS_INC_VALUE and RTS_DEC_VALUE are changed from 1 to 5 and repeated several times. Figure 4.4 to figure 4.7 are provided to show the change of goodput in 8, 12, 16 and 20 node string topology.

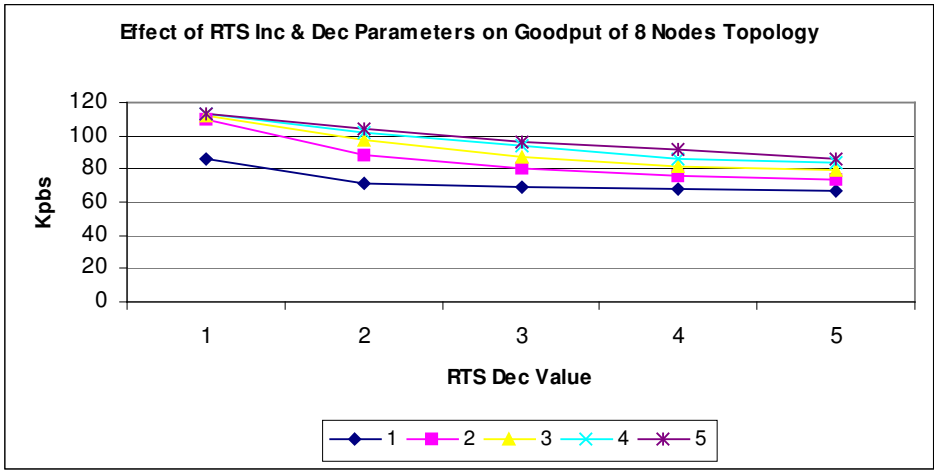


Figure: 4.4: Change of goodput according to RTS_DEC_VALUE and RTS_INC_VALUE in 8-node string topology.

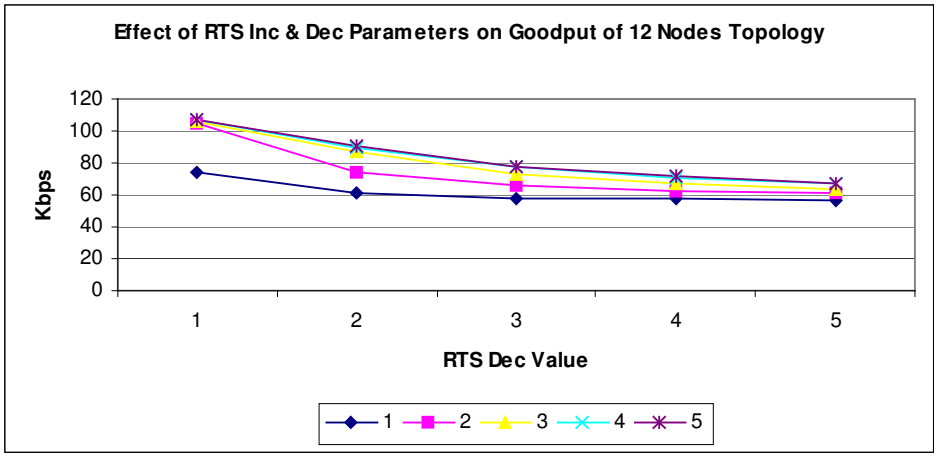


Figure: 4.5: Change of goodput according to RTS_DEC_VALUE and RTS_INC_VALUE in 12-node string topology.

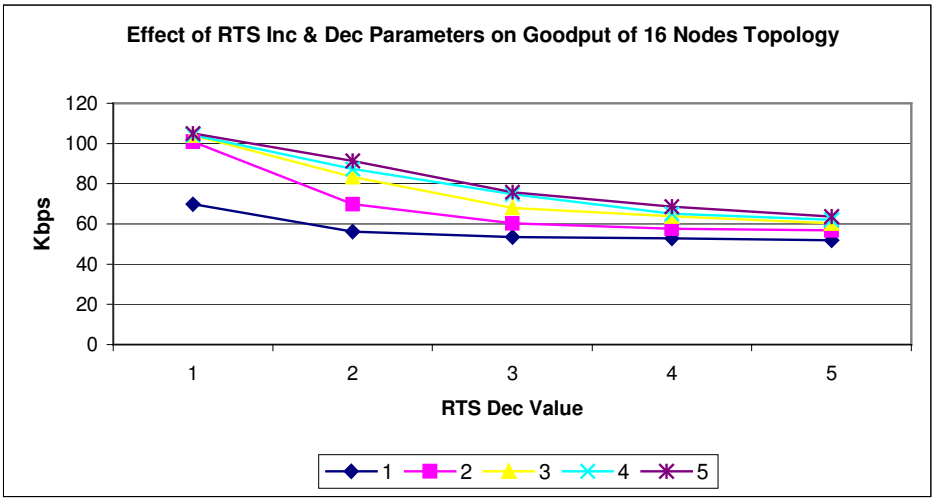


Figure: 4.6: Change of goodput according to RTS_DEC_VALUE and RTS_INC_VALUE in 16-node string topology.

Simulation results show similar behavior for each topology. It is seen that goodput decrease as the RTS_DEC_VALUE increases. On the other hand, goodput increases with the increase in RTS_INC_VALUE as expected. For each RTS_INC_VALUE, maximum goodput is achieved when RTS_DEC_VALUE is one. So, RTS_DEC_VALUE is selected as one in all following simulations. When RTS_DEC_VALUE is selected as one, values of RTS_INC_VALUE above one have close results. For this reason, RTS_INC_VAL is selected as three in all following simulations.

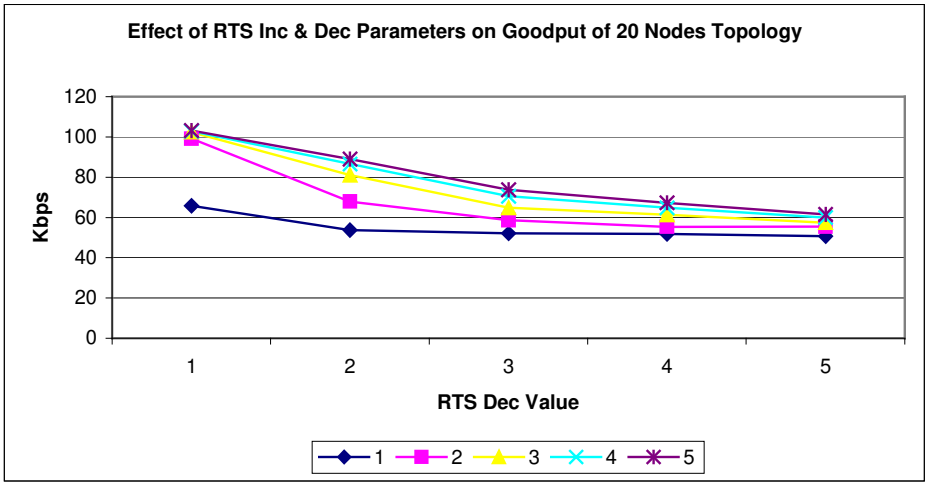


Figure: 4.7: Change of goodput according to RTS_DEC_VALUE and RTS_INC_VALUE in 20-node string topology.

4.1.3. Change of RTS Retry Limit

Figure 4.8 is provided to show the change of RTS Retry Limit. Figure 4.8 is obtained from one 200 seconds simulation of 8 Node string topology. Measurement is done at fifth node of topology. It is seen that RTS retry limit changes its value according to the load in network. In some cases, source decreases its transmission due to the packet drops and RTS retry limit decreases down to 7.

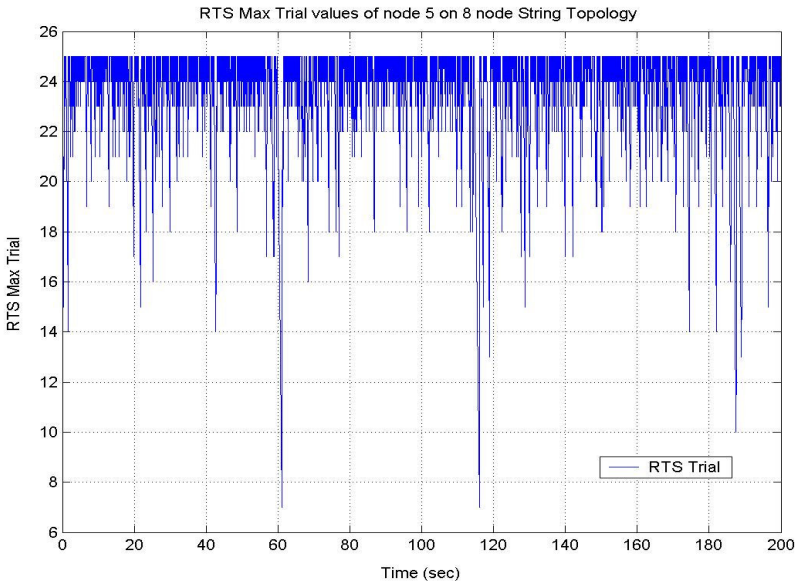


Figure 4.8: RTS retry limit change according to adaptive algorithm at 5th node of 8-node string topology.

4.2. Simulation Results of Adaptive Algorithm

In this section, simulation results of adaptive algorithm are provided. At first three subsections, goodput, Delay and Jitter measurements are given. Simulations in different topologies follow the basic measurement results. During these simulations, Max_Diff_Time is set to 0.05 seconds; RTS_INC_VALUE and RTS_DEC_VALUE values are set to 3 and 1 respectively.

4.2.1. Goodput Measurement of Adaptive Algorithm

Figure 4.9 is provided to compare goodput measurement of FTP connection when RTS retry limit is fixed to seven and adaptive. As seen in the figure 4.9, for all simulation topologies, goodput is increased with adaptive algorithm.

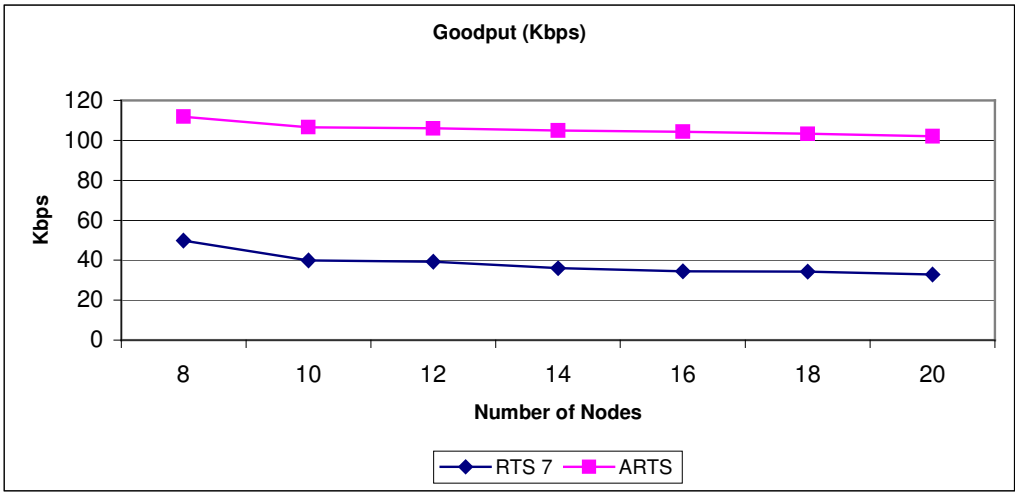


Figure: 4.9: Goodput measurement for adaptive algorithm and RTS-7.

4.2.2. Delay Measurement of Adaptive Algorithm

Figure 4.10 shows measured values of the average delay when RTS retry limit is set to seven and adjusted using adaptive algorithm. Average delay increases with adaptive algorithm as expected from previous simulation results.

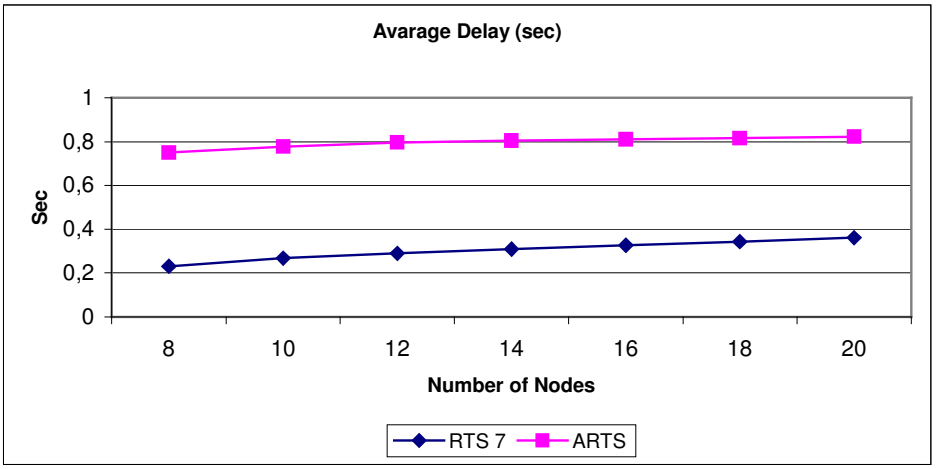


Figure 4.10: Average delay comparison of adaptive algorithm and RTS-7.

Minimum delay measurement is given in figure 4.11. As seen in these results, minimum delays are closed to each other's as observed in previous simulations. Minimum delays are obtained as the minimum of all simulation runs.

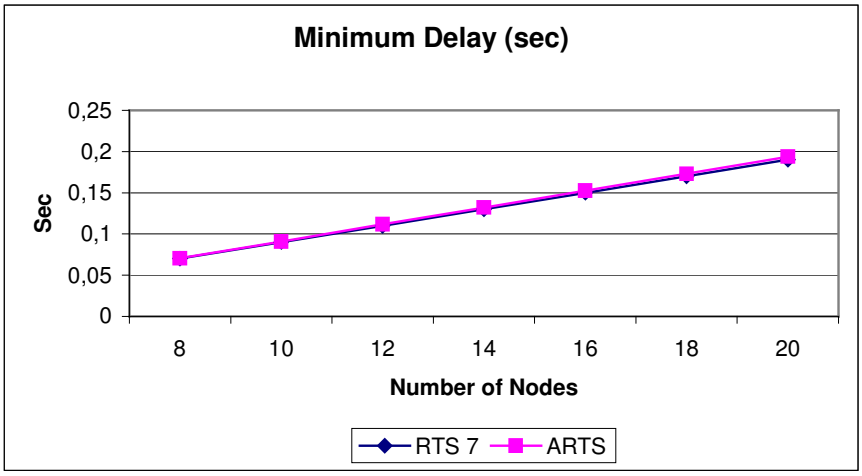


Figure 4.11: Minimum delay comparison of adaptive algorithm and RTS7.

Figure 4.12 is provided to compare maximum delay experienced by packets when RTS retry limit is fixed to seven and adaptive. As seen in the figure, adaptive algorithm and RTS-7 results are very close. As stated before, reason for this long delays are due to the AODV local route recovery feature. Secondly, maximum delay values of adaptive algorithm show that, even it is able to decrease the number of packets drops and route failures, there are still some cases that route failures exists even if adaptive algorithm is used.

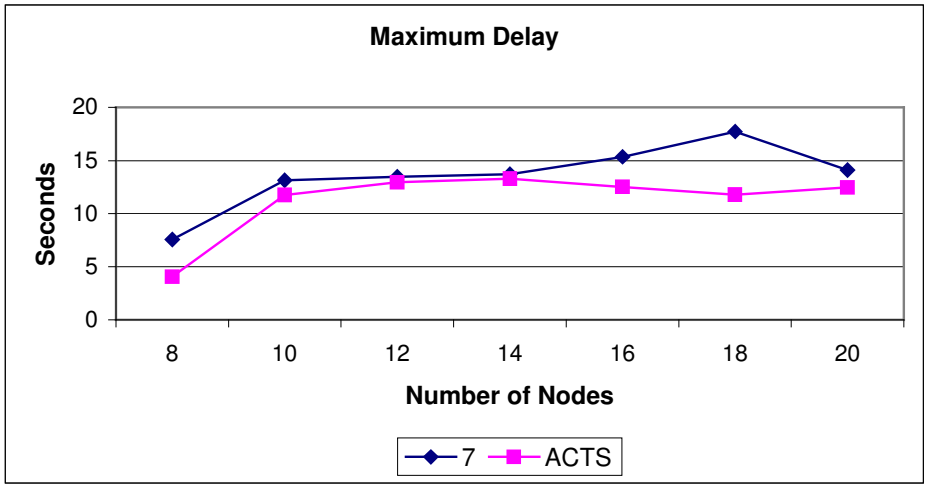


Figure 4.12: Maximum delay comparison of adaptive algorithm and RTS7.

Figure 4.13 to figure 4.16 are provided to show the delay distribution of packets in 8, 12, 16 and 20 node string topologies. As seen from adaptive algorithm results, highest percentage of packet experiences the delay between 0,5 to 1 seconds. On the other hand, peak of RTS-7 moves as the number of nodes increases.

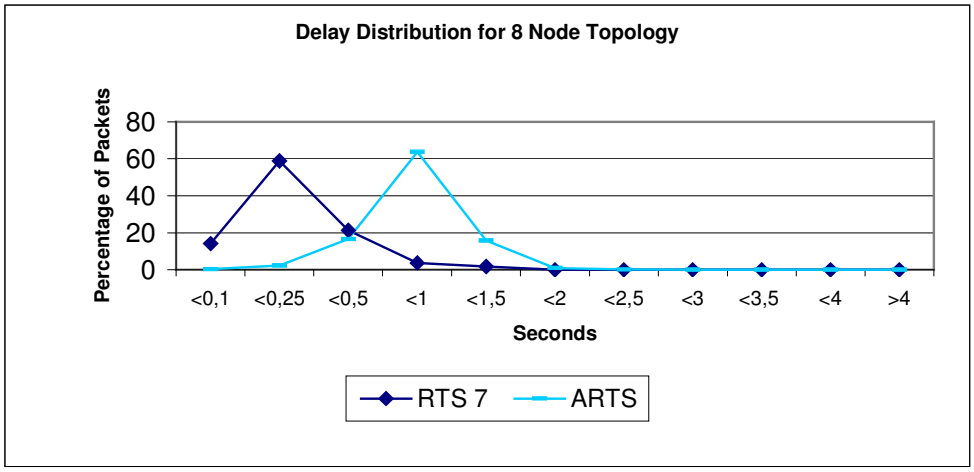


Figure 4.13: Delay distribution of adaptive algorithm and RTS7 in 8-node string topology.

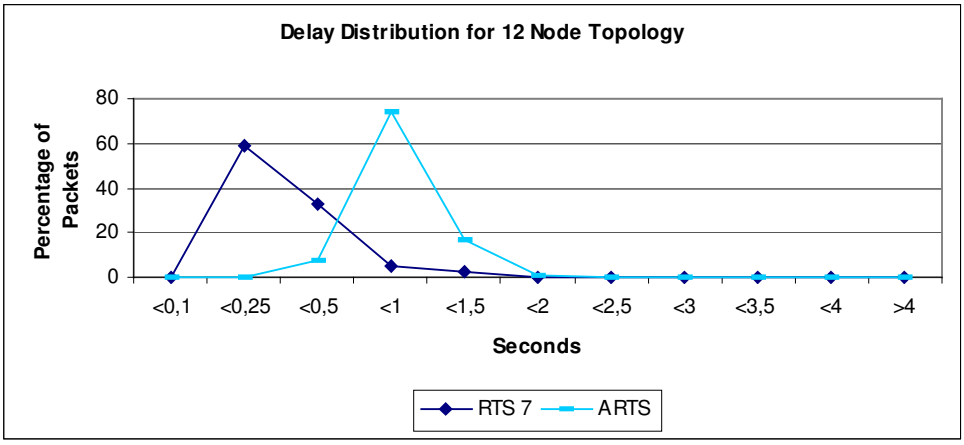


Figure 4.14: Delay distribution of adaptive algorithm and RTS7 in 12-node string topology.

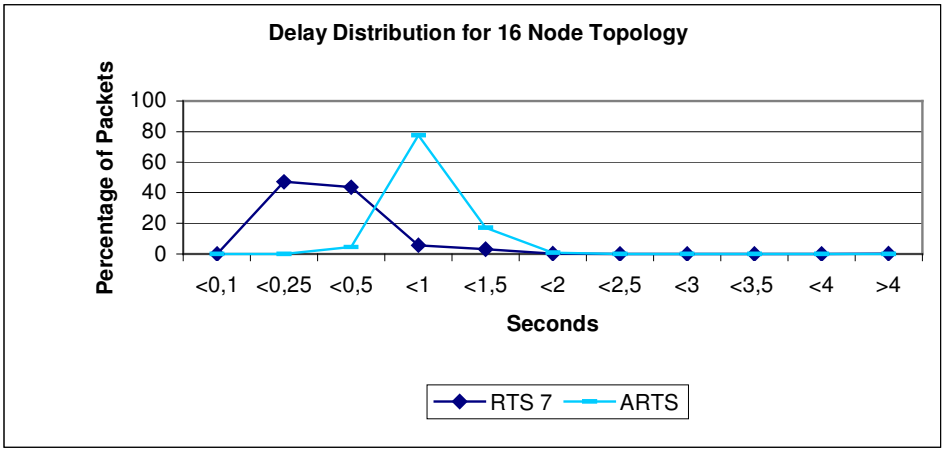


Figure 4.15: Delay distribution of adaptive algorithm and RTS7 in 16-node string topology.

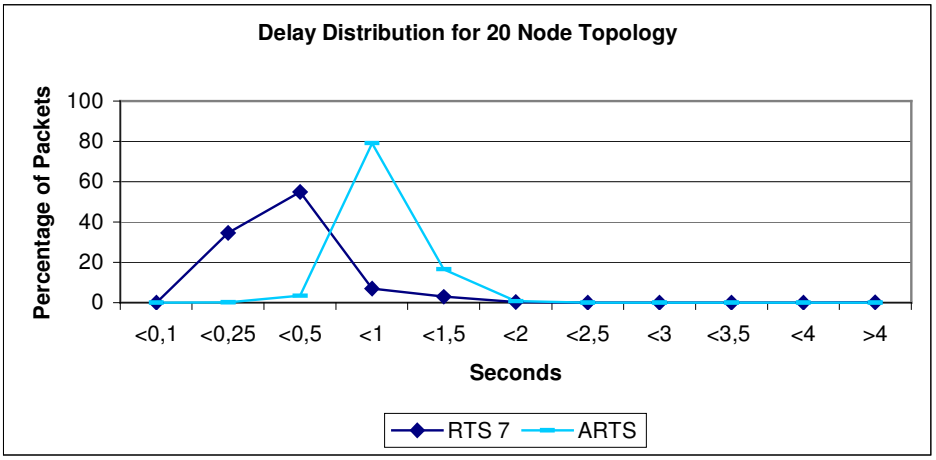


Figure 4.16: Delay distribution of adaptive algorithm and RTS7 in 20-node string topology.

4.2.3. Jitter Measurement of Adaptive Algorithm

Figure 4.17 compares the jitter measurements when RTS retry limit is fixed to seven and adaptive. This result shows us that jitter is improved with the adaptive algorithm as expected from the previous simulation results.

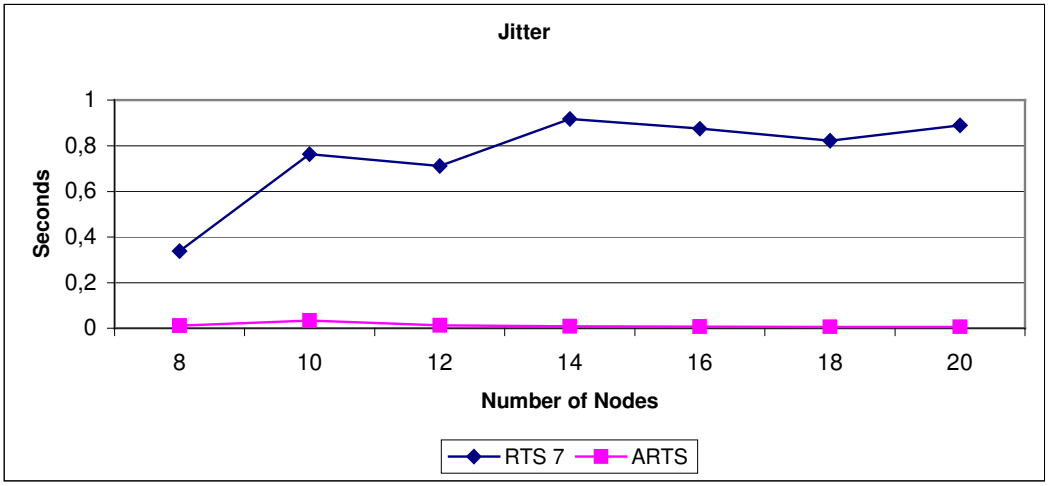


Figure 4.17: Jitter measurement of adaptive algorithm and RTS7.

4.2.4. Two Parallel String Topology

Goodput performance of adaptive algorithm is tested on a topology composed of two parallel string topologies. Simulation topology is given in figure 4.18. In this topology, nodes are placed 200 m away from each neighbor node. Each string has 12 nodes and totally there are 24 nodes in simulation. Two independent FTP traffic is generate by Source 1 and Source 2 to the Destination 1 and Destination 2 respectively.

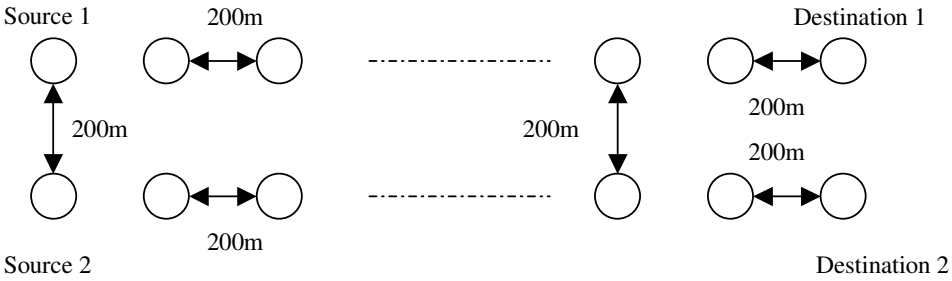


Figure 4.18: Node placement of two parallel string topology.

This simulation is run 10 times and results in table 4.1 are obtained. This results shows that adaptive algorithm performs well in this simulation topology. Also these results show that adaptive algorithm does not suffer from the channel capture problem. Channel Capture problem exits if one of the connections suppresses the other connection and uses the available bandwidth. This is an unfairness problem seen in wireless networks. Figure 4.19 is provided to compare average goodput of 10 simulation runs.

Table 4.1: Measured goodput values of each connection in two parallel string topology for adaptive RTS and RTS-7.

	Connection-1 (Goodput Kbps)	Connection-2 (Goodput Kbps)	Total (Goodput Kbps)
RTS-7	25,152	24,464	46,616
RTS-7	21,178	23,874	45,053
RTS-7	24,08	22,208	46,28
RTS-7	22,408	22,829	45,237
RTS-7	22,128	23,48	45,608
RTS-7	22,101	25,25	47,352
RTS-7	22,712	23,586	46,298
RTS-7	24,664	24,685	49,349
RTS-7	22,448	25,944	48,392
RTS-7	23,9927	20,65	43,706
ARTS	44,44	49,349	93,789
ARTS	49,141	47,834	96,976
ARTS	53,816	42,768	96,584
ARTS	47,701	47,504	95,205
ARTS	48,926	45,661	94,624
ARTS	49,757	47,442	97,2

ARTS	51,32	47,04	98,36
ARTS	45,613	51,384	96,997
ARTS	54,509	42,208	96,717
ARTS	47,501	50,757	98,258

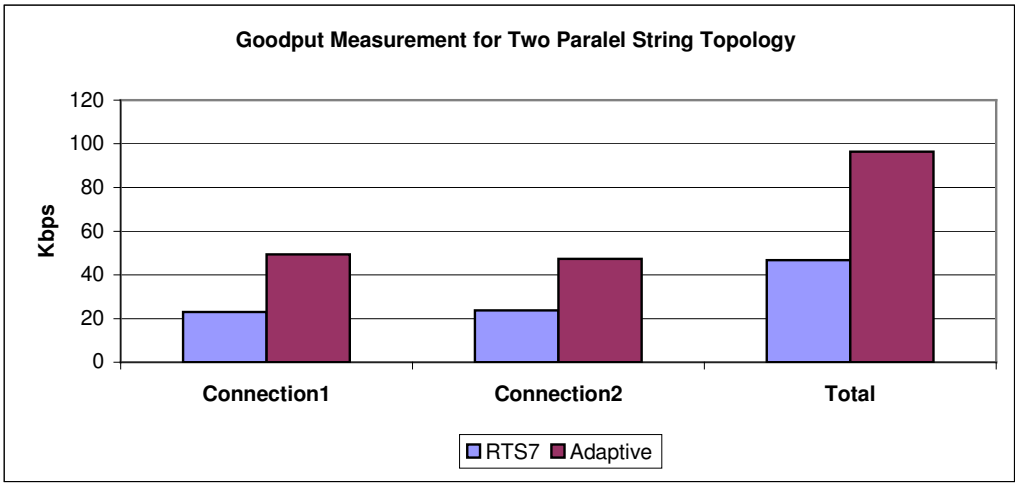


Figure 4.19: Comparison of average goodput values of connections in two parallel string topology for adaptive RTS and RTS7.

4.2.5. Cross Topology

Cross-topology is also used to test the performance of adaptive algorithm. In cross-topology, two-string topologies are connected to each other via their 6th node. Every node is placed 200m away from each other. Two independent FTP connections are created during the simulation. Simulation is run for 3000 seconds and repeated 10 times. Results in table 4.2 are obtained at the end of simulations.

In table 4.2, it is seen that total goodput increase when the adaptive algorithm is used. On the other hand, adaptive algorithm suffers from channel capture. In some simulations, one of the connections of adaptive algorithm captures the channel and suppresses the other connection. In figure 4.21, averages of 10 simulations are provided.

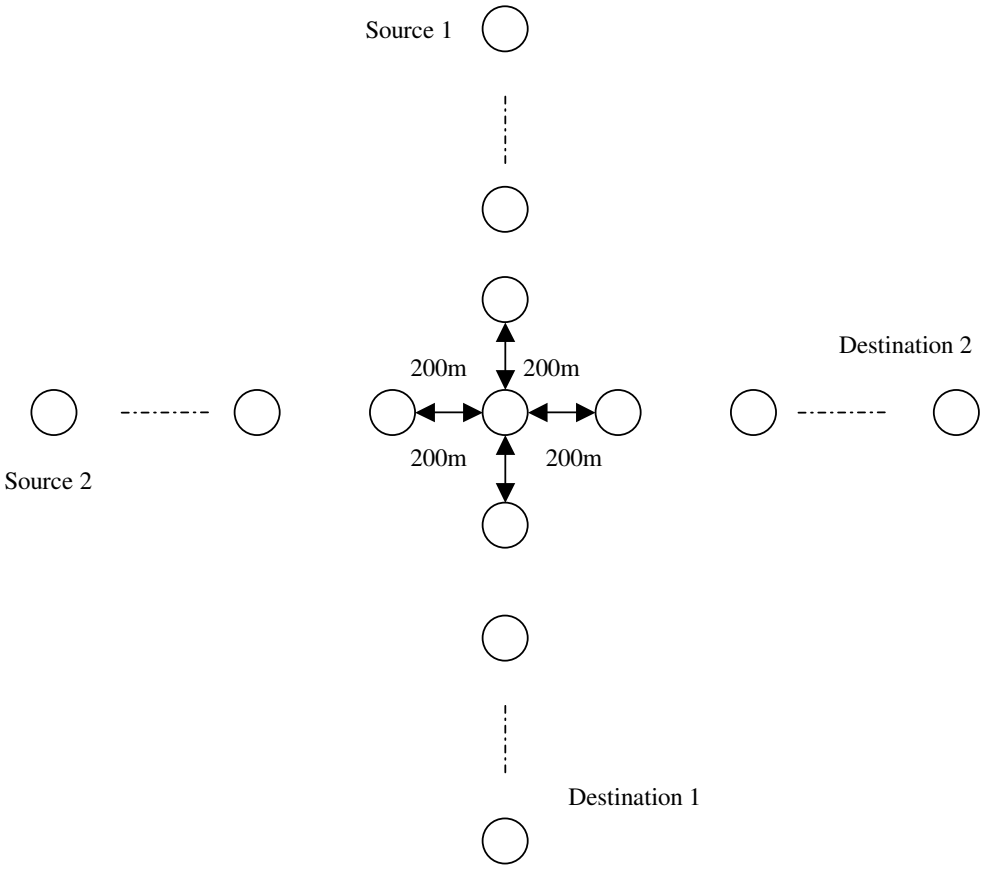


Figure 4.20: Placement of nodes in cross-topology.

Table 4.2: Measured goodput values in cross-topology for adaptive algorithm and RTS-7.

	Connection-1 (Goodput Kbps)	Connection-2 (Goodput Kbps)	Total (Goodput Kbps)
RTS-7	27,377	27,489	54,866
RTS-7	22,691	33,097	55,789
RTS-7	25,731	33,695	59,426
RTS-7	29,105	30,937	60,043
RTS-7	30,433	25,883	56,317
RTS-7	29,209	27,073	56,282
RTS-7	29,252	24,686	53,938
RTS-7	27,568	30,638	58,237
RTS-7	31,249	25,987	57,237
RTS-7	26,841	31,697	58,538
ARTS	45,138	60,28	105,418

ARTS	47,802	56,808	104,61
ARTS	102,447	4,314	106,762
ARTS	46,092	61,08	107,173
ARTS	8,747	96,722	105,469
ARTS	0,402	105,572	105,975
ARTS	105,591	0,989	106,581
ARTS	67,856	38,735	106,594
ARTS	63,117	44,77	107,888
ARTS	17,976	88,585	106,562

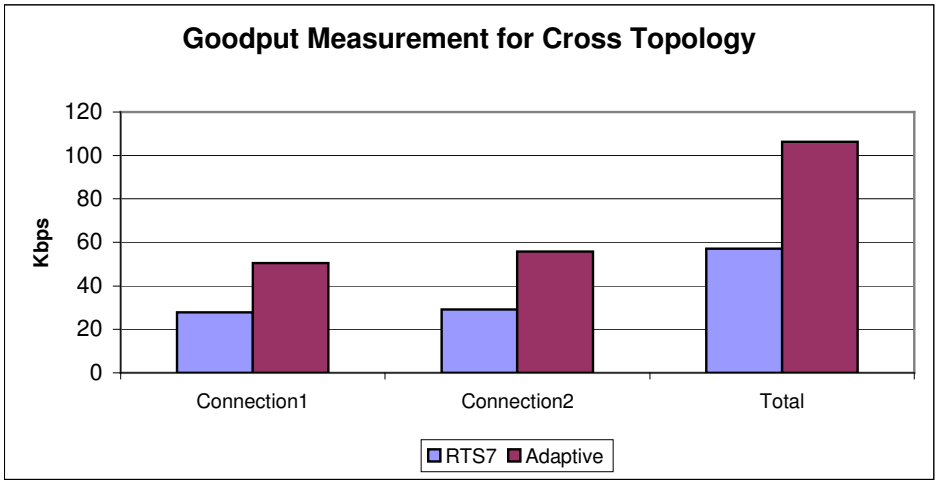


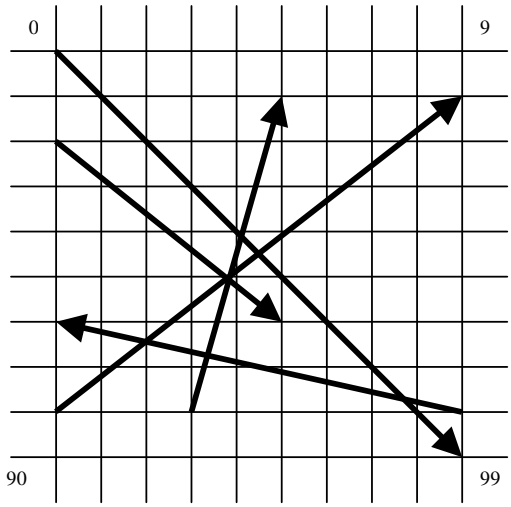
Figure 4.21: Comparison of average goodputs of adaptive algorithm and RTS7 in cross-topology.

4.2.6. Mesh Topology

In this simulation, 10x10 nodes Mesh topology is created and nodes are placed 200m apart from each neighbor nodes. Random node movement feature is also enabled in NS2 simulation configuration. But it must be stated that same seed is used for these simulations. That means random movement of nodes in each simulation is identical. Five FTP connections are created and simulations are run for 20000 seconds. Simulation topology is given in figure 4.22. Results provided in Table 4.3 are obtained from these simulations.

Table 4.3: Measured goodput values in 10x10 Mesh topology.

	Con.-1 Goodput Kbps	Con.-2 Goodput Kbps	Con.-3 Goodput Kbps	Con.-4 Goodput Kbps	Con.-5 Goodput Kbps	Total Goodput Kbps
RTS-7	12,075	20,945	15,818	11,196	15,911	75,947
ARTS	20,607	45,526	20,994	18,225	41,625	146,978



Connection	Source Node	Destination Node
1	0	99
2	20	65
3	83	15
4	80	19
5	89	60

Figure 4.22: Simulated 10x10 Mesh topology.

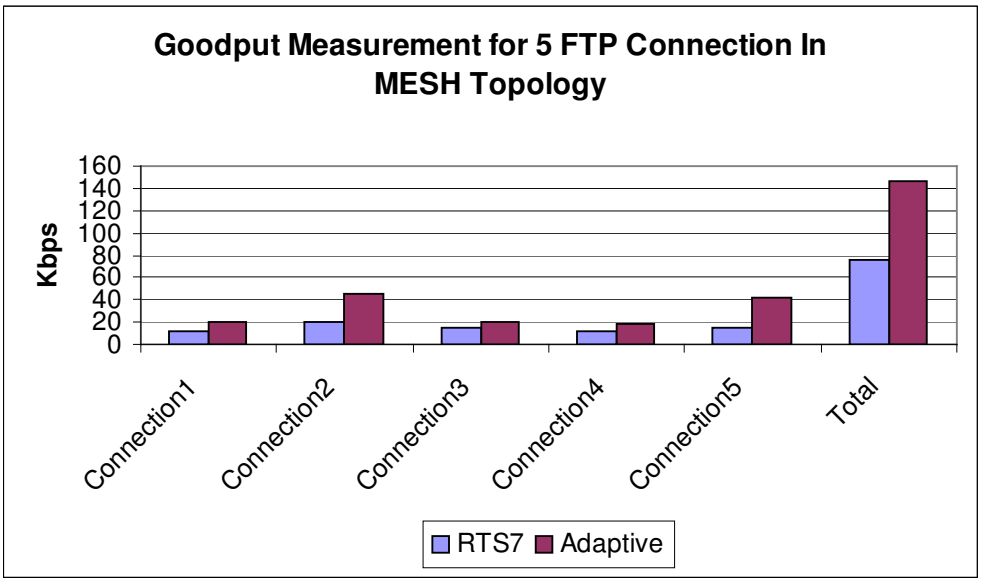


Figure 4.23: Goodput measurement of 5 FTP connections in 10x10 Mesh topology.

It is seen in figure 4.23 that setting RTS retry limit with adaptive algorithm results better performance in all five connections then setting RTS retry limit to seven. Also total Goodput is almost doubled with adaptive algorithm.

4.2.7. Performance of Adaptive Algorithm in UDP Traffics

12-Node string topology is used to measure the performance of adaptive algorithm on UDP traffics. In simulation configuration, CBR traffic with 1Mbps rate and 1000 Bytes packet size is generated. Packet interval time is changed to create different loads. Figure 4.24 provides the results of simulations. Throughput is calculated as the sum of the bytes received in destination node.

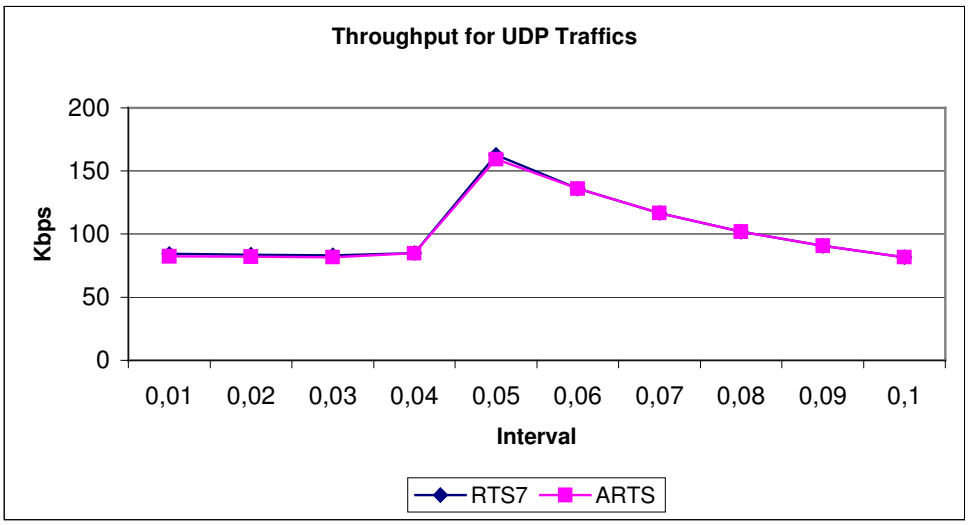


Figure 4.24: Goodput measurement on 12-Node string topology obtained by changing interval time between consecutive UDP packets.

It is seen that setting RTS retry limit adaptively has similar results as setting RTS retry limit to seven. It is expected to see an improvement when adaptive algorithm is used. However, there is no improvement because, in adaptive algorithm node increases RTS retry limit when it receives the CTS message that is not directed to it self. For this reason, first two nodes never get the CTS message of any node due to the one-way traffic direction. This weakness in first two nodes reduces the packet injection in to the network and most of the packets are dropped due to RTS Retry limit overflow in these nodes. As a result, adaptive algorithm performs similar performance as the RTS retry limit is set to seven.

5. CONCLUSIONS AND FUTURE WORK

During this work, it was shown with the help of the simulations that exposed terminal problem is the basic reason of performance decrease in multi-hop IEE 802.11 based wireless networks. Exposed terminal problem results nodes to exceed their RTS retry limit. As seen from the simulation results, setting RTS retry limit to seven is not enough to prevent route break and packet drops. Effect of increasing RTS retry limit is investigated during this work and it is shown that increasing RTS retry limit improves the goodput and jitter performance.

On the other hand, setting RTS retry limit parameter to large values may cause additional problems. Such as, when nodes are mobile, nodes will not be able to detect route failures as fast as when RTS retry limit is seven. For this reason, fixing the RTS retry value will not be suitable for all types of wireless configurations.

In this work, an adaptive algorithm is provided to adjust RTS retry limit according to the network load. This algorithm detects the network load by calculating inter arrival time of CTS messages which are not directed to itself. According to the inter arrival time between consecutive CTS messages, the node increases or decreases RTS retry limit. Increase, decrease and threshold parameters are selected according to simulation results to find suitable configuration.

Several simulations are done to investigate the performance of adaptive algorithm. It is seen that, setting RTS retry limit adaptively improves the goodput and the jitter performance but makes the average delay worse. Beside the basic measurements, adaptive algorithm performance is measured in two parallel strings, cross and 10x10 mesh topologies. In cross-topology simulation results, it is seen that adaptive algorithm suffers from fairness problem.

In the provided adaptive algorithm, only inter arrival time of CTS messages are used to detect the load in network and RTS retry limit is increased and decreased comparing the measured value with predefined threshold. In this work, queue sizes in

nodes are never taken in to account. As a future work, queue sizes can be taken in to the account while adjusting RTS retry limit.

This work also showed that RTS retry limit can be used to differentiate the performance of different connection. For this reason, RTS retry parameter might be used for QoS differentiation in wireless networks.

REFERENCES

[1] **Xu, S. and Saadawi, T.**, 2001. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks, *IEEE Communications Magazine,* **39(6)**, 130–137.

[2] **Sundaresan, K., Anantharaman, V., Hsieh, H.-Y. and Sivakumar**, **R.**, 2003. ATP: a reliable transport protocol for adhoc network, *ACM Mobile Ad Hoc Networking and Computing (MOBIHOC),* **3**, 64–75

[3] **Chandran, K., Raghunathan, S. and Prakash**, **S.R.,** 2001. A feedback based scheme for improving TCP performance in adhoc wireless networks, *IEEE Personal Communications*, **8**, 34–39.

[4] **Holland, G. and Vaidya, N.**, 1999. Link failure and congestion: analysis of TCP performance over mobile ad hoc networks, *ACM Mobile Computing and Networking (MOBICOM)*, **5**, 219–230.

[5] **Zhenghua, F., Haiyun, L., Zerfos, P., Songwu, L., Lixia, Z. and Gerla, M**., 2005. The impact of multihop wireless channel on TCP performance, *ACM Mobile Computing,* **4(2)**, 209 - 221

[6] **Yuki, T., Yamamoto, T., Sugano, M., Murata, M., Miyahara, H. and Hatauchi, T.,** 2004. Performance improvement of tcp over an ad hoc network by combining of data and ack packets*, IEICE Transactions on Communications,* **2,** 18-25.

[7] **Oliveria, R. and Braun**, **T.**, 2004. Dynamic Adaptive Acknowledgment Strategy for TCP over Multihop Wireless Networks, *Technical report*, **IAM-04-005**.

[8] **Huei-Jiun, J., Rubin, I. and Yen-Chang, K.**, 2003. An adaptive RTS/CTS control mechanisms for IEEE 802.11 MAC Protocol, *Vehicular Technology*, **2,** 1469 –1473.

[9] **Güneş, M., Hecker and M., Bouazizi**, **I.,** 2003. Influence of adaptive RTS/CTS retransmissions on TCP in wireless and ad-hoc networks, *Proceedings of the Eighth IEEE International Symposium on Computers and Communication*, **7,** 213-218.

[10] **Perkins**, **C.,** 1997. Ad-hoc on-demand distance vector routing, *MILCOM'97,* **4,** 13-25.

[11] Network Simulator, ns version 2.8, available at http://www.isi.edu/nsnam/ns/

[12] **Petrovic, M. and Aboelaze**, **M.,** 2003. Performance of TCP/UDP under ad hoc, *IEEE802.11 Telecommunications,***1**, 700 - 708.

[13] **Chaudet, C., Dhoutaut, D. and Lassous, I.G.,** 2005. Performance issues with IEEE 802.11 in ad hoc networking, *IEEE Computer Networking,* **43(7)**, 110 - 116

[14] **Z. Fu, X. and Meng, S. Lu.**, 2003. A transport protocol for supporting multimedia streaming in mobile ad hoc networks, *IEEE Journal on Selected Areas in Communications,* **21 (10)**, 1615–1626.

[15] **Gupta, R. and Ravishankar, C.V.**, 2003. Interactions between TCP and the IEEE 802.11 MAC Protocol, *DARPA Information Survivability Conference and Exposition Proceedings*, **1,** 273-282.

RESUME

Koray Atalay received his BS degree in Electronics and Communication and Computer Engineering from the Istanbul Technical University, Istanbul Turkey in 2002 and 2003 respectively. He is currently a MS candidate in Istanbul Technical University, Computer Engineering department. Also, he is working for Beko Electronik A.Ş. since 2004.