UNIVERSITY OF OSLO
Department of Informatics

Performance evaluation of QoS provisioning in ad hoc networks: SWAN and its enhancement

Cand Scient thesis

Shiva Kumar Sitaula

29th April 2006

Contents

Abstract

The fast adaption of IP-based communication of mobile and hand-held devices equipped with wireless interface is creating a new challenge for Quality of Service (QoS) provision. Due to the bandwidth constraint and dynamic topology of Mobile Ad hoc Networks (MANET), supporting Quality of Service (QoS) in MANETs is a challenging task. A lot of research efforts have been done on supporting QoS in Internet and other network infrastructures, but majority of them are not suitable in the MANET environment. To satisfy the quality of service requirement in wireless ad hoc networks several competing schemes have been proposed. In this thesis, we evaluate by means of simulation, the efficiency of QoS model for wireless ad hoc networks, called SWAN, with different admission control rate for real-time traffic than its researchers have purposed, and investigate the impact on aggregate real-time traffic throughput, average end-to-end real-time delay as well as analyze the effect of node mobility on average end-to-end delay and aggregate real-time throughput. SWAN uses admission controller(AC) to regulate real-time traffic flow and rate controller(RC) to regulate best-effort traffic. Based on the simulation results and analysis, we present our approach that we believe will enhance SWAN.

Chapter 1

Introduction

In this chapter we give introduction and some background information about Wireless Networks and Ad Hoc Networks as well as present our problem statement. The chapter also includes overview of the different chapters.

1.1 Wireless Networks

The concept of wireless networks incorporates several different network technologies, communication ranges and transmission bandwidths. They range from local coverage indoor networks (as IEEE 802.11) to large wide area coverage networks like third generation mobile telephone systems (UMTS for example). One common attributes of wireless networks is that they suffer from relatively low bandwidth and high bit error rates compared to wired networks. The proliferation of mobile computing and communication devices (e.g., cell phones,laptops,handheld digital devices, PDA) is driving a revolutionary change in our information society. We are moving from the Personal Computer age to the Ubiquitous Computing age in which a user utilizes, at same time, several electronics platforms through which he can access all the required information whenever and wherever needed.

The nature of ubiquitous devices makes wireless networks the easiest solution for their interconnection and, as a consequence, the wireless arena has been experiencing exponential growth in the past decade. Following are some of application scenarios of wireless networks: **(a)** Using cellular phone to check email or browse Internet **(b)**Travelers with portable computers can surf the Internet from airports,railway stations etc.

Among all the applications and services run by mobile devices, network connection and corresponding data services are without doubt the most de-

manded service by the mobile users.

Today, several standards exist for WLAN applications: 802.11(WLAN), HiperLAN, Home RF Shared Wireless Access Protocol, and Bluetooth. Among these classes of category, mobile ad-hoc network(MANET) belongs to the IEEE 802.11 (WLAN) standard, which is the main focus of thesis. The IEEE 802.11 standard specifies the over-the-air interface between a wireless client and a base station or access point. The standard also specifies the interface for connections among wireless clients. Typical communication range of WLAN lies between 100-500 meters, and is operating in the ISM (Industrial, Scientific and Medical) band.

The IEEE 802.11 standards are part of the IEEE 802 Local Area Network (LAN) standards family, and on top of the IEEE 802.11 MAC and PHY layer, we find the IEEE 802.1 Logical Link Control(LLC). The LLC layer is same for the all IEEE 802 transport technologies as shown in figure 1.1. The IEEE 802.11 standards consists of a common MAC layer and various PHY layers that use different transmission techniques and offer different bandwidth. Figure 1.2 on the next page compares the main differences between the different standards . Wide- and Metropolitan-area networks are multi-

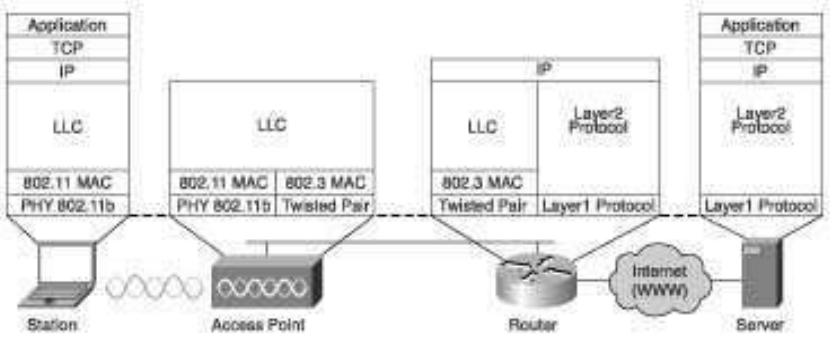

Figure 1.1: 802.11 Protocol Stack

hop based wireless networks which are not fully implemented because of challenges these networks present. On the other hand, BAN, PAN and LAN

are single-hop based networks, which constitutes building blocks of multi-hop ad hoc network.

	802.11a	802.11b	802.11g
Frequency	5 GHz	2.4 GHz	2.4GHz
Rate	54 Mbps	11Mbps	54Mbps
Market	Home	Wireless office	Home and office applications

Figure 1.2: Overview of 802.11 standards

WLAN architecture has three components:

- Wireless end stations
- Access points
- Basic service sets

The wireless end station can be any device that can communicate using the 802.11 standard (laptops, workstations, and PDAs).

The access point (AP) is a device that can provide two functions: It acts as a network platform for connections between WLANs or to a wired LAN and as a relay between stations attached to the same AP.

Whereas the wireless station and the access point are both physical components, the basic service set (BSS) is the logical component of wireless architecture. The BSS in general is a set of wireless stations controlled by a single management function and has two configuration options : **IBSS** and **BSS**.

- **IBSS** : In an IBSS, the stations communicate peer-to-peer basis without the need for an access point (AP), also known as ad hoc network, as shown in figure 1.3.

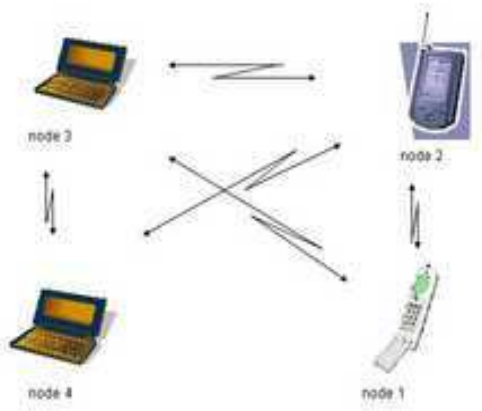

Figure 1.3: Independent mode BSS (IBSS)

- **The Infrastructure BSS** : In an infrastructure BSS, there is a connection to the wired network and all the communication between mobile nodes is passed through one access point (see figure 1.4 on the next page).

1.2 Mobile Ad Hoc Network (MANET)

A mobile ad hoc network is a concept that has received attention in scientific research. Due to the fact that term ad hoc network is used in many different ways in literature, it is difficult to give clear picture of ad hoc network.

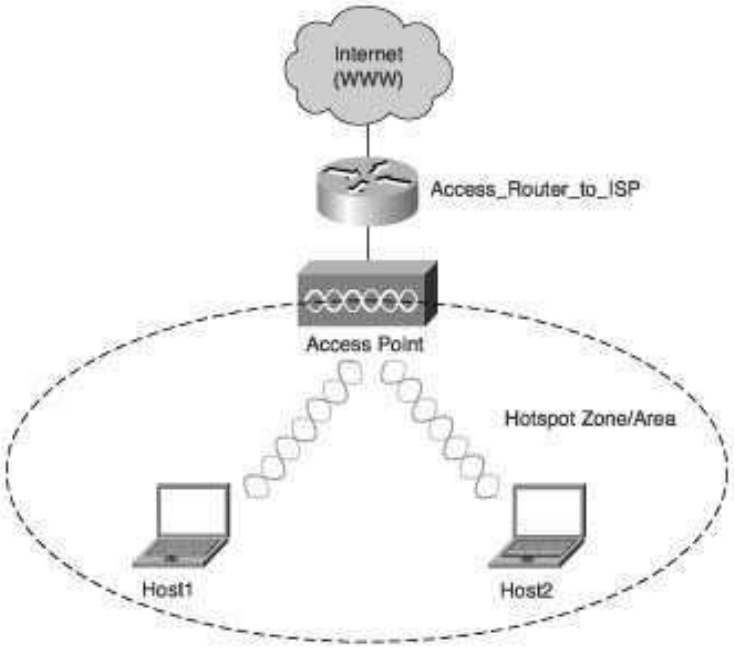

Figure 1.4: Infrastructure based mode

These networks can be formed on the fly, without requiring any fixed infrastructure. As these are infrastructureless networks, each node should act also as a router. The definition provided by the Internet Engineering Task Force(IETF) is given below:

A mobile ad hoc network (MANET) is autonomous system of mobile routers (and associated hosts) connected by wireless links. The router are free to move randomly and organize themselves arbitrarily, thus network's wireless topology may change rapidly and unpredictably. Such a network may operate in a stand-alone fashion, or may be connected to the larger Internet .

MANET paradigm could be used to efficiently extend the capacity/coverage of Wi-Fi hot spots. Spreading in a hot spot a large number of APs to guarantee the coverage is not appealing both from the economic(infrastructure cost) and technical standpoint. AP's interference and the ad hoc paradigm can possibly offer an efficient solution to this problem by upgrading APs with multi-rate high-speed technologies to achieve required coverage .

1.3 Motivation

Wireless local area networks are a viable technology to support multimedia traffic. In wireless multimedia networks, mobile stations will be capable of generating a heterogeneous traffic mix with varying bandwidth requirements. There is an increasing demand for wireless multimedia networks due to the attractiveness of providing network services to communicate using any type of media without any geographical restrictions. Therefore next generation wireless networks are expected to support multimedia services with guaranteed Quality of Service (QoS) for diverse traffic types (video, audio, and data). Wireless Local Area Networks (WLANs) have developed into a viable technology to support multimedia traffic transmission . A network's ability to provide a specific QoS depends upon the inherent properties of the network itself, which span over all the elements in the network. Characteristics of Mobile Ad Hoc network(MANET) such as lack of central coordination, mobility of hosts, dynamically varying network topology, and limited availability of resources makes QoS provisioning very challenging in such networks. Many proposals have been presented to support QoS in WLAN including MAC protocols, QoS routing protocol etc. While these proposals are sufficient to meet QoS needs under certain assumptions, none of them proposes a QoS model for mobile ad-hoc networks(MANETs).

Among the QoS model proposed for multi-hop wireless networks, SWAN[4] is one of the most promising QoS models for mobile ad-hoc wireless networks. SWAN[4] researcher has set predefined real-time bandwidth rate (so called Admission Control Rate) to provide QoS and to allow minimal delay for real-time traffic. It would be desirable to evaluate its performance under different real-time bandwidth rate than predefined by researcher in a bid to evolve it's effectiveness in MANETs. The main contribution of this thesis work is to see whether that predefined admission control rate is optimal to provide QoS for real-time traffic, and whether higher or lower admission control rate than originally purposed will contribute to enhance SWAN model .

1.4 Methodology

To answer our problem statement we had to study thoroughly different Quality Of Service (QoS) model proposed for Mobile Ad hoc networks (MANETs). The study was need to have insight into ideas, limitations and problem associated with those QoS models. We used Network Simulator,NS-2 [1] with its wireless extensions developed at CMU for our analysis. We im-

plement NS-2 with SWAN code downloaded from the SWAN web page [http://www.comet.columbia.edu/swan/simulations.html]. NS-2 is implemented with SWAN extension which includes an admission controller, a rate controller (AIMD), a probe protocol , a mechanism for packet delay mechanism and the ECN mechanism.

1.5 Outline of the thesis

Chapter 2 gives a overview of difficulties and issues on provisioning QoS support on Ad Hoc networks (MANETs). This chapter also describes existing QoS support in wired network and its unsuitability in MANET's environment .

Chapter 3 discuss the issues to be considered while designing the QoS framework for ad hoc networks. This chapter also gives overview of few QoS model proposed for ad hoc networks.

Chapter 4 discuss SWAN models for MANETs .Most part of this chapter is used to present SWAN model and its operational framework

Chapter 5 discuss the suitability of SWAN parameter or real-time bandwidth rate proposed by SWAN researcher . We compare the SWAN's performance by modifying purposed admission control rate by its researcher to see the impact of QoS through simulation .

Chapter 6 is the conclusion of this thesis. In addition to the conclusion, it holds proposal for future work that could be valuable to investigate on the subject of Quality Of Service(QoS)on ad hoc networks.

Chapter 2

QoS on ad hoc networks

In this chapter we give overview of issues and difficulties in provisioning QoS in MANETs. This chapter also provide overview of existing QoS technology for Internet and it's unsuitability in terms of MANET.

2.1 QoS issue on Ad-hoc networks

In order to facilitate QoS support in MANETs, we first need to define the metrics to quantify QoS and understand issues in provisioning QoS in ad hoc networks. As different applications have different requirements, the services required by them and the associated QoS parameters differ from application to application. Unlike traditional wired networks, where the QoS parameters are mainly characterized by the requirements of multimedia traffic, the QoS parameters in MANETs are more influenced by the resource constraints of the nodes.

QoS is usually defined as *a set of service requirement that needs to be met by the network and network is expected to guarantee a set of measurable pre-specified service attributes to users in terms of end-to-end performance like delay,bandwidth* .

For mobile ad hoc wireless networks with time-varying low-capacity resources,the notion of being able to meet specific application requirement such as delay is not plausible. Therefore, the definition may not be valid for mobile ad hoc networks since even network with fixed communication links and high speed capacity unable to deliver guaranteed end-to-end services. Hence, quality of service in mobile ad hoc network could mean to provide a set of parameters in order to adapt the applications to the "quality" of network while routing them through the network.

QoS provisioning often requires negotiation between mobile node and network, call admission control, resource reservation, and priority scheduling of packets. QoS can be rendered in MANETs by per flow, per link , or per node thus QoS support in MANETs encompasses issues at all layers of IP network infrastructure. There is not any clear definition of boundary between the service provider (network) and the user (host), thus making it essential to have better coordination among the nodes to achieve QoS . The mobile multihop wireless networks differ from traditional wired Internet infrastructures which introduce unique issue and difficulties for supporting QoS in MANETs environments.Some of the main issues are described in following section .

- **Node mobility :**

  In MANETs, nodes moves freely and do not have any restriction on mobility causing frequent changes to network topology. Hence, the admitted QoS sessions may suffer due to frequent path breaks, therby requiring such sessions to be re-established over new paths. Hence, some packets of sessions could not meet their delay target due to delay caused by session re-establishment process, which is not acceptable for applications that have stringent QoS requirements.

- **Route maintenance :**

  In most cases, the nodes in an ad hoc wireless network maintain both the link-specific state information and flow-specific state information. Especially, it is hard to maintain flow-specific state information due to dynamic nature of the network topology and channel characteristics. In ad hoc network, nodes can join and leave any time and frequent update of routing path is essential with minimal overhead and delay. Hence, routing decision may not be accurate, resulting in some of the real-time packets missing their deadlines. Also, QoS-aware routing would require reservation of resources at the intermediate nodes (router). Thus, reservation maintenance with updates in the routing path becomes cumbersome.

- **Power Constraint:** In MANETs,node's processing capability is limited because of limited battery power. This feature require low processing overheads of nodes.Therefor, the control algorithm and QoS mechanisms should keep the inter-router information exchange to a minimum.

- **Hidden and Exposed Terminal Problem :**

In wireless ad hoc networks that rely on a carrier-sensing random access protocol (**CSMA/CD**), the wireless medium characteristics generate complex phenomena such as the hidden and exposed terminal problems. The hidden-terminal (see figure 2.1) problem occurs when two (or more) stations can not detect each other's transmission but their transmission ranges are not disjoint.

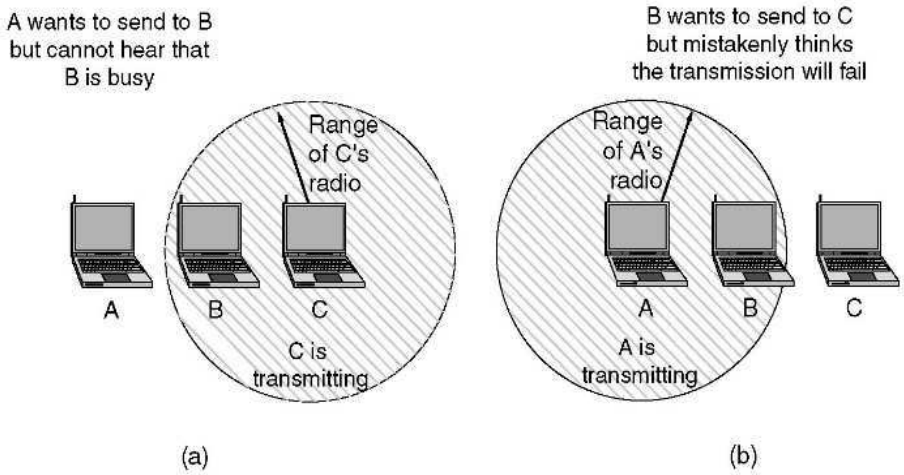

Figure 2.1: Hidden-terminal problem

As shown in fig.2.1, a collision may occure, for example, when the station **A** and station **C** start transmitting towards the same receiver, station **B** . The exposed-terminal (see figure2.2) problem results from

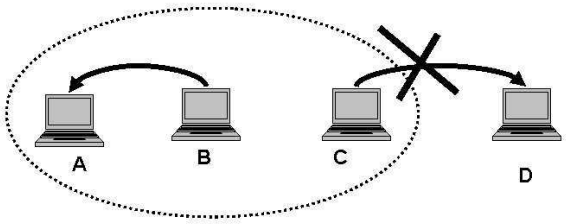

Figure 2.2: Exposed-terminal problem

situations where a permissible transmission from a mobile station to antoher station has to be delayed due to the irrelevant transmission activity between two other mobile stations within sender's transmission range.

- Security:

  Security can be considered a QoS attribute. Without adequate security,unauthorized access and usage may violate QoS negotiations. The physical medium of communication in wireless networks is inherently insecure, and a need to design securityaware routing algorithm is arises.

Among the QoS attributes, power consumption and service coverage area are are specific to MANET. Furthermore, QoS metrics could be defined in terms of one of the parameters or a set of parameters in varied proportions. Since providing QoS in a time-varying environment is very challenging, the notion of being able to guarantee hard QoS is not plausible. So, applications must adapt to time-varying low-capacity resources offered by the network. Hence, there exits mainly two different compromising principles, *soft QoS* and *QoS adaption*(dynamic QoS) provisioning in the MANETs .

Soft QoS means that after connection setup, there may exist transient periods of time when the QoS specification is not honored.Here, the degree of QoS support measured in terms of fraction of total disruption time over the total connection time.

In a **dynamic QoS**, one can allow a reservation to specify a range of values rather than a single point. With such an approach, as available resources change, the network can readjust allocations within the reservation range. Furthermore, QoS adaption can be also done at various layers .

2.2 QoS models for Internet

The potential commercial applications of MANETs require seamless connection to the Internet. Thus the QoS model for MANETs should also consider the existing QoS architectures in the Internet . In following sections , we give short descriptions of **IntServ** and **DiffServ** and their suitablility in terms of MANET environment .

2.2.1 IntServ/RSVP

The basic idea of the Integrated Service (IntServ) model is to keep flow-specific states in every IntServ-enabled router. In an IntServ-enabled router, IntServ is implemented with four componets: *the signaling protocol, the admission control routine, the classifier* and *packet scheduler*. The Resource ReSerVation Protocol(RSVP) is used as the primary signaling protocol to setup and maintain virtual connections. In additon to Best Effort Service, IntServ proposes three main catagories of services that can be provided to users: **Guaranteed Service** , **Controlled load Service** and **Best-Effort Service** . The **Guaranteed Service** is provided for fixed delay bound applications and **Controlled Service** is provided for applications requiring enhanced best effort services . **Best-Effort Service** are characterized by absence of a QoS specification and the network delivers the best possible quality . All of these service classes use RSVP for to reserve resources before transmission . The IntServ provides quantitative QoS for every flow . There is number of scenarios which makes IntServ unsuitable as QoS model for ad hoc networks as listed below :

- **Scalability**

  As IntServ provides per-flow QoS granularity, a huge storage and processing overhead results in mobile node whose storage and computing rsources are scarce. It is believed that due to the

development of fast radio technology and potential large numbers of users in future, scalability problem may occur in MANETs. One can argue that processing and storing capacity also increase, should high-speed MANETs are developed in future .

- **Signaling**

  Signaling protocol generally contain three phases : *connection establishment*, *connection maintenance* and *connection teardown*. In highly dynamic networks such as MANETs, this is no promising approach since routes may change very fast and the adaptation process of protocols using a complex handshaking mech nism would just be too slow. Furthermore, the RSVP signaling packets contend for bandwidth with the data packet and consume substantial percentage of limited available bandwidth in MANETs

2.2.2 DiffServ

DiffServ is designed to overcome the difficulty of implementing and deploying IntServ and RSVP in the Internet backbone. The model aims at moving the complexity to the edge of the network , resulting in a system which is scaleable to an arbitary extend. In DiffServ,traffic is divided into classes according to priorities and resource requirement. At the boundary of a network, the boundary routers perform admission control and classify, shape and police the traffic so that it confirms to the SLAs. The term SLA is defined as *a set of parameters and their values which together define the service offered to a traffic stream by a DS domain*. Within the core of the network, packets are forwarded according to the per-hop behaviour (PHB) associated with the DSCP (Differentiated Service Code Point). This eliminates the need to keep any flow state information elsewhere in the network, but the offered service level may vary .

The shortcomings av DiffServ in MANETs environment are described below:

- **Storage Cost & Ambigious boundary router:**

  The benefit of DiffServ is that traffic classification and conditioning only has to be done at the boundary router. This makes quality of service provisioning much easier in the boundary router. In MANETs though, there is no clear definition of what is the boundary router because every node is a potential sender, receiver and router. Due to

the diverse functionalities of a mobile node,heavy storage cost occurs in every mobile node. This drawback would again take us back to the IntServ model where several separate flow states are maintained.

- **Service Level Agreement(SLA):**

  The SLA is a kind of contract between a customer and its Internet Service Provider (ISP) that specifies the forwarding services the customer should receive. Administration of a DiffServ domain must assure that sufficient resources are provisioned to support the SLA's committed by the domain. DiffServ uses notion of SLA to support QoS in wired network. But, the concept of Service Level Agreement(SLA) in the Internet does not does not exit in MANETs. The SLA is indespensable because it includes the whole or partial traffic conditioning rules, which ar used to re-mark traffic streams, discard or shape packets according to the traffic characteristics such as rate and burst size. How to make a SLA in MANETS is difficult because there is no obvious scheme for the mobile nodes to negotiate the traffic rules .

- **Soft QoS guarantees:**

  DiffServ uses a relative-priority scheme to map the quality of service requirements to a service level. This aggregation results in a more scalable but also in more approximate service to user flow.

2.3 IntServ over DiffServ

IntServ/RSVP and DiffServ [3] can be seen as complementary technologies in the pursuit of end-to-end QoS. Together, these mechanisms can facilitate deployment of multimedia applications. IntServ enables hosts to request per-flow, quantifiable resources, along end-to-end data paths and to obtain feedback regarding admissibility of these requests. DiffServ enables scalability across large networks. The primary benefit of combining IntServ and DiffServ is the increased scalability, provided through the aggregate traffic control of DiffServ. Also terminal mobility may be more easily supported. Since both approaches (**IntServ** & **DiffServ**) have limitations and constraints in terms of scalability and the level of Qos performance assurance in MANETs environment, this approach could be effective approach on provisioning QoS in ad hoc networks .

2.4 Conclusion

Neither *IntServ* nor *DiffServ* model provide QoS assurance in MANETs environment. There is need of QoS model which take account of dynamic nature of ad hoc networks. As mentioned in previous section, one solution is to combine *IntServ* and *Diffserv*. Obviously,the necessity of more effective and efficient QoS approach in MANETs arises. In the next chapter, we present QoS protocols and QoS model proposed for ad hoc networks.

Chapter 3

QoS Model and Design Issues

In this chapter, we present overview of different existing QoS model and framework proposed for Mobile Ad Hoc networks (MANETs), and other components of network that support achieving QoS in ad hoc network. The chapter also pinpoint some of the fundamental issues to be considered while designing a qual ity of service infrastructure for MANETs.

3.1 QoS Analysis

Generally, a QoS model or infrastructure does not define specific protocol implementation. Instead , it defines the methodology and architecture by which certain type of services (e.g. per-flow or class-based) can be provided in the network. The network needs are governed by the service requirements specified by the end user applications. Due to time-varying characterstics of MANETs, providing QoS is not related to specific layer in the networks. Hence providing QoS in MANETs spans over all the layers of network, so layers must co-operate with each other to provide QoS in MANETs.

The physical layer should take care of changes in transmission quality, for example, by adaptively increasing or decreasing the transmission power. Similarly, the link layer should react to the changes in link error rate, including the use of automatic repeat-request (ARQ) technique. Such a inter-layer coordination will eventually require the QoS-aware protocol in different layers. The protocols such as routing ,resource reservation signaling and MAC must co-operate to achieve the goals outlined by the QoS model. Thus, the QoS solution can be classified in two ways. One classification is based on the QoS approach employed , while the other one classifies QoS solutions based on the layer at which they operate in the network protocol layer.

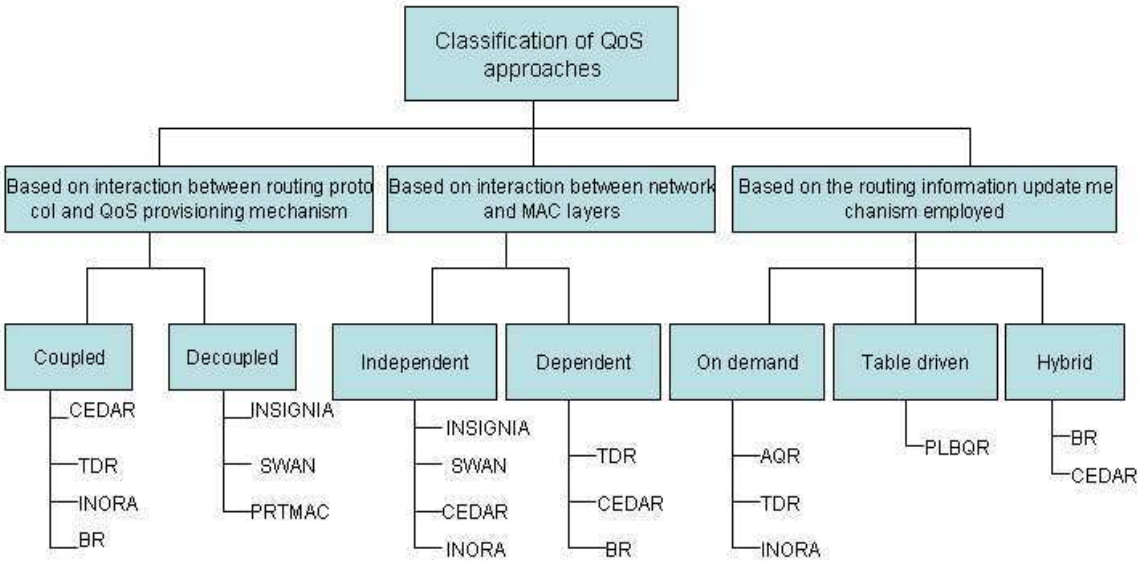

Figure 3.1: Classification of QoS approaches [22]

The QoS approach can be classified based on the interaction between the routing protocol and the QoS provisioning mechanism, as well as based on the interaction between the networks and mac layers as shown in figure 3.1. Also, QoS provision can be classified into two catagories : **Independent** and **Dependent** based on the routing protocol and MAC protocol . Under the **Dependent** QoS approach, MAC and routing protocol interact with each other to provide QoS provision, where under **Independent** QoS approach, the network layer is not dependent on the MAC layer of QoS provision.

Similarly, as shown in figure 3.2 on the next page , QoS solution can also be classified based on which layer in the network protocol stack they operate in . Under the Layer approach , many QoS-aware protocol has been designed

to provide service differentiation in MANETs paradigm. Among other, QoS MAC, QoS routing and resource-reservation signaling are important componets.

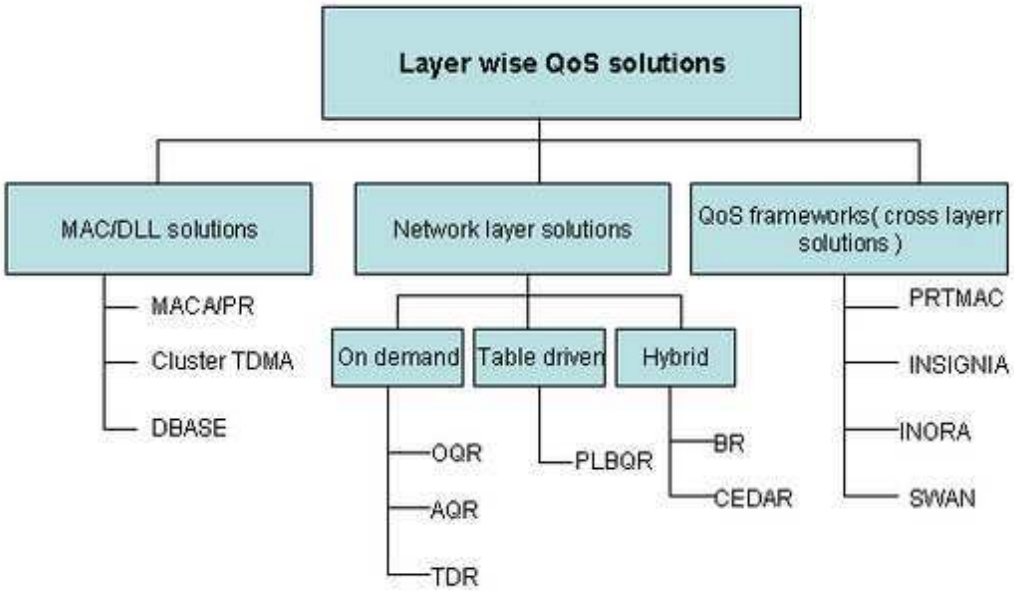

Figure 3.2: Layer-wise classification of QoS solutions [22]

The quality of service that an application requires depend strictly on the available network resources that reside both in the wireless medium and in the mobile nodes as well as the stability of these resources. Therefore, some proposed cross-layer quality of service model separates QoS metrics at the different layers [17]. The notion of QoS metrics is different in each layer depending the role each layer play to provide quality of service in MANETs environment.

INSIGNIA,**iMAQ** and **SWAN** are cross-layer QoS frameworks for MANETs.

The interaction and cooperation of protocols such as routing , resource-reservation signaling and MAC is necessary to achieve the goals outlined by the QoS model. QoS signaling acts as the control center in QoS support. It coordinates the behaviors of QoS routing , QoS MAC, and other components such as admission control and scheduling. The QoS modules should react promptly to changes in the network state and flow state . Broadly speaking, we can divide the QoS in MANETs into following:

1. QoS Model
2. QoS Resource Reservation
3. QoS Routing
4. QoS MAC

The following section discuss issues to be considered while designing a QoS framework in MANETs .

3.2 QoS-Signaling and Routing Interaction

QoS signaling and the QoS routing are closely related and may be coupled or decoupled in QoS architecture. Without QoS routing, QoS signaling can still work but the resource reservation may fail because selected path may not have enough resources. QoS signaling will work better if it coordinates with QoS routing, and hence allows reservation delays to be minimised and packet loss reduced. Moreover, scalability and overhead are improved beacause a minor number of update message are sent and coupling option ensures that the request for a QoS reservation only occurs when there are valid routes in the netwok. There are three scales of coupling which are described below :

The **de-coupled** option refers to the fact that QoS signaling and routing mechanism operate independently of each other and the QoS-signaling implementation is not dependent on a particular routing protocol . When a mobile node change its location, the QoS for traffic stream will be disrupted while until a new reservation is installed via refresh message between nodes. Whereas old reservation path must be left to time out, which is not most efficient use of network resources.

In a **loosely coupled** approach , QoS signaling and routing interact with each other . Interaction can be understood as bi-directional.Some routing protocols allow upper layers for installation of upcall procedures to be called whenever a route changes. This might significantly decrease the reaction time

of the QoS signaling to restore a certain reservation for a flow rerouted,thus minimize the disruption to the application's traffic streams.

On the other hand, QoS signaling could provide feedback information to the routing layer regarding the route chosen and ask the routing protocol for alternate routes if the route provided doesn't satisfy the QoS requirements. The loosely coupled option ensures that the new routing information has been distributed into the network before attempting to install the reservation. Despite of these benefits any kind of interaction between QoS signaling and routing may lead to a solution which is dependent on the specific routing protocol.

In a **closely coupled** approach, there exists strong coupling between QoS signaling and routing due to having the same signal mechanism to propagate the mobility and QoS information. Hence, it can reduce amount of signaling required to setup reservations, and also lead to fast flow reservation. Regardless of providing improvements in performance and efficiency,it has also some major drawbacks. First, this approach is dependent on particular routing portocols which is used to convey QoS information across the network . Hence, the QoS implementation will be specific to a particular routing protocols, which is not suitable, because routing in MANETs still underlies heavy research . Second, the route computation with this strategy may be too complex [20].

3.3 QoS Signaling: Issues

The QoS signaling mechanism is the protocol in charge of resource reservation,renegotiation of flows in the network, as well as sending and receiving reports describing the network conditions from the other end.Each flow having some QoS guarantee will have to be allocated some resources which will be exclusively for its use. Thus, packets from that particular flow will not have to wait for some resource to be freed and, packets will be transmitted instantly to next node. There are a few issues to be considered when designing a QoS signaling protocol (especially in MANETs) as to how the control information is carried along with data and how the flow path is established.Signaling that a particular resource has been allocated for a particular flow can be done in two ways:

3.3.1 In-Band vs. Out-of-Band Signaling

- In-Band Signaling:

In this, the signaling information (control information) is piggybacked on the data packet i.e the current data packet will carry information about the next packet. The signaling path will always follow the data path, even in cases when the route has changed because of failure of an intermediate node. This leads to very fast flow state restoration.

- Out-of-Band Siganling:

  In this, the signaling information is exchanged through a separate channel between the nodes. This consumes more bandwidth and leads to an increase in network traffic. In this, the signaling packets compete with the data packets for bandwidth and, signaling packets must have higher priority than data packets in order to achieve on-time notification. This can lead to a complex system where performance will degrade substantially. MANETs can not tolerate complex signaling protocol, hence this approach is not suitable for resource limited MANETs.

3.3.2 Soft-state versus Hard-state

We can define two different methods to keep resource reservation: Soft and Hard state. In hard-state , the resource reservation is kept (guaranteed) for the duration of the session. This happens until an explicit release message is sent. In soft-state, when a reservation is established, a timer is triggered. Within a certain time period, specific signaling or traffic will update the soft state reservation and reset the timer. Otherwise, when the timer times out without receiving any refreshing messages, the soft state reservation will be released.

Hard-state method is not suitable in mobile ad hoc networks, where the route discovery and resource reservation need to adapt to topology changes in timely manner. For example , once a mobile node has lost its connection with the network, it might not have a chance to send any signaling for hard-state release. So the reserved resource would permanently be kept unused. Soft-state method may solve this problem because the reservation state just times out.

3.3.3 One-pass versus Two-pass

3.4 QoS Adaptation

Due to the dynamic network characteristics of mobile ad hoc networks, the QoS provision may change rapidly in mobile ad hoc networks. The same level of QoS may not be available if, in case, the sessions are re-routed. Also, the network quality is also dynamic. By network quality, we mean the available resources int network. So applications can t rely on the QoS investigation done during session establishment. Instead, applications must adapt to time-varying low-capacity resources offered by the network. The adaptive strategies play very important role in supporting QoS in MANETs. In addition, it decides how and when QoS of a certain flow has to be investigated and, determines how to react to QoS changes. In the following we outline the different types of adaptation strategies:

- Fair Strategy:

  As the number of application flows competing for resources increases, rather than simply refusing to admit new flows or pre-emptingexisting flows, a fair adaptation strategy attempts to adjust the allocation for each flow, so that all flows can be accommodated. The strategy attempts to give each flow the minimum bandwidth requested, plus a fraction which is proportional to the requested bandwidth range.

- Greedy Strategy:

  The Greedy adaptation strategy is the simplest possible. Regardless of any end-to-end information every node tries to allocate the bandwidth requested.

- Bottleneck driven Strategy:

  In a bottleneck driven strategy each node would only try to allocate as much bandwidth as has been allocated already by previous hops for a certain flow, except the first node in the path of course which always tries to allocate maximum bandwidth. This avoids temporarily bandwidth waste but on the other hand makes it very difficult to increase end-to-end bandwidth for a flow.

The following sections presents overview of some of QoS-aware protocols defining their role in providing QoS in MANETs .

3.5 QoS MAC protocol

QoS MAC protocols solve the problems of medium contention, support reliable unicast communications, and provide resource reservation for real-time in a distributed wireless environment . A lot of MAC protocols proposed so far solve only medium contention and hidden/exposed terminal problems and improve throughput. Most of them do not provide resource reservation and QoS guarantees to real-time traffic .

MAC protocol which can provide QoS guarantees to real-time traffic in a distributed wireless netweork include **GAMA/PR** protocol,**MACA/PR** protocol and Black Burst (**BB**). We give brief description of *MACA/PR* protocol and *BB* scheme below :

3.5.1 Black Burst Contention Scheme

The Black burst (***BB***) contention scheme can be seen as an extension to existing CSMA/CA type protocol.The main goal of this scheme is to avoid packet collision and packet starvation .

According to scheme, when the medium remains idle for long time, each real-time node first contend for transmission right by jamming the media with pulses of energy, which are called BB's, before initiating transfer of packets. Following each BB transmission, a node senses the channel for an observation interval. Each contending nodes uses BB's with different length where each node can determine whether its BB is of greatest length. Thus only one winner is produced after contention , who will trasmit its real-time packets successfully. The length of BB is measured in terms of number of black slot.The number of slots that forms a BB is an increasing function of contention delay.(***BB***) contention scheme ensures that real-time packets are transmitted without collisions and with priority over best-effort packets.

3.5.2 MACA/PR

Multihop Access Collision Avoidance with Piggyback Reservation (*MACA/PR*) provides rapid and reliable transmission of non-real-time datagrams as well as guaranteed bandwidth support (via reservation) for real-time traffic. It establishes real time connections over a single hop only . Notice that *MACA/PR* relies on two underlying assumptions :

1. that real-time packets arrive at constant time intervals
2. that communication links are symmetric .

For the transmission of best-effort traffic, node must wait for free window in the reservation table (**RT**), which records all reserved send and received windows of any node within transmission range. After waiting additional random time on the order of single hop round-trip delay , node proceeds with *RTS-CTS-PKT-ACK* dialogue for packet transmission if node sense the free channel. If not, node repeats same procedure.

The behavior of MACA/PR is different in terms of transmission of real-time packets.In order to transmit the first data packet of a real-time connection, a RTS/CTS dialog is used on each link and proceeds with PKT-ACK dialogue if the CTS is received . For subsequent data packets(excluding first packet) of a real-time connection, only PKT-ACK dialogue are used . Any node near the sender which hears the RTS will defer long enough so that the sender can receive the returning CTS . Any node near the receiver which hears the CTS will avoid colliding with the following data packets.

In order to reserve bandwidth for real-time traffic, the real-time scheduling information is carried in the headers of PKTs and ACKs. The sender piggybacks the reservation information for its next data packet transmission on the current data packet . Upon receiving data packet correctly, the intended receiver enters the reservation into its reservation table and confirms it with an ACK to the sender .

The neighbours which hear the data packet can learn about the next packet transmission time . And neighbours at receiver side which hears the ACK will avoid to send at the same time when the receiver is scheduled to receive next packet . The real-time packets are protected from hidden hosts by the propagation and maintainance of reservation tables(**RT**) among neighbours, not by the RTS-CTS dialogues.

If the sender fails to receive ACK N times , it assumes that the link is not satisfying the bandwidth requirement, and notifies the upper layer , i.e.,the QoS routing protocol . So ACK serves as a mechanism to inform the sender if something is wrong on the link. The ACK serves as renewing of reservation rather than for recovering from packet loss since packets are not retransmitted if ACK is not received.

3.6 QoS routing protocol

A fundamentally different aspect of QoS support in mobile wireless ad hoc networks is that the characteristics of the routing should meet the demands of realtime applications to the largest extent possible. QoS routing is an essential part of the QoS architecture whose main goal is to determine a path from a source to destination based on some knowledge of the quality of network, and according to the quality of service requirements of flows . To achieve this goal ,routing protocols must balance efficiency and adaptivity while maintaining low-control-overhead because of features that routing functionalities are distributed accross all network participants.

The objective of QoS routing are threefold:

1. if one exists, find a path that has sufficient available resources capable of satifying the QoS requirement.

2. optimize the use of network throughput and network resources.

3. adapt to network congestion, providing smooth performance degradation to lower-priority traffic

Routing protocol proposed for the MANETs can be classified into three broad catagories: (a)*precomputed table-based routing scheme* , (b) *on-demand source-based routing scheme*, and (c)*constraint-based routing scheme.*

Under the table-based scheme, each node maintains tables of routing information which contains next-hop information to reach destination.By sending periodic updates throughout the network,the scheme tries to maintain table information consistent.

In case of on-demand source-based routing schemes, the routes are created whenever necessary based on a query-reply process. Once a route is found, it is maintained by a route maintainance procedure until the route is no longer needed. Under this routing scheme, shortest path between source-distination is created by taking account of nodes status and network configuration when a route is desired. A widely used ad hoc routing protocol AODV [2] is an example of source-based routing scheme.

Routing protocols under the constraint-based scheme use metrics other than shortest-path for finding a suitable and feasible route.Protocols like Associativity-Based Routing (ABR) and Signal Stability Routing(SSR) take account of the node's signal strength and location stability sothat the path chosen is more likely to be long-lived.

The large number of routing scheme found on litterature were proposed for routing message on the shortest available path or within system-level

constraints.Routing messages in such paths may not be adequate for applications that require QoS support. Only a few QoS-aware routing protocols such as CEDAR,ticket-based, and QoS routing based on bandwidth calculation are proposed for yet for MANETs. Below we outline feature of CEDR and ticket-based routing scheme.

3.6.1 CEDAR

CEDAR[21] routing protocol is proposed as a QoS routing scheme for small to medium size ad-hoc networks consisting of tens to hundreds of nodes . It dynamically establishes the core of the network, and then incrementlly propagates the link states of stable high-bandwidth links to the core nodes. The route computation is on demand basis, and is performed by the core nodes using only local state. while most routing protocols use indirect transmissions to distribute control messages across the network, *CEDAR* transmits directly . CEDAR uses a modified MAC-level frame format to identify redundant message transmissions.*CEDR* includes three main components which are outlined below :

- **Core Extraction**

  The purpose of core extraction is to elect a set of hosts to form a core of the network by using only local computation and local state. The core nodes are elected by approximating a minimum dominating set[1] of the ad-hoc network. Each node in the core then establishes a unicast virtual link with other core nodes a distance of three or less away from it . Every host not in the dominating set (DS) choose a core neighbour as its dominator. All routing functionality is limited to the core host.

- **Link stat propagation**

  QoS routing in CEDAR is achieved by propagating the bandwidth availaibility information of stable links to all nodes. As each host monitors the available bandwidth on the links to its neighbor, the respective dominators will be informed about any changes on the link to initiate core broadcast. The basic idea is that the information about stable high-bandwidth links can be made known to nodes far away in the network, while information about the dynamic or low bandwidth links remains local.

[1] A dominating set is a subset of the network in which every node not in the set is adjacent to at least one node in the set.

- **Route computation**

  The QoS route computation in CEDAR includes three key phases:(a) locating the destination node and establishing a core path to the destination . (b) finding a stable route using established core path as a directional guide.(c)dynamically re-computing QoS routes upon link failures or topology changes. Upon the failure of a link, CEDAR initially attempts to recompute an admissible route at th point of failure.

3.7 QoS Models and frameworks in MANET

A framework for QoS is a complete system in which some kinds of services could be provided in MANETs. All componets within this system co-operate together in providing the required services. The key component of any QoS framework is the QoS model, which defines the way users requirements are met. A QoS model defines the nature of service differentiation , and it is the system which should implement such service differentiation . In following sections, we review and discuss some of QoS models and frameworks and their shortcomings proposed for ad hoc wireless network (MANET). SWAN[4] model is main focus of this thesis and it is described detail in Chap.4 .

3.7.1 INSIGNIA

There exists few inter-layer QoS frameworks for MANETs of which INSIGNIA[?] is one such model. INSIGNIA is the first signaling protocol designed explicitly for MANETs, which can be combined with a variety of routing protocols. INSIGNIA is only one component of the QoS architecture and does not define specific algorith but assumes the availiability of routing,packet scheduling and MAC protocol [20]. It supports fast flow reservation,restoration and adaptation algorithm that are specifically designed to deliver adaptive real-time service in MANETs environment.INSIGNIA framework can provide assured adaptive QoS levels to real-time applications, based on the QoS requested by application and the resource availability in the MANET.

INSIGNIA can be characterized as an in-band RSVP signaling protocol since it encapsulates control signals in th IP option of every IP data packets. Each INSIGNIA IP option carries all the information required to setup or maintain a reservation. Utilizing in-band signaling prevents QoS-related control packets from having to compete with the data packets for access to the channel

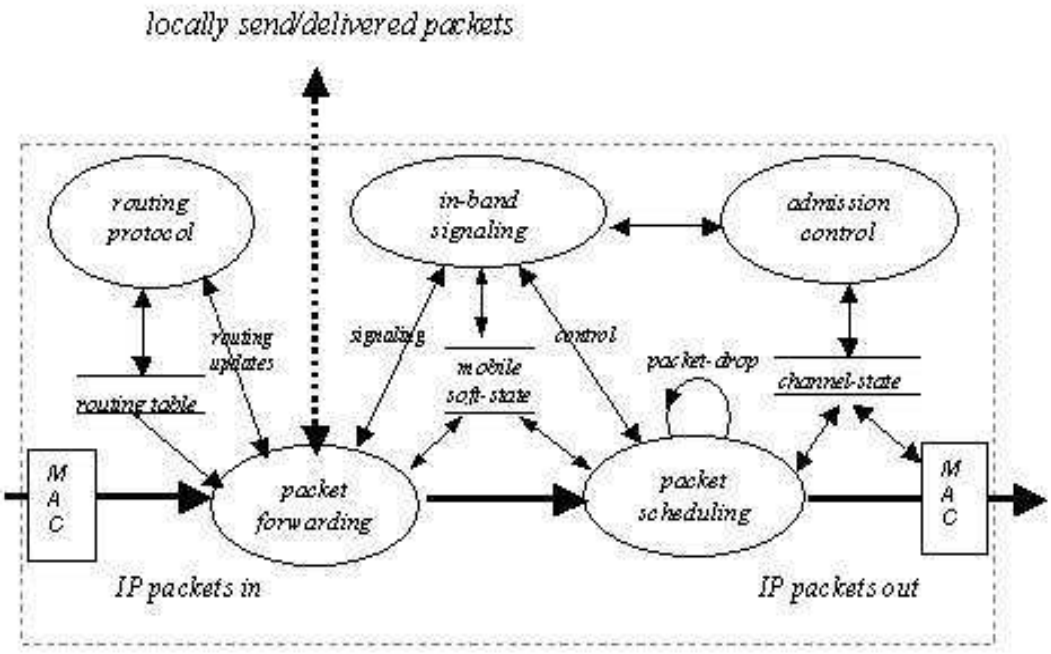

Figure 3.3: INSIGNIA Framework

and can fecilitate rapid restoration of QoS requirements along a new path during topology changes. Under this scheme,user session is adapted to the available level of service without explicit signaling between source-destination pairs. The INSIGNIA module adopts soft-state mechanism where resources are released if a signal does not arrive in intervals from the time of the reservation.

Fig. 3.3 shows the architectural componets of INSIGNIA framework. As mentioned previous, INSIGNIA is responsible for multiple operations likeestablishing,restoring,adapting and tearing down real-time connections.

Admission controll allocates resources to flows based on base/enhanced(i.e maximum) QoS request. Afterresources are allocated, the resources are then periodically refreshed by a soft-state mechanism through the reception of packets. Additionally, QoS reporting (see fig. 3.4 on the next page is used to notify source nodes of the current status of flow. Figure 3.4 shows soft-state mechanism of INSIGNIA scheme. **Packet forwarding** classifies incoming packets as signalling or data packets, and forward them to the appropriate

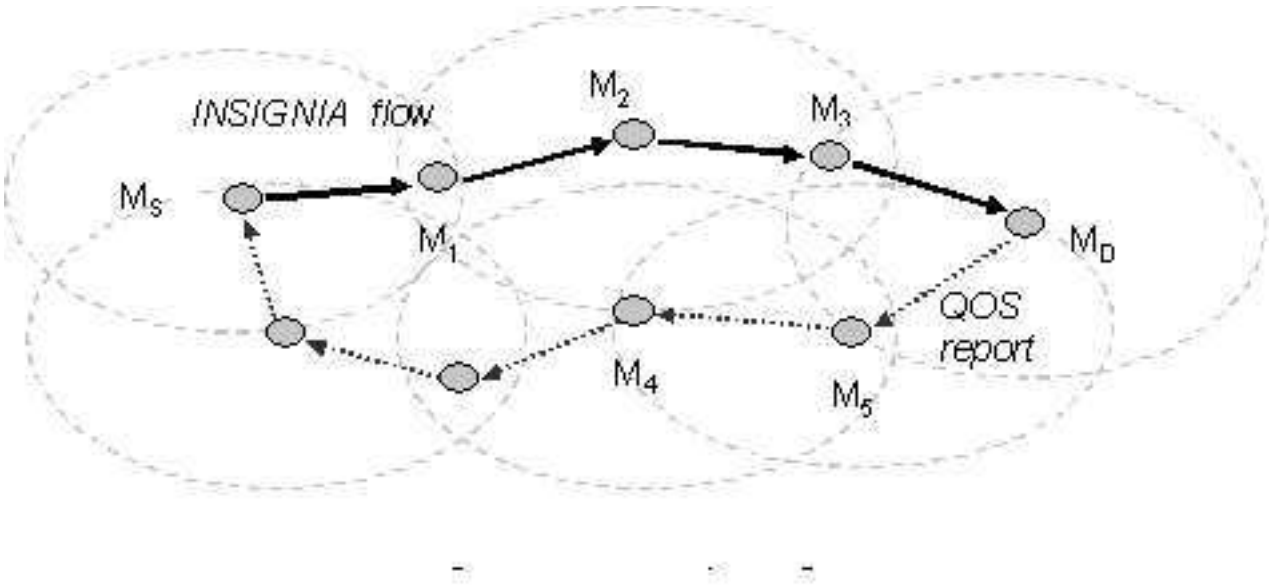

Figure 3.4: INSIGNIA QoS Reporting Scheme

module. **Routing** adapts to the dynamics of the network and provides routing table to the *Packet forwarding* module. **Packet scheduling** responds to location-dependent channel conditions when scheduling packets in MANETs.

3.7.2 FQMM

FQMM QoS model is designed for small to medium-sized MANETs using a flat nonhierarchical topology. This is an inter-layer based model for MANETs working in the IP layer with the co-operation of the MAC layer . FQMM considers the characteristics of MANETs and combines both the per-flow stat property of `IntServ` and the service differentiation of `DiffServ`. Therefore, FQMM applies a hybrid provisioning where both IntServ and DiffServ scheme are used separately . `FQMM` model try to preserve the per-flow granularity for a small portion of traffic in MANET given that the large amount of traffic belongs to per-class granularity. This model is based on the assumption that not all packets in ad-hoc network are actually seeking for highest priority.

FQMM consist of three key features: *(a) dynamic roles of nodes (b) hybrid provisioning and, (c) adaptive conditioning.* This hybrid QoS model defines three types of nodes(see figure 3.5 on the following page) as in `DiffServ`:(a) a **ingress node** that sends data (b) an **interior node** that forwards data for other nodes and (3) an **ingress node** which is a destination node. The role

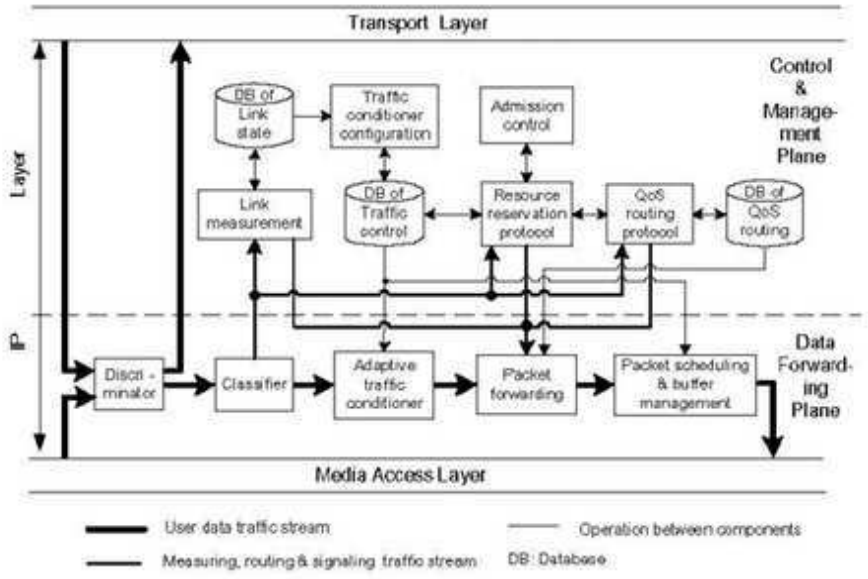

Figure 3.5: FQMM Framework

of a mobile node is adaptively changing based on its position and the network traffic.Provisioning is used to find and allocate needed resources at various points (mobile nodes) in ad-hoc network. The traffic conditioner, which polices (marks) the traffic according to the traffic profile ,is placed at the ingress node at which traffic originates. It is responsible for re-marking or discarding packets according to the traffic profile, which describes the temporal properties of the traffic stream such as transmission rate and burst size. The source and intermediate stations perform traffic shaping according to those marks. FQMM suggests to mark packets as in-profile and out-of-profile by making use of token bucket algorithm. Packets marked as out-of-profile are dropped with high probability than in-profile packets.

A framework architecture of FQMM (see fig. 3.5) defined two differ-

ent planes : (a) Data forwarding Plane (b)Control and Management Plane. The modules in the data forwarding plane handle the data packet forwading function while the components in the management plane prepare for the operation of the data forwarding plane according to specific protocols[The hand book of] .All the modules in different panel co-operate with each other to provide desire QoS service.

3.7.3 iMAQ

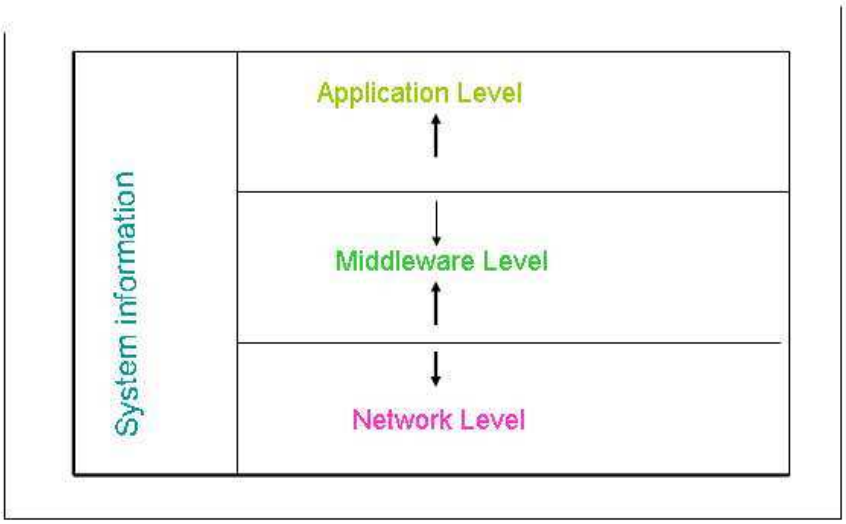

Figure 3.6: iMAQ Framework

The integrated Mobile Ad hoc QoS (iMAQ) framework is one of the cross-layer architeture to support real-time traffic over a MANET . As illustraded in figure 3.6, the framework consists of middleware service layer and network layer which cooperate with each other to provide QoS guarantee to multimedia traffic. By using location and moving pattern information, middleware tries to predict networking partitioning which will occur after it replicates data between different network groups.

In order to provide better data accessibility , middleware includes a data lookup service and a data replication service. The data lookup service maintains ,on each node, a data availability table and messages advertising about data availability are excanged between group members periodically.Depending on the reservation mechanism, the table entries will be updated. For example, in the case of soft-state approach, table entries is refresh once such a message is received. In addition, middleware layer negotiates with the applications in case of QoS degradation in the network.

In the iMAQ framework, the nework layer is fecilitated with a predictive location-based QoS routing protocol,which tries to predictfuture location of nodes based on their location/resources updates.The process of predicting location is given below:

Every mobile nodes generates it update message perodically when its moving pattern or resource availiablity change considerably. Taking the generated updates into account, routing protocol try to calculate end-to-end delay of a packet to its destination. Based on this estimation and destination's location updates, protocol will try to predict the destination's location at the moment packet is expected to arrive. The process continue until the destination is reached.

3.8 Conclusion

In this chapter, we discussed some of the issues which are relevant in designin the QoS framework for mobile ad hoc networks, and presents some QoS framework designed especially for MANETs. Both QoS model (**INSIGNIA**,**FQMM**) attempt to provide quality of servic to applications by taking account of the dynamic nature of mobile ad hoc network.But both of the9m have some shortcomings.

INSIGNIA supports operational transparency among multiple routing protocols through the separation of routing,signaling, and packet forwarding. But, it is important to notice that INSIGNIA is just the signaling protocol and, that there is a need to include routing protocol which can track the topology in ad hoc networks. Under the INSIGNIA framework, the idea is to avoid explicit signaling and hard state reservation in order to deal with dynamics of mobile ad hoc networks in a flexible way. The major drawbacks of INSIGNIA is that flow state (i.e. reservation) information should be kept in the mobile hosts, which could becomes a scalability problem as the number of flow states increases.

FQMM is the first attempt at proposing a QoS model for MANETs and, is different approach to employ DiffServ in ad hoc networks [9]. However, there exists some drawbacks in this approach. In FQMM, both IntServ and DiffServ scheme are separately used for different priority classes. Therefore, the drawbacks related to IntServ and DiffServ reamain to be drawback in FQMM. Another drawbacks of this approach is that source nodes have to take great care in regulating their traffic, since the rate of in-profile traffic must be processable in all network regions .
A more effective QoS model, which can utilizes network resources effectively while providing quality of service to time-bounded applications in ad hoc networks, is required. In next chapter, we outline the SWAN, a distributed quality of service model for MANETs, which is more robust than FQMM and INSIGNIA .

Chapter 4

SWAN

The goal of this thesis is to enhance SWAN model by analyzing parameter's set by SWAN to support QoS in MANETs. In this chapter, we give detail overview and introduction of QoS model for MANETs, called SWAN (Service Differentiation in Stateless Wireless Ad Hoc Networks).

4.1 Introduction

SWAN [4] is a newly introduced independent QoS model that operates on wireless ad-hoc networks. As an independent QoS model, SWAN is flexible and may run overany routing protocol layer or Media Access Control (MAC) such as IEEE802 Distributed Coordinated Function (DCF). SWAN uses a stateless distributed approach to solve the dynamic of QoS issues. As a result, there is no need for signaling or complex control mechanism to update, refresh and remove per-flow state information, as is the case with "stateful" mobile ad hoc networks. Changes in network topology and link failure do not affect the operation of the SWAN control system.
An important contribution of SWAN is that it is designed to handle both real-time UDP traffic, and best effort UDP and TCP traffic without the need for the introduction and management of per-flow state information in the network. Furthurmore, SWAN support per-hop and end-to-end control algorithms that primarely depend rely on the efficient operation of TCP/IP protocols.
Among the proposed QoS models, SWAN [4] has illustrated higher level of robustness and ability to recover from adverse mobility situations when compared to FQMM, or INSIGNIA due to its stateless model. SWAN also introduces the concept of "soft" real-time service guarantees, under which an admitted real-time sesson at any time during the lifetime of this session

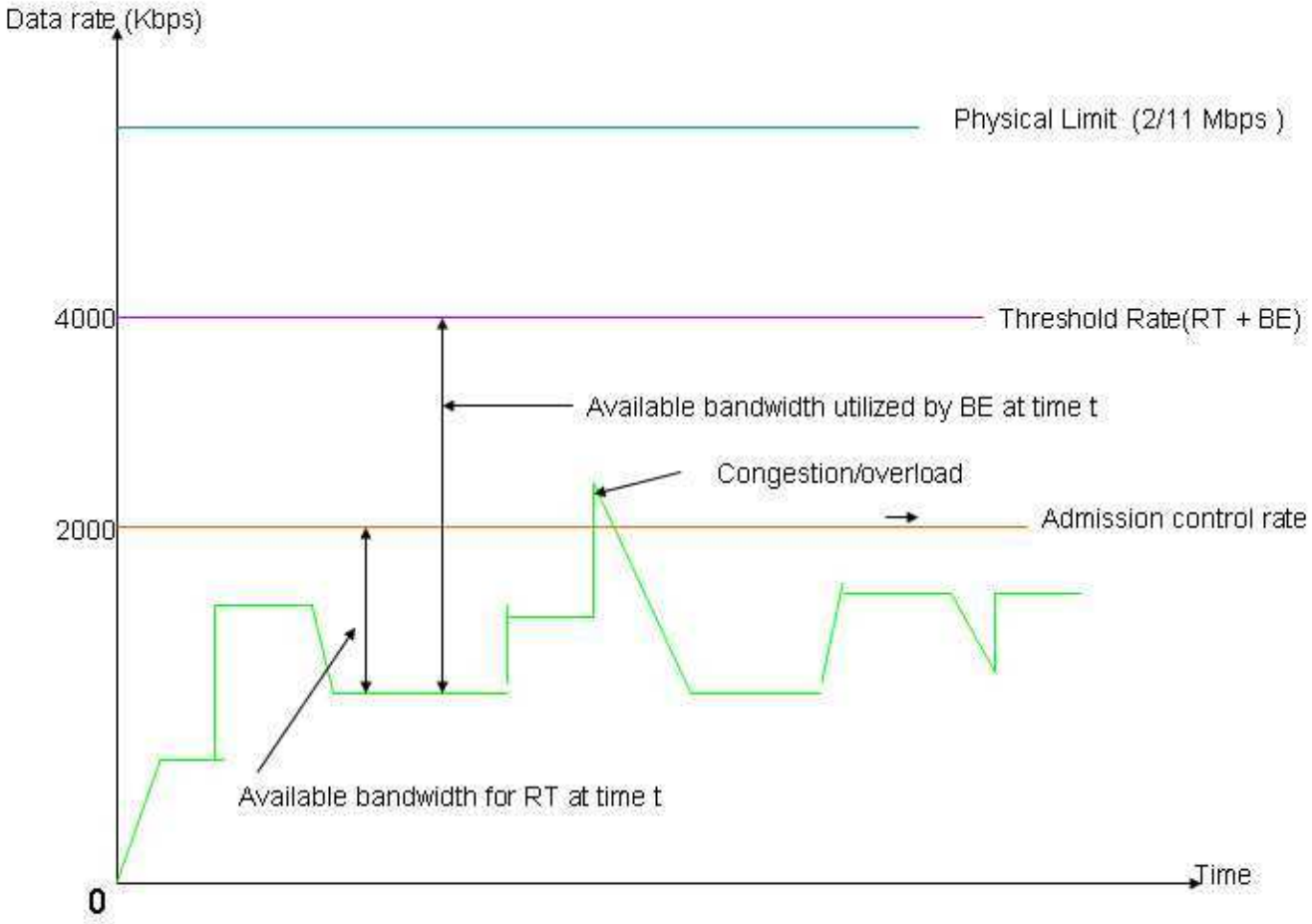

Figure 4.1: Graphical presentation of data rate in SWAN

could be downgraded to best-effort, or stopped all together in response to nework dynamics.
SWAN uses local rate control for UDP and TCP best-effort traffic, and sender-based admission control for UDP real-time traffic. Rate control is designed to restrict best-effort traffic yielding the necessary bandwidth required to support real-time traffic. Total rate of both real-time and best-effort traffic transported over each local shared media channel is maintained below certain "threshold rate", limiting any excessive delays.SWAN adopts engineering techniques that attempt to set the admission threshold rate at mobile nodes to operate under the saturation level of wireless channel. SWAN adopt the well-known additive increase multiplicative decrease (AIMD) rate control algorithm (see fig. 4.4 on page 38) to address challenges related to congestion in the network.

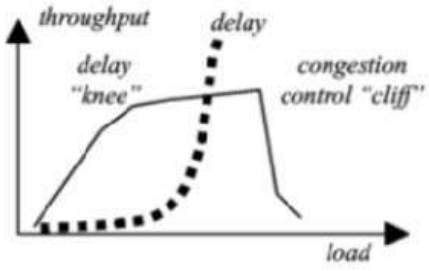

Figure 4.2: Congestion Controlled System

In response to network dynamics, which leads to occasional network overload condition, SWAN uses a regulation mechanism based on ECN to dynamically regulate admitted real-time flows, which was originally proposed for controlling and improving TCP traffic performance in IP networks.

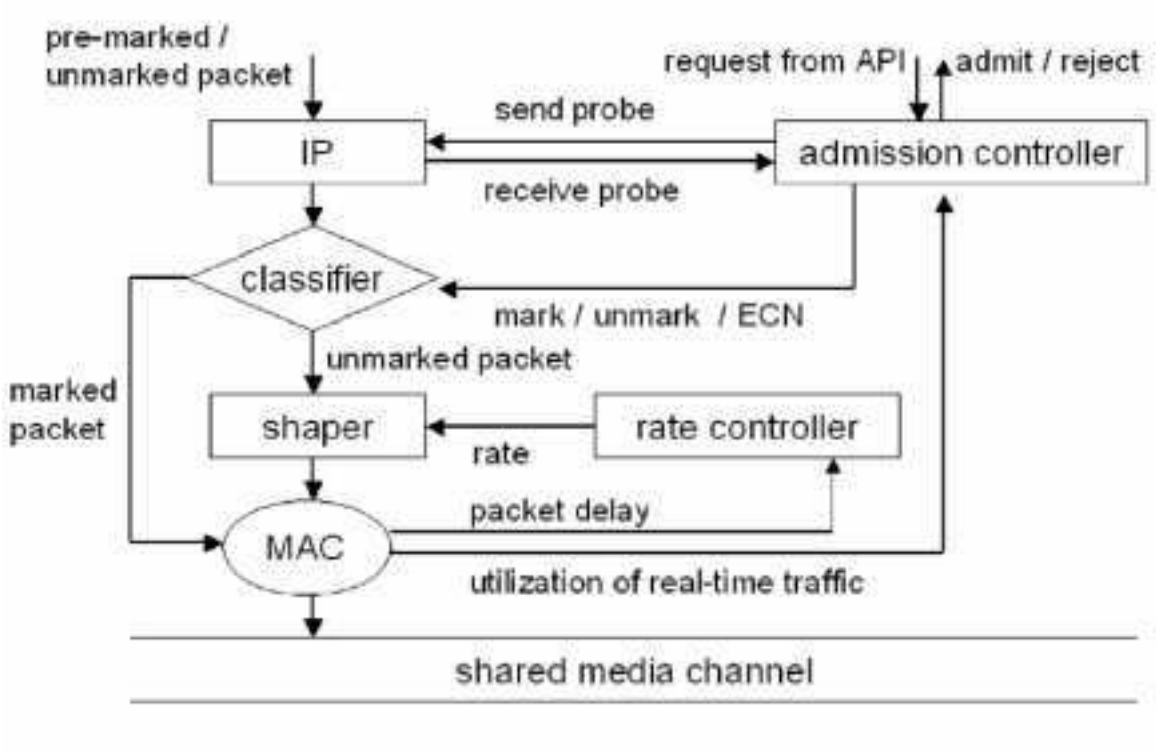

Figure 4.3: SWAN Model

4.2 SWAN Architecture

As mentioned in previous section, proposed architecture is designed to handle both real-time UDP traffic, and best effort UDP and TCP traffic. As illustrated in fig. 4.3, SWAN model consists of a number of mechanisms to support service differentiation in mobile wireless ad hoc networks. In the following sections, we presents overview of main components of SWAN architecture.

4.2.1 Admission Control

SWAN Admission Control(AC) adopts well known Additive Increase Multiplicative Decrease **AIMD** algorithm (see fig. 4.4 on the next page), which has been used by the TCP congestion controller. In contrast to traditional TCP congestion controll algorithm, SWAN **AIMD** admission control algorithm follows a comparatively conservative approach in which SWAN keeps the system at the delay "knee", where the system throughput is almost the same as at the cliff, but queues are significantly less loaded (

```
Procedure update_shaping_rate ( )
/* called every T second period */
Begin

if (n > 0)                      /* one or more packets have delays
                                   greater than the threshold delay d sec */
   s ← s * (1 – r / 100)        /* multiplicative decrease by r% */

else

   s ← s + c                    /* additive increase by c Kbps */

if ( (s – a) > a * g / 100)     /* difference between actual rate and shaping
                                   rate is greater than g% of actual rate */
   s ← a * (1+ g / 100)         /* adjust shaping rate to match actual rate */

end
```

Figure 4.4: AIMD rate control algorithm

see fig. 4.2 on page 36).

To estimate the threshhold rate that would trigger excessive delay is very difficult in ad hoc networks. The threshhold rate changes according to traffic pattern in MANETs and thus estimation of avialable local shared bandwidth is challenging. SWAN take simple approach and admit real-time traffic up to a rate that is more conservative than the threshhold rate .

In SWAN, admission control rate is predefined value and set to conservative value of 2 Mbps, for a channel data rate of 11 Mbps. In addition, this slack of bandwidth can be viewed as a safety measure against the ad-hoc dynamic such as bandwidth variation and mobility. Hence, the estimated available bandwidth of a local shared median is simply the difference between this admission control rate and the current rate of real-time traffic. Also, the threshhold rate is predefined value and set to value of 4 Mbps.SWAN aims to maintain total traffic rate of both real-time UDP traffic and, best effort UDP and TCP under this threshold rate .

An admission controller(**AC**) is responsible to admit real-time flow in a ad-hoc network. Any mobile hosts using shared wireless channel can listen to packets sent within its radio transmission range. **AC** use this feature to measure local resource avialibility and real-time traffic at each node. In

SWAN, a mobile ad-hoc node can admit a real-time flow only after obtaining an accept message.The process of admitting a new flow is as follows :

Admission Controller located at the source node starts by sending a probe request packet to the destination node to estimate end-to-end bandwidth. The probing request packet is a UDP control packet that contains a "bottleneck bandwidth" field. Probe request packet visit each node in the route provided by a MANET routing protocol. Intermediate nodes intercepts the request and update it with the bottleneck bandwidth if the bandwidth availability at the nodes is less than the current value of the field. Intermediate nodes use their **AC** to calculate bandwidth for newer real-time sessions, but they do not implement resource, or bandwidth allocation. The destination node replies with a probe response packet that contains the bottleneck bandwidth along the path between source-destination pairs.

Upon receiving the probing request packet, source node can execute the simple source-based admission control by comparing ene-to-end bandwidth availability with the bandwidth requirement for the new real-time flow. If the new real-time flow is admitted, packets associated with flow are marked as **RT** (real-time packets) and bypass the shaper mechanism remaining unregulated.

During the life-time of the session, the bandwidth probe instantaneously reads the state of the path provided by routing protocol and makes a local source based admission decision based on the bandwidth probe reply.

4.2.2 Rate Controller

SWAN uses local rate control at each node for best effort traffic to yield bandwidth necessary for real time traffic. The Rate Controller(**RC**) determines the departure rate of the shaper using an AIMD rate control algorithm (see figure 4.4 on the page before based on feedback which represents the packet delay measured by the MAC layer. The packet delay is measured at the source node by substracting the time that a packet is passed to the MAC layer from the time an ACK packet is received from the next-hop receiver.

The RC detects excessive link delay when one or more packets have greater

delays than a threshhold link delay **d**(sec).The threshhold delay is based on the real-time delay requirements of applications in wireless network. When excessive link delay is detected, RC backs off the rate by **r**% .
The rate controller has to re-adjust parameters (d and r) every T seconds. The period T should be small enough to be responsive to the dynamic of MANETs.

The RC monitors the actual transmission rate to avoid a scenario where a node is capable of transmitting burst due to large difference between the shaping rate and the actual transmission rate. RC adjusts the shaping rate to be **g**% above the actual rate should the difference between the shaping rate and actual rate is greater than **g**% of the actual rate.

4.2.3 Shaper

The shaper represents simple leaky bucket traffic shaper. The goal of the shaper is to delay best effort packets in conformance with the rate calculated by the rate controller to reduce the contention between neighbouring stations.

4.2.4 Classifier

A classifier operate between the IP and best effort MAC layers and is capable of differentiating real-time and best effort traffic. It selects real-time packets to bypass the shaper. SWAN implicitly assumes that real-time flows need not to be policed.

4.2.5 ECN-based Regualtion of Real Time Traffic

SWAN adopts the explicit congestion notification(ECN) regulation algorithm to recover from congestion conditions caused due to the rerouting of admitted flows(mobility) or false admission. Each node continuously, and independently, moniters the utilization fo its real-time traffic to estimate the local available bandwidth. Mobile nodes can detect violation (overload condition) using this periodic traffic measurement. Congested nodes will then use the ECN bits in the IP header of the real-time packets to inform destination node of the existence of congestion. Each destination node will issue a regulate message to the relevant data flow originator about congestion. Source nodes will then re-initiate new probe requests to locate possibly a better service route to the destination based on its original bandwidth needs. A source node terminates the session if the estimated

end-to-end bandwidth indicated in the probing response packet can not meet its existing session needs.

According to this regulation approach, all congested nodes mark the packets of flows traversing through these nodes with congestion experience (**CE**) and thereby forcing every flows to reprobe their real-time service. This is an inefficient approach and, the more efficient and systematic approach, which may only penalize a small number of sources is needed. To overcome this scenario, SWAN has proposed two different regulation approach, namely source and network based regulations. Both approaches mark ECN packets differently, but use the same consequences afterwards.

- **Source-Based Regulation**

  Under this regulation, when a sender node receives regulate message, it immediately perform multiplicative decrease on relevant real-time flows. As a result,the congested node experience a gradual decrease of real-time traffic until congestion condition is removed(i.e rate of real-time traffic is under admission control rate), at that point, the congested node stops marking CE bits.

  The source nodes waits for a random amount of time before re-initiating a probe request to re-establish the desire level of service should the used bandwidthof a specific RT (real-time) flow is unsatisficatory to source nodes. In source-based regulation, source nodes have to stagger the regulation probe in order to avoid a flash-crowd condition, where, a number of sources simultaneously re-initate probe at the same time and see the path overbooked and drop their real-time flows .

- **Network-Based Regulation**

  Under this regulation algorithm, a overloaded node randomly select a subset of all its real-time flows to be a "congestion set" or "victim flow set" and only marks packets associated with this subset. This regulation approach follows the same procedure as in source-based regulation, when a packet of victim real-time flows will reach relevant destination nodes marked with CE. A congested node marks the congested set for T seconds and then calculates a new set of victim flows. As in previous regulation mechanism, there is need for intermediate routers to distinguish between flows that have been falsely admitted or not. As a solution to this problem, SWAN

suggests applying some intelligence at the overloaded node,namely, to let source nodes inject RT flows with RT-old or RT-new,using the IP-TOS field. Thus congested nodes can use a biased function to form the set of victim flows out of newer flows in a bid to decrease false admission.

Notice that both regulation types have theier disadvantages as well. In the **source-based regulation**, all RT flows going through one overloaded node is regulated eventhough amount of bandwidth violation is limited. In addition, it does not descriminate between different RT flows which leads to a scenario where, higher quality real-time flows might be denied service re-probe while low quality real-time flows maintain connectivity. In another words, sources who waits the shortest period of time before initiating probe have greater chances to drop theier RT sessions. A disadvantage of **network-based regulation** is that it requires some intelligence at intermediate nodes to manage the congested set and descriminate against newly admitted real-time flows.Also, the idea of loading IP-TOS field may conflict with flows that need to use this field when extending over the Internet [16].

4.3 Cons and Prons of SWAN

Like any other QoS model for ad hoc netwoks, SWAN exhibits both advantages and disadvantages. In this section, we summrize the SWAN model by identifying its shortcomings and merits on multi-hop ad hoc networking environment.

4.3.1 Cons

SWAN enjoys some advantages over other independent QoS models, as mentioned in previous section, because of its distributed stateless principle, where operation is fully decentralized. An important merit of SWAN is that, it is independent of the underlying MAC layer and can be potentially suited to a class of physical/data link wireless standards. SWAN mechanism tries to take the special characteristics of wireless links into account .

4.3.2 Prons

SWAN is vulnerable to problems related to mobility and false admission. In addition, SWAN only consider per-link delays not end-to-end delays in

measuring the quality of real-time flows. Real-time flow becomes more sensitive to network dynamics the more end-to-end delay of real-time flow approaches a certain threshhold. Hence, SWAN ignore the dimension of real-time flow delays. Another disadvantage of SWAN is related to the selection of amount of bandwidth available for realtime traffic. Choosing this value too high results in a poor performance of real-time flows and starvation of bulk flows, and choosing too low results in the denial of realtime flows for which the available resource would have sufficed. There is no fairness in bandwidth allocation in this scheme. Admission control is based on first-come first-serve principle. Which flows must adapt to in response to available bandwidth variations, and how much they should adapt, is not deterministic.

In a multi-hop environment, the direct packet capturing is not capable of hearing packet transmission generated by two-hop neighbours. The SWAN admission control for real-time traffic is based on the rate measurement from aggregated real-time at each intermediate node. Howerever, packets sent by a hidden node are not included in the rate measurement. Also ,under SWAN, under-utilization of channel occurs because of exposed terminal phenomenon. SWAN treats all UDP traffic with equal priority and each one of them has an equi-probable chance of getting dropped, or rerouted during congestion. This may not be sufficient for a complete QoS solution for the network. In addition, SWAN enqueues all real time packets into a single fifo queue, without any distiction, thus it can not be called as a complete QoS solution.

Chapter 5

Simulation Results

In this chapter we summerize the thesis with achievements and results. The chapter also includes a critical review of our work and some pointers to further research on the topic.

5.1 Achievements and results

The goal of this thesis is to analyze and enhance SWAN model by simulating original SWAN model with different real-time bandwidth rates than originally proposed by SWAN researcher, and thus comparing analyzed results wih original SWAN model in a bid to see its effectiveness. To achieve this, we need to run simulation implementing SWAN model in ad hoc network environment. In this chapter, we present the results and performance analysis of SWAN through simulation .

5.2 Simulation Environment

We used Network Simulator ns-2[1] with its wireless extensions developed at CMU for our analysis. SWAN code for ns-2 is downloaded from the SWAN webpage [http://www.comet.columbia.edu/swan/simulations.html] and installed on ns-2. The SWAN's ns-2 (2.1b9a) extensions include the AIMD rate controller, admission controller, packet delay measurement mechanism, local utilization monitoring, probe protocol for bandwidth availability estimation, and ECN.

We have two folded simulation scenarios :**Single Hop** and **Multi Hop** . Table 5.1 on the following page lists the common parameter settings used in

Simulation period	300 seconds
Transmission range	250 meters
Routing protocol	AODV
Packet size	512 bytes
Background traffic	TCP
Nominal data rate	11 Mbps
RTS/CTS	ON
MAC protocol	IEEE 802.11
Mobile nodes	50

Table 5.1: Simulation configuration

the both simulations. For the other parameters, the default values of ns-2 are used.
we use a simulated multi-hop network with 50 nodes. During simulation, video flows are active for the duration of 300 seconds representing real-time traffic. Number of video flows depend on allowed admission control rate during different SWAN configuration. Video traffic is modeled as 200 Kbps constant rate traffic with a packet size of 512 byetes. As a background traffic,we have some FTP sources representing best effort TCP traffic. TCP flows are greedy FTP traffic with packet size of 512 byets. SWAN model is simulated with admission control rate 2.0 MB , 1.6 MB og 2.6 MB respectively.
The movement of the nodes is defined by the random way point model, where each node selects a random destination and moves with a random speed up to a maximum speed of 10 m/s. We control the degree of mobility through the pause time. We use pause times of 0,50,100,150,250 and 300 seconds, where pause time of 0 seconds implies constant movement. The source and destination nodes associated with flows are distributed among the mobile nodes in the wireless ad hoc network.

5.2.1 Single Hop Scenario

In this scenario, the network area has a 170m x 170m flat square such that each node lies within transmission range of every other nodes. AODV [2] is used for routing and RTC/CTS mechanism was turned on in the simulated network. Throughout the simulation, each mobile node has a transmission range of 250 meter and shares an 11 Mbps radio channel with its neighbouring nodes. The simulation includes IEEE 802.11 MAC protocol.

5.2.2 Multi Hop Scenario

This scenario is same as Single Hop scenario except the fact that sending-receiving nodes are not within the range of each others. In order to reach the receiver, the transmitting node has to pass its data to the next intermediate node lying on the path between the sending-receiving pair nodes (see figure 5.1). A network area of 1500m x 300m flat square is used in simulation. In our multi-hop simulation environment, average hops is between 2 and 5.

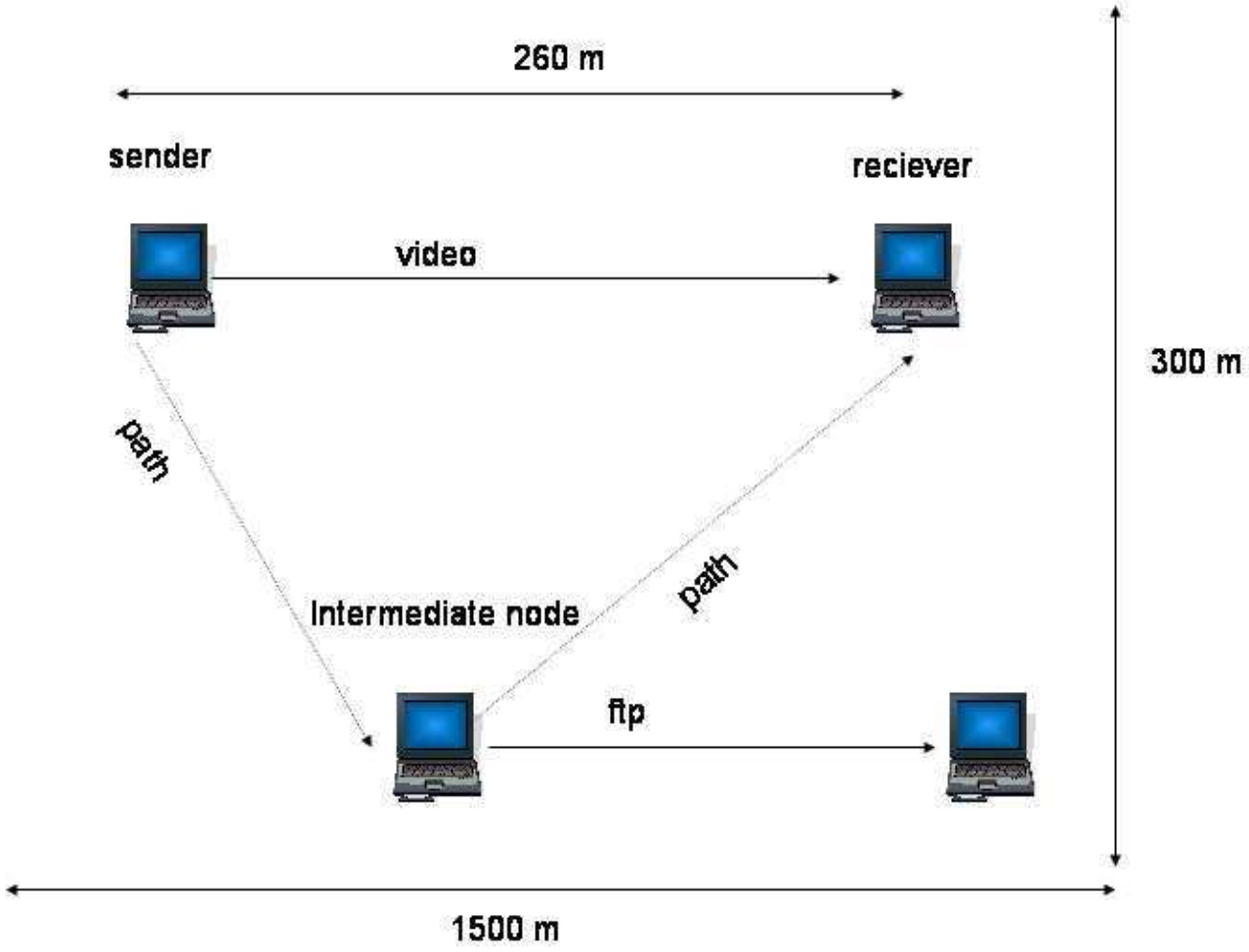

Figure 5.1: Multi Hop Scenario

5.3 Are parameter settings in SWAN reasonable ?

We mentioned in previous chapter about the difficulty of selecting optimal amount of available bandwidth rate (Admission Control Rate) for real-time traffic in SWAN. Since choosing high value for admission control rate contributes to poor performance of realtime flows and choosing low value results in the denial of real-time flows, our main objective on this thesis is two-folded. First, to evaluate whether this admission control rate is optimal to support bound delay for real-time traffic. Second, to see whether SWAN exhibits better QoS support and demonstrate effective real-time performance by increasing or decreasing the admission control rate for real-time traffic than originally proposed. To achieve this, we simulate and evaluate the impact of SWAN model with different bandwidth rate(admission control rate) than originally proposed by SWAN researcher and compare the results. We used the terms average and aggregate interchangeably in our simulation results to indicate the aggregated measurement of all flows. We elaborate further on these issues in the following sections :

5.4 Performance of Single Hop Scenarios

Under this simulation scenario, all simulated nodes lies within the transmission range of each other i.e. sender nodes communicates directly with the receiver nodes.

5.4.1 Impact of Admission Control Rate : 2.6 MB

Here, simulations are based on 10 and 13 video flows representing real-time UDP traffic, and different number of TCP background traffic. The background traffic is modelled as a pair of 2, 4,6,8 og 10 FTP flows. The simulation based on 10 video flows represent the scenario where the SWAN model is simulated with admission control rate 2000 Kbps, and the simulation based on 13 video flows represent the scenario where SWAN model is simulated with admission control rate 2600 Kbps. In figures below, terms ***swan-orig*** and ***swan-2.6*** refer to simulation scenarios, where the SWAN model is simulated with admission control rate set to 2000 Kbps and 2600 Kbps respectively. The simulation graphs below shows the performance of SWAN in each of these cases. We used pause time = 0

second in this simulation. Hence, the mobile nodes are in constant movement.

Real-time traffic throughput :

Fig. 5.2 shows a trace of real-time traffic throughput versus simulation time. The background TCP best-effort traffic consists of 2 FTP flows. As illustrated in fig. 5.2, *swan-2.6* has slightly high real-time traffic throughput than *swan-orig*. As shown in figure, the real-time traffic experience the variation in throughput in both *swan-orig* and *swan-2.6* during the simulation.

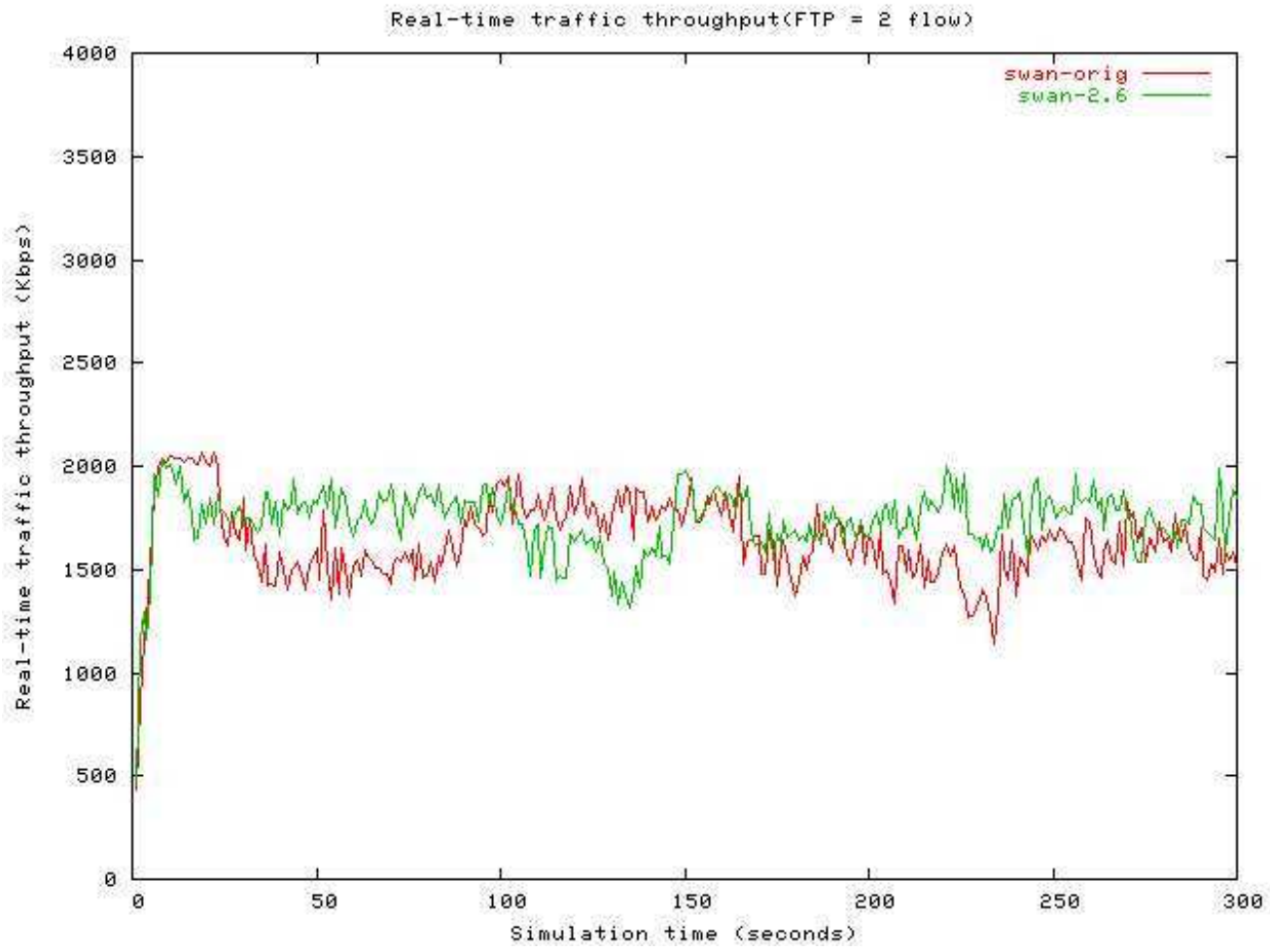

Figure 5.2: Real-time traffic throughput

Fig. 5.3 on the next page shows a trace of real-time traffic throughput with

10 TCP flows. As shown in figure, the *swan-2.6* experience higher real-time traffic throughput than *swan-orig* during the first 50 seconds of simulation period besides the fact that the fluctuation in throughput performance occurs equally in both simulation scenario. On average, there is almost no difference on traffic throughput between *swan-orig* and *swan-2.6*. *swan-2.6* allowed more real-time flow than *swan-orig* which contribute to congestion in network, and SWAN's ECN mechanism could be triggered. Due to ECN, a lot of regulate packet messages will be generated in the network generating more traffic which effect both throughput and end-to-end delay performance. This fact explain the reason why *swan-2.6* does not have high throughput performance than *swan-orig*.

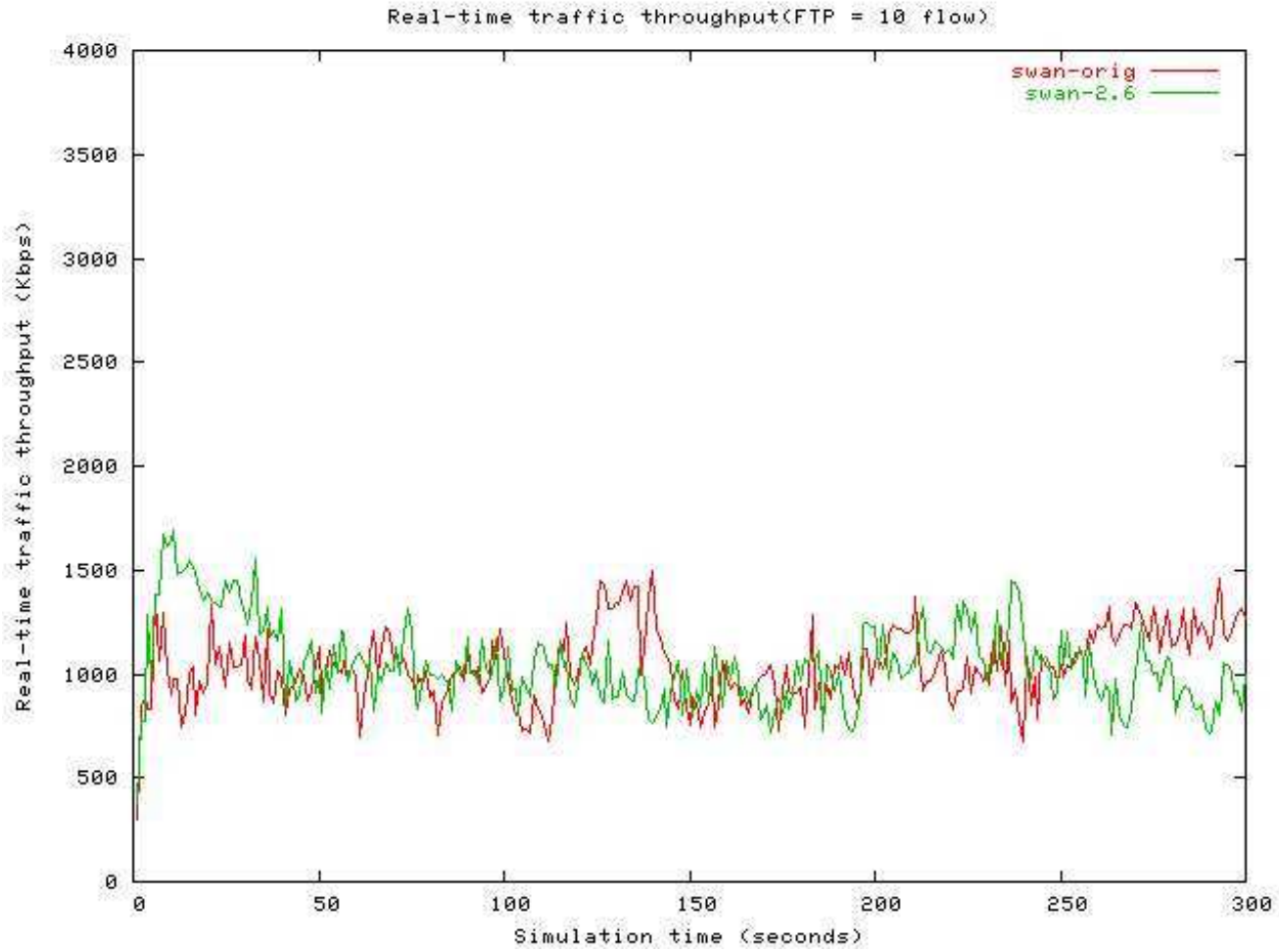

Figure 5.3: Video throughput

Figure 5.4 on the following page shows the simulation results of aggregate

real-time traffic throughput in growing number of background TCP traffic. As shown in figure, the aggregate real-time traffic throughput is slowly decreasing as the number of background TCP flows increase under *swan-orig* and *swan-2.6*. But, the *swan-orig* shows higher aggregate real-time traffic throughput than *swan-2.6* when TCP flow grows from 2 to 10 flows. *swan-2.6* has higher aggregate throughput performance than *swan-orig* when there are 2 or less best-effort TCP flows are present in the network. Since, allowed real-time traffic bandwidth is set to 2600 Kbps under *swan-2.6* compare to 2000 Kbps under *swan-orig*, performance of *swan-2.6* in terms of aggregate real-time traffic throughput is relatively weaker than *swan-orig*.

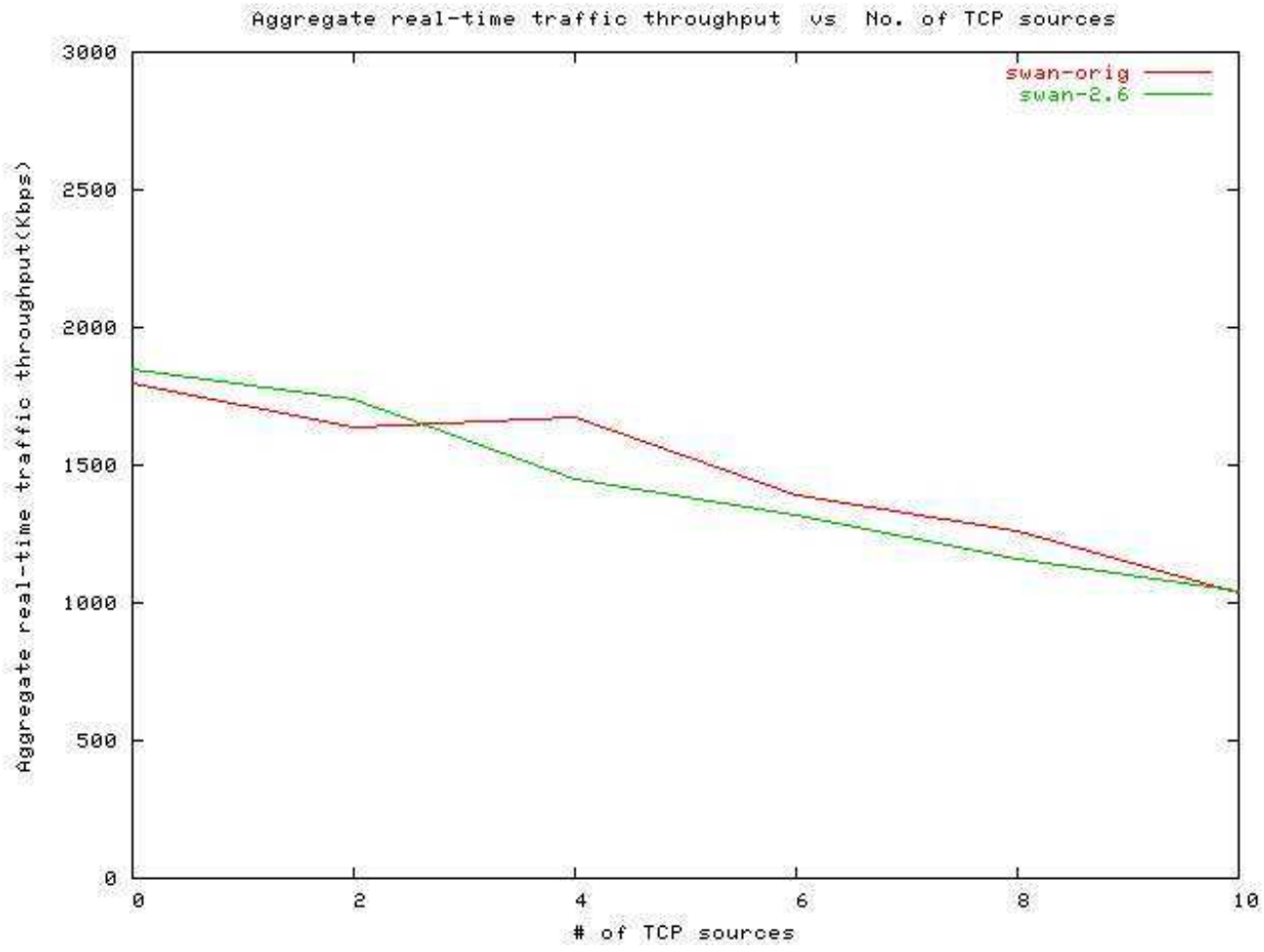

Figure 5.4: Average real-time throughput VS number of tcp flow

Average real-time delay :

Here, we present the result of our simulations for *swan-2.6* and *swan-orig* in terms of average end-to-end real-time traffic delay. As mentioned above, simulations is based on 10 video flows under *swan-orig*, and 13 video flows under *swan-2.6* with varying number of best-effort background TCP flow. The background TCP traffic is modelled as a pair of 2,4,6,8 and 10 FTP flows, and each video flow has a bandwidth rate of 2000 Kbps.

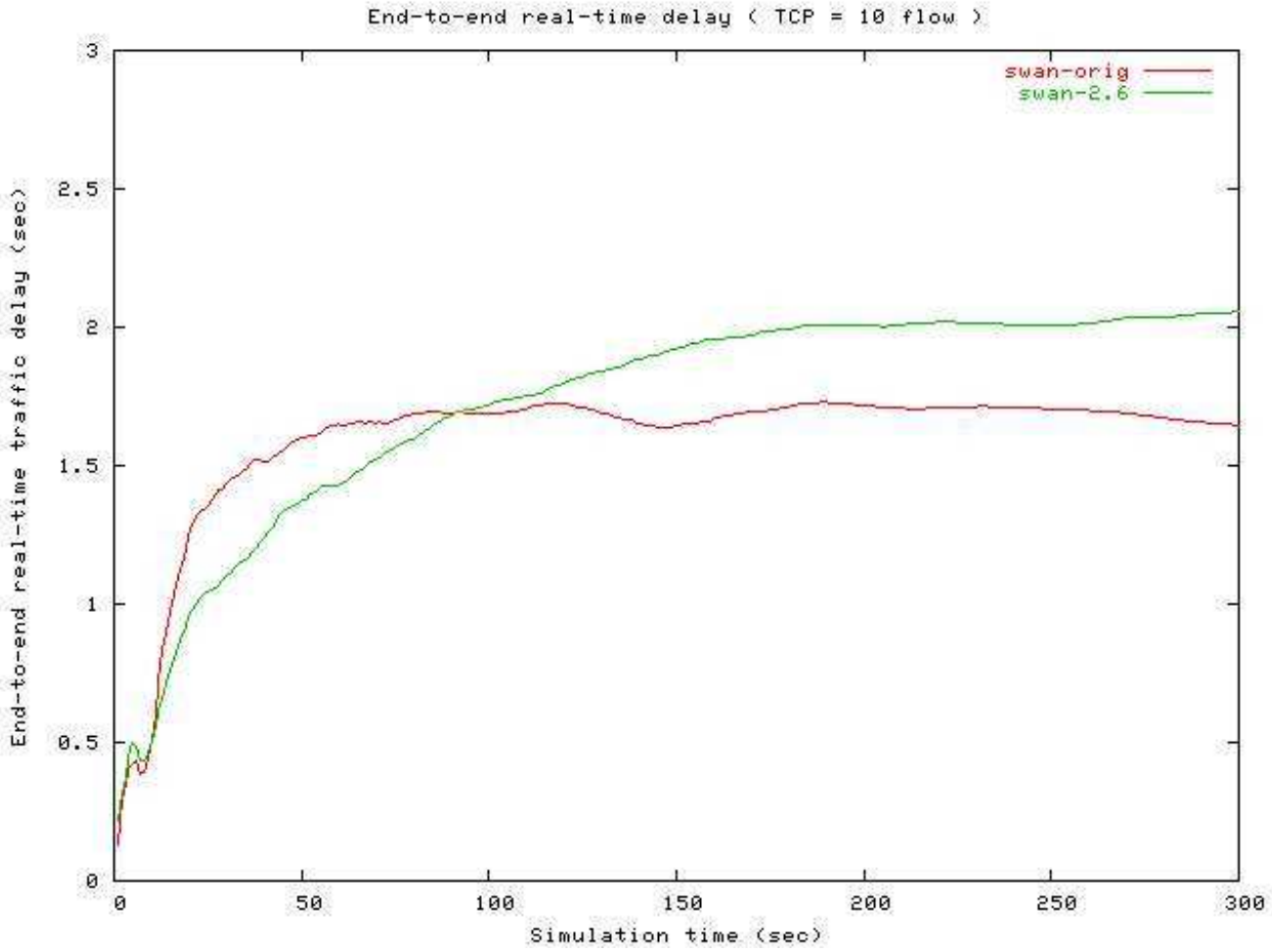

Figure 5.5: End-to-end real-time delay versus simulation period

Fig. 5.5 shows the traces of end-to-end delay of real-time traffic consisting 10 TCP flows. We can observe from fig.5.5 that the end-to-end delay of real-time traffic is lower under *swan-2.6* (i.e when admission control rate = 2.6 Mbps) than under *swan-2.0*(i.e when admission control rate = 2.0

Mbps) only in the first 100 seconds of total simulation period. But , as we can see in figure 5.5, end-to-end delay is increasing linearly from 100 secs onwards under *swan-2.6* where the delay under *swan-orig* is decreasing. This tendency is continuing until the end of simulation period. In another word, *swan-orig* has lower end-to-end real-time delay than *swan-2.6* in more than 70% of simulation period. Also, the highest delay experienced by *swan-orig* lies around 1.5 seconds where *swan-2.6* experience the highest delay of 2.0 seconds.

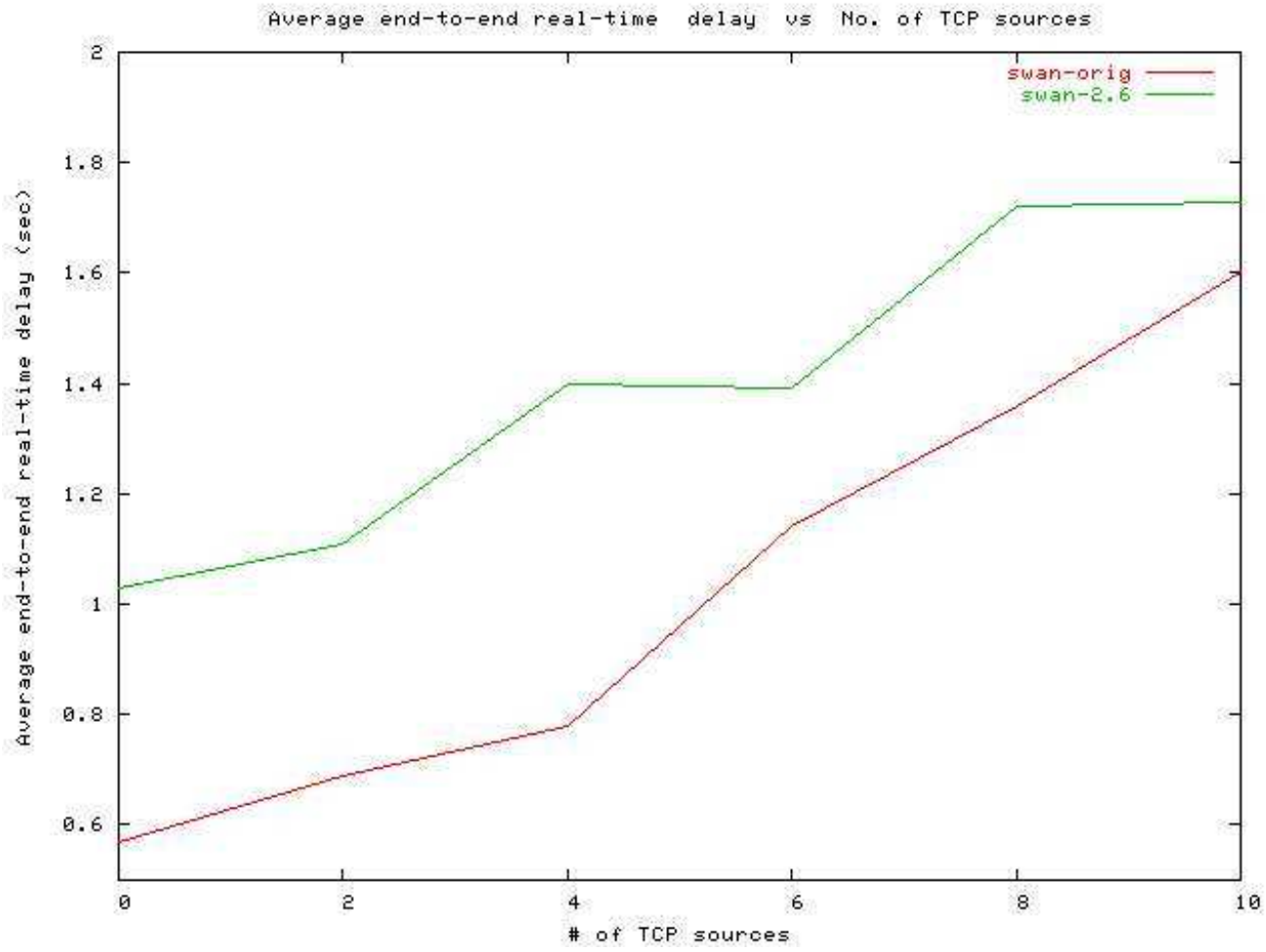

Figure 5.6: Average end-to-end real-time delay vs. number of tcp flow

Figure 5.6 shows average end-to-end real-time traffic delay under growing number of TCP flows. The end-to-end real-time delay increasing constantly in both *swan-orig* and *swan-2.6* when TCP flow increase from 2 to 10

flows. Average end-to-end real-time delay is more than 1.0 sec. under *swan-2.6*, where real-time delay under *swan-orig* is below 0.6 seconds when there are no TCP flows present at all. *swan-orig* consistently maintain the lower end-to-end delay than *swan-2.6*, hence *swan-orig* have better QoS support for real-time traffic. As mentioned before, the higher delay in *swan-2.6* is due to SWAN's ECN probe packets in network because of more admitted flows. SWAN model aimed to achieve higher real-time traffic throughput as well as low end-to-end real-time delay. This objective is fullfilled under *swan-orig* as shown by figs. 5.6 and 5.4. Actually, the high throughput for *swan-2.6* is expected since more real-time flows are admitted. Hence, the more delay will be experienced because of more packets in buffer due to CSMA/CA among nodes. But, the simulation results shows the low throughput and high delay in *swan-2.6* which indicates that higher admission controlrate than originally purposed by SWAN's researchers does't help to enhance SWAN model.

5.4.2 Impact of Admission Control Rate : 1.6 MB

Here, the simulation is based on 8 video flows representing real-time traffic with varying number of best-effort TCP flows. Video traffic is modeled as 2000 Kbps constant rate traffic with a packet size of 512 bytes generating total of 1600 Kbps(1.6 MB), and the best-effort TCP background traffic is modelled as a pair of 2,4,6,8 and 10 greedy FTP flow with a packet size of 512 bytes. The terms ***swan-1.6*** and ***swan-orig*** refers to simulation scenarios, where the SWAN is simulated with admission control rate is assigned to 1600 Kbps (1.6 Mbps) and 2000 Kbps (2.0 Mbps) respectively.

Average real-time delay :

Fig. 5.7 on the next page shows the end-to-end delay of real-time traffic with best-effort TCP traffic consisting of 2 TCP flows. As shown in figure, the real-time delay is steadly higher in *swan-orig* than *swan-1.6* from begining to the end of simulation period. As shown in fig. 5.7, the end-to-end delay of the real-time traffic in *swan-orig* start to increases linearly as the simulation period increases where delay in *swan-1.6* start to decrease correpondingly. The end-to-end delay in *swan-1.6* drops down to zero seconds during the simulation period as contrast to *swan-orig* where real-time traffic experience the delay of nearly 0.3 seconds already in the begining of the simulation.
Figure 5.8 shows simulation results of end-to-end real-time traffic delay based on 10 TCP flows representing best-effort traffic. Here, the end-to-end

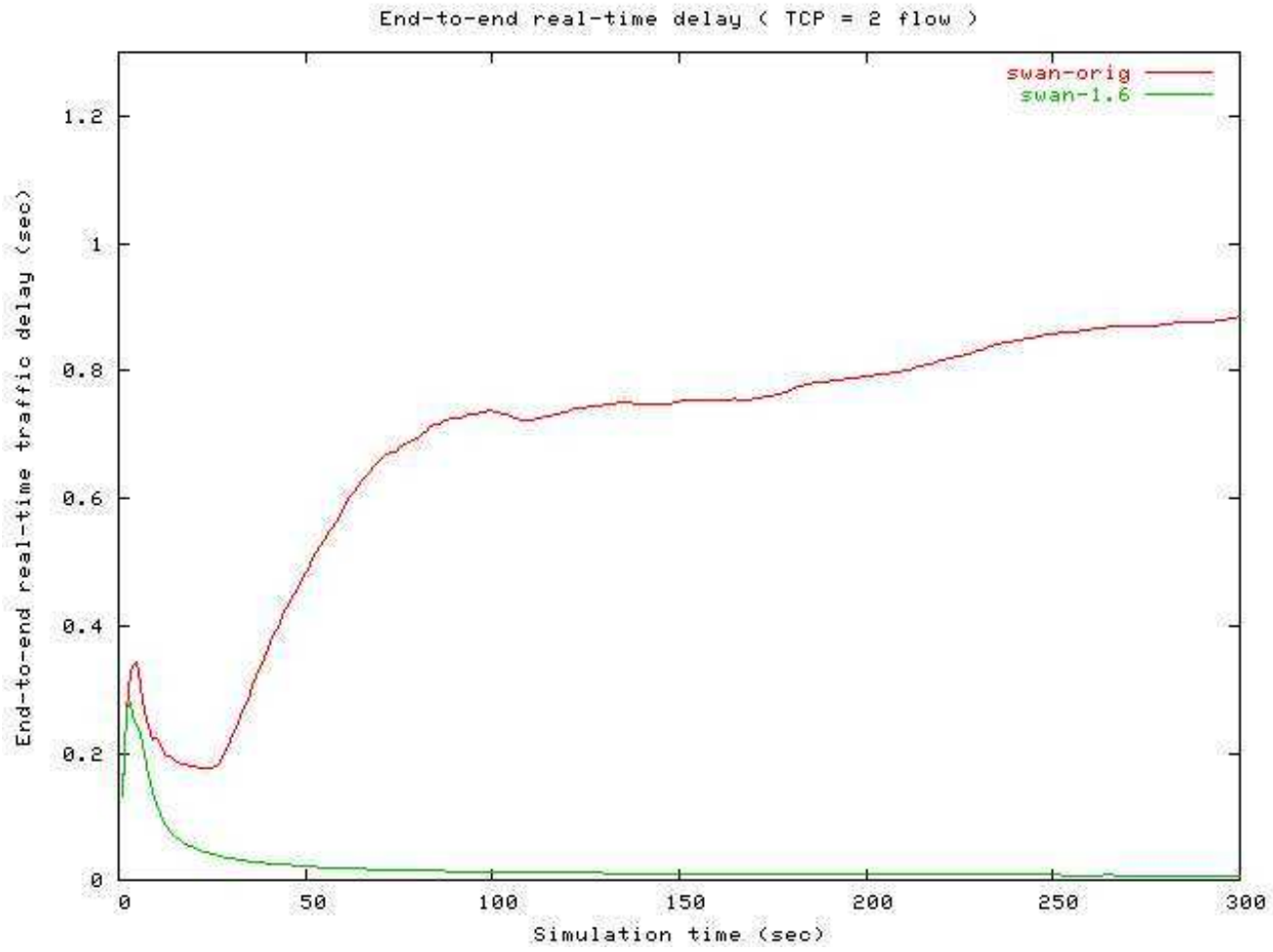

Figure 5.7: Average real-time delay,tcp flow = 2

delay of real-time traffic in *swan-1.6* is consistently higher compare to *swan-orig* throughout the simulation like the simulation based on 2 TCP flows,The average end-to-end delay of real-time traffic in *swan-1.6* lies around 1.0 seconds where the average delay lies above 1.5 seconds in *swan-orig* during simulation.
Figure 5.9 on page 56 shows the traces of average end-to-end real-time delay under growing number of best-effort TCP traffic flows. With no presense of TCP traffic flow, the average delay under *swan-orig* approaches 0.6 seconds while the average delay under *swan-1.6* drops nearly to zero seconds. The average delay of real-time traffic delay in the *swan-orig* grows from 0.6 to above 1.4 sec when the number of TCP sources increases from 2 to 10 flows. In contrast, the average delay of the real-time traffic in

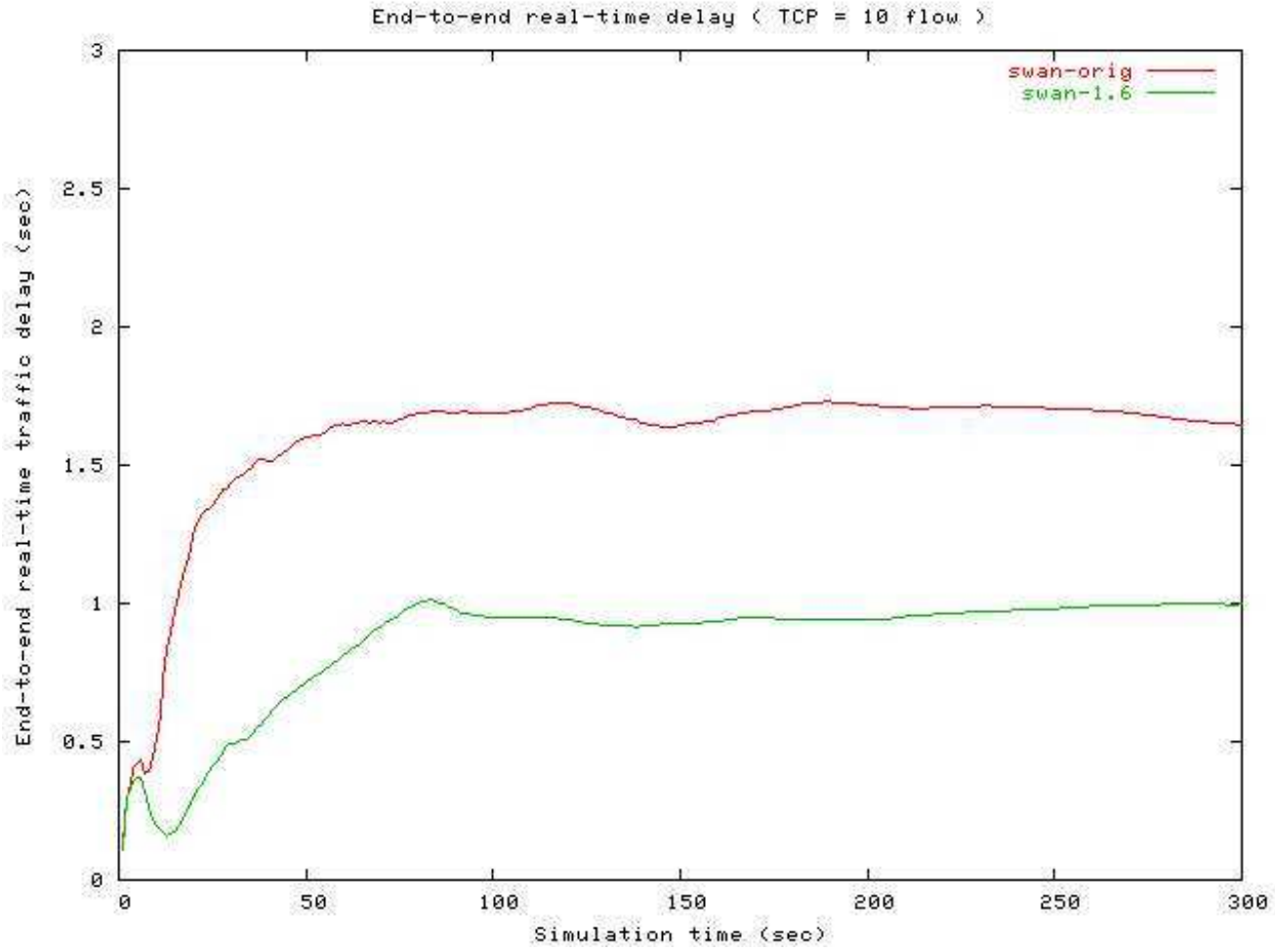

Figure 5.8: Average real-time delay, tcp flow = 10

swan-1.6 grows from 0.02 to 0.86 seconds. The differences in experienced delay could be due to the presence of lower real-time traffic flow. Under *swan-.16*, real-time traiffic bandwidth is limited only to 1600Kbps, hence network will less congested compare to *swan-orig* with real-time traffic bandwidth 2000 Kbps. With less congested network, the chances of initiation of ECN mechanism will also be minimal which effect real-time delay. From simulation results, we can conclude that average delay of real-time traffic in *swan-1.6* is lower than *swan-orig* for growing number of background TCP flows.

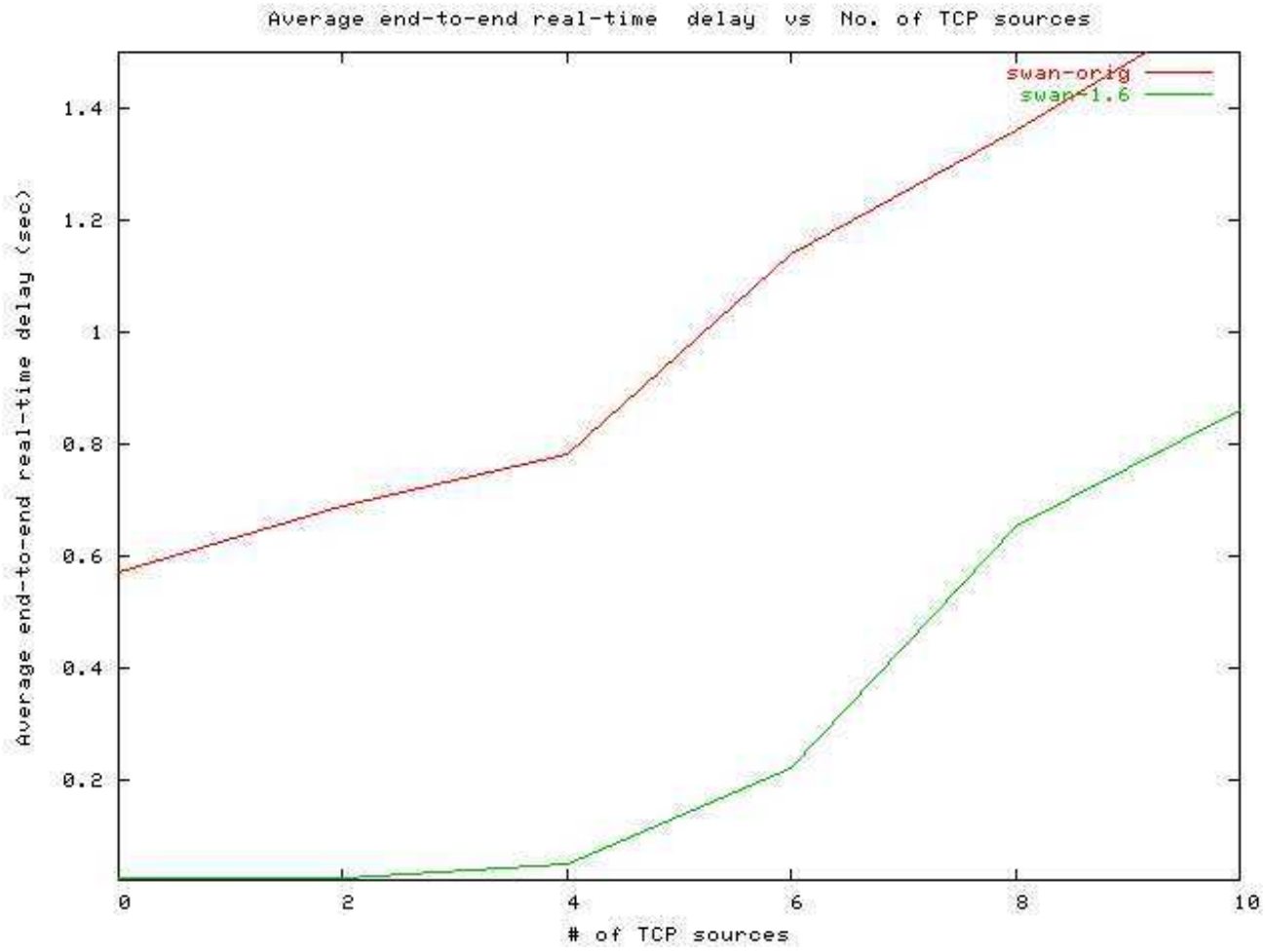

Figure 5.9: Average end-to-end real-time traffic delay vs. number of tcp flow

5.5 Performance of Multihop Scenarios

In this scenario, we consider a simulated multi-hop network with 50 mobile ad hoc nodes. The network area has a rectangular shape of 1500m x 300m that minimizes the effect of network partitioning. AODV [2] is used for routing in the simulated network. Under this simulation scenario, sending-receiving pair nodes does not lie within the transmission range of each other. To reach the destination node ,the packets from sender node have to traversed through intermediate node. The flows travers 2-5 hops between source-destination pairs. Here, we use pause time of zero second i.e. nodes are in constant mobility.

5.5.1 Impact of Admission Control Rate : 2.6 MB

The real-time traffic is modelled as 13 video flows. The background traffic is modelled as a pair of 2,4, 6, 8 and 10 FTP flows. Here, packets have to travel through intermediate nodes to reach destination nodes. In this case, there are total 26 nodes which are source-destination pairs for video traffic and varying number of nodes from 4 to 20 (depending on FTP flows) are transmitting-receiver pair of greedy FTP traffic. At largest simulation scenario (i.e 13 video flow and 10 FTP flow), there are 46 nodes which are either sending and receiving, and rest four nodes can be intermediate nodes.

Real-time traffic throughput

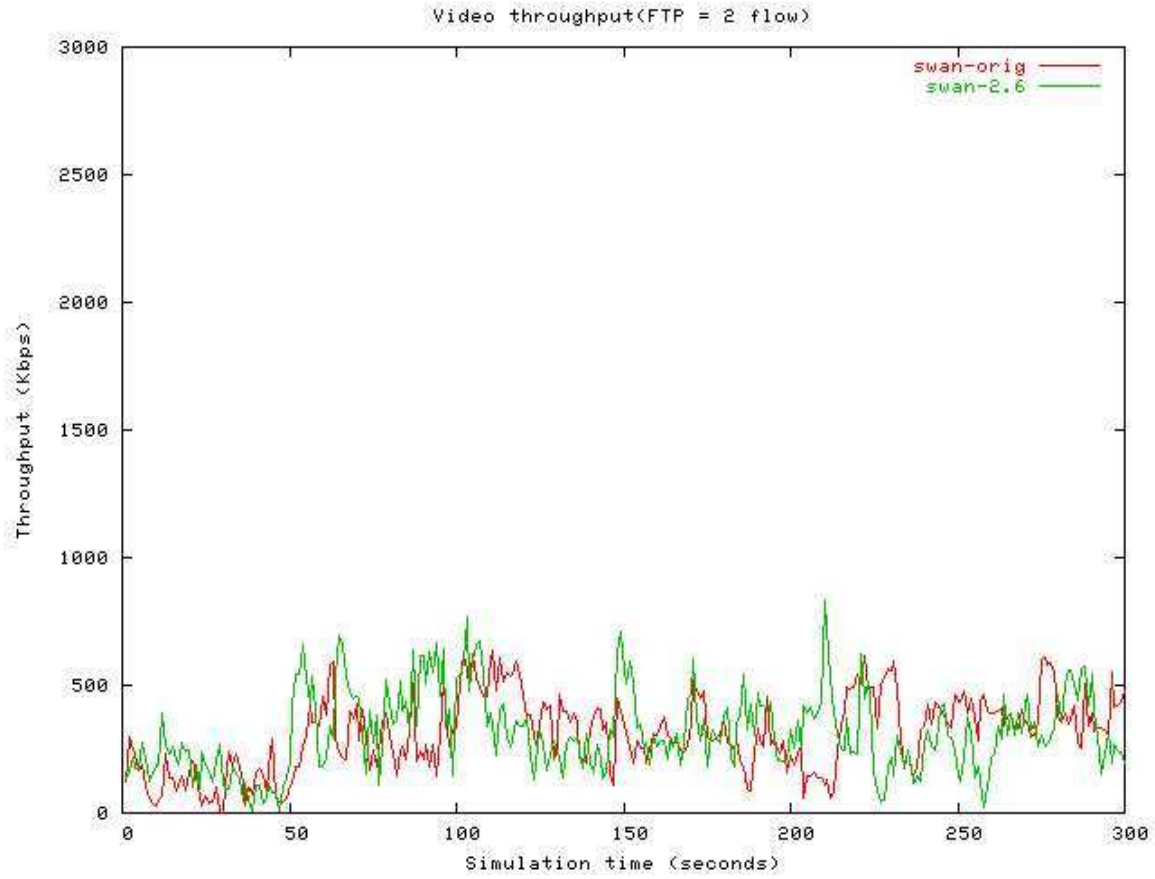

Figure 5.10: Real-time traffic throughput

Figs. 5.10 and 5.11 on the following page show the average throughput of real-time traffic consisting of 2 and 10 FTP flows representing best-effort

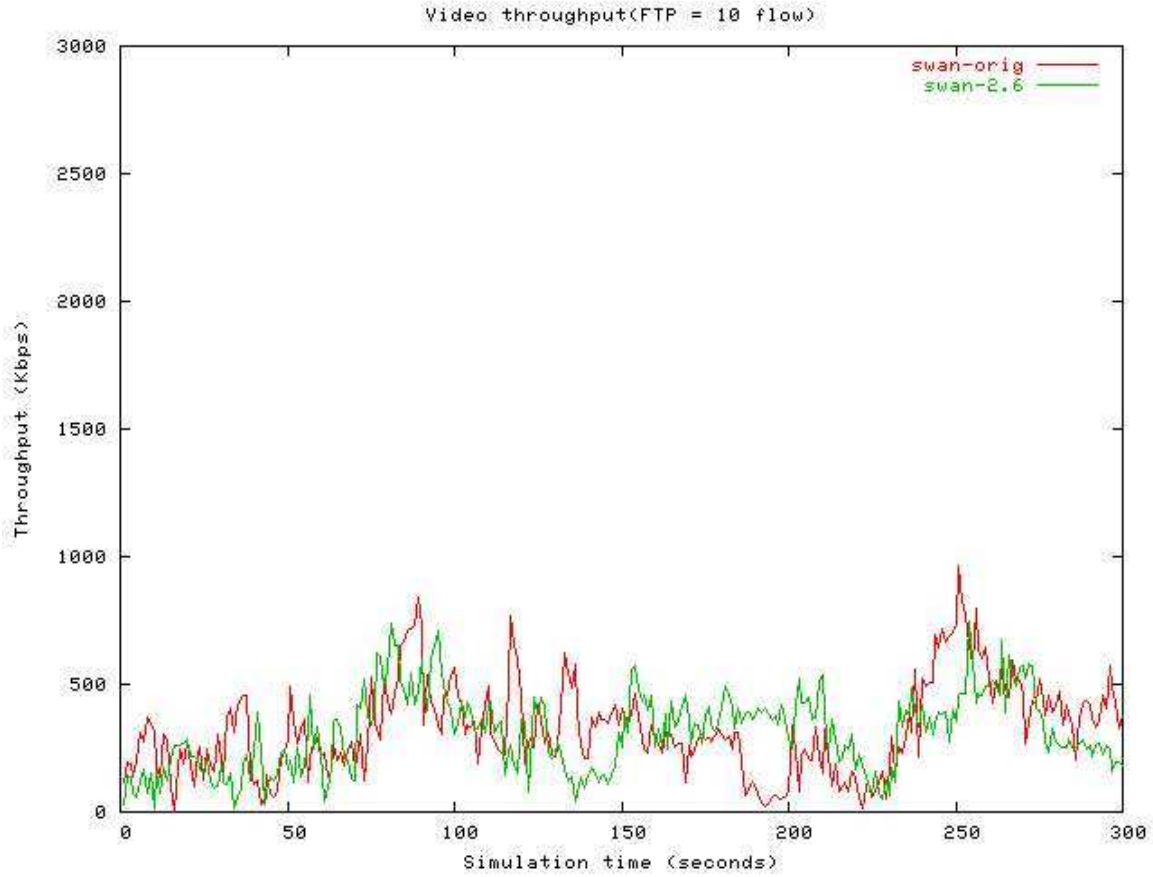

Figure 5.11: Real-time traffic throughput

TCP background traffic respectively. We observe that *swan-2.6* has higher video throughput during the first 35 secs., when simulation consists of only 2 FTP flows, than the simulation based on 10 TCP flows in which *swan-orig* has higher video throughput during approximately the same simulation period.
As shown on these figures, *swan-2.6* has higher video throughput when there is only 2 FTP background flows, and *swan-ori* has higher throughput of real-time traffic under 10 different TCP background flows. The real-time traffic throughput under *swan-2.6* is lower than *swan-orig* in both cases from the second half of simulation period as shown in both figures.

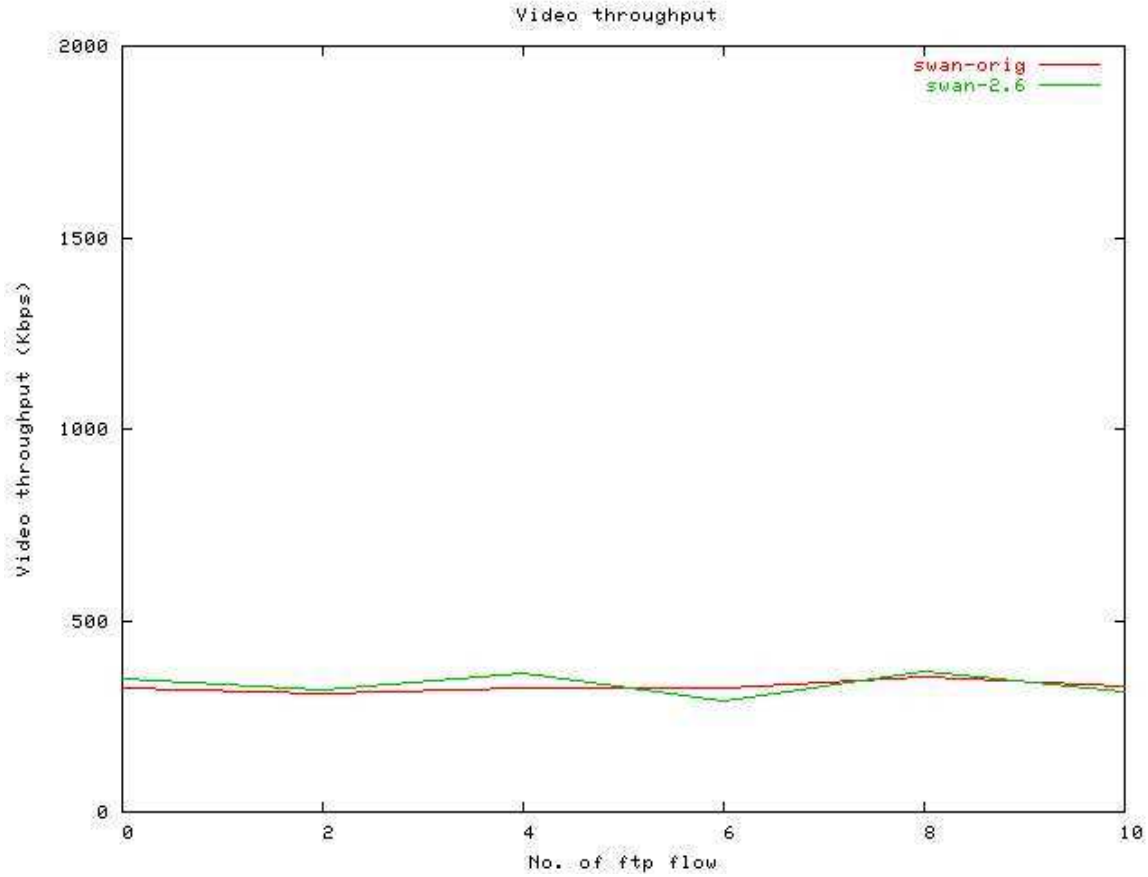

Figure 5.12: Average real-time throughput VS number of tcp flow

The impact on aggregate real-time traffic throughput by growing number of background FTP flows is illustrated in figure 5.12. The real-time traffic throughput under *swan-2.6* is contstantly higher than *swan-orig* when there is less than 6 FTP flows present. As the number of FTP flows increases from 6 to 10, the aggregate real-time traffic throughput fluctuates sharply under *swan-2.6*. In other hand, *swan-orig* shows increasing aggregate real-time traffic throughput as the number of FTP flows increases. So by looking at figure 5.12, we can conclude that *swan-2.6* has higher level of performance throughput than *swan-orig* when there is less best-effort traffic present, and also aggregate real-time traffic throughput in *swan-2.6* is sensitive to FTP flows. *swan-orig* maintain the aggregate real-time traffic throughput well above 300 Kbps, where as the aggregate real-time traffic throughput in *swan-2.6* drops below 300 Kbps.

Average end-to-end real-time delay

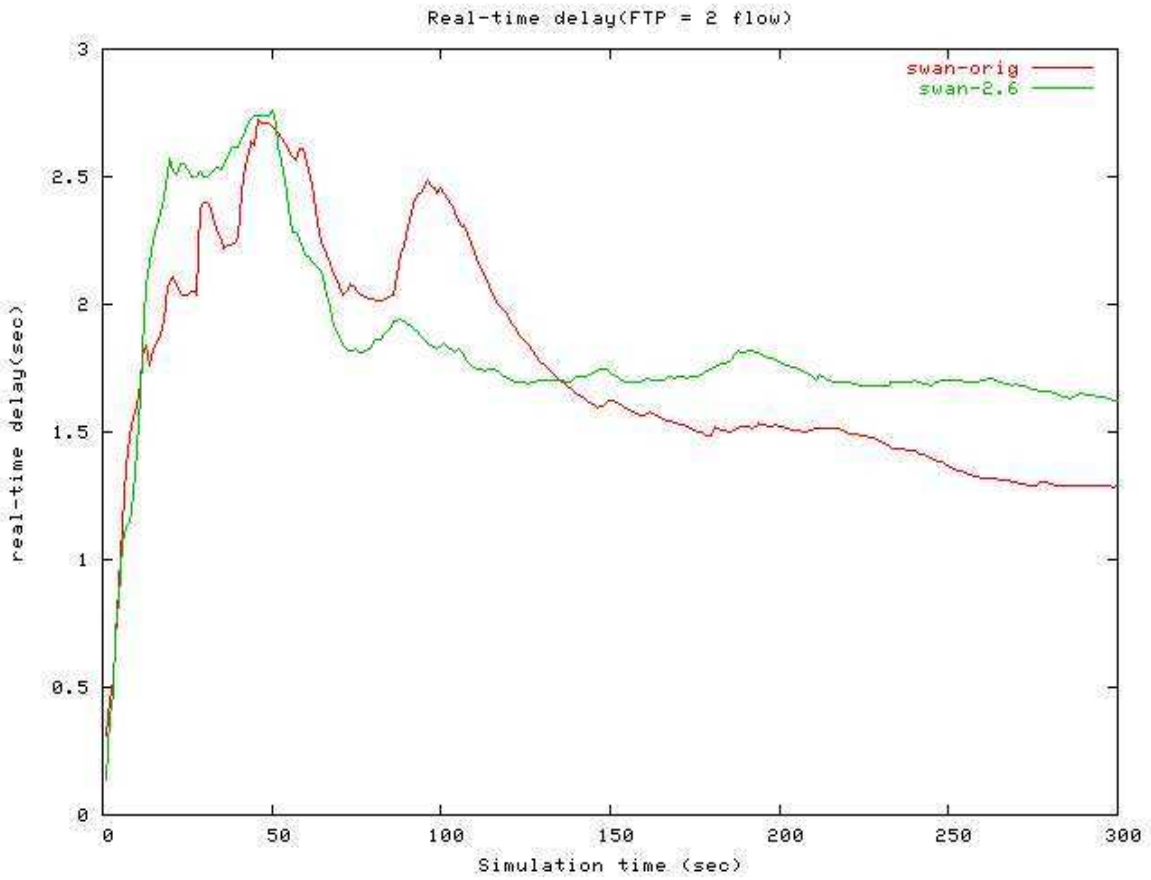

Figure 5.13: Average real-time delay,tcp flow = 2

Fig. 5.13 shows the traces of average end-to-end real-time delay with background TCP traffic consisting of 2 FTP flows. As shown in figure 5.13, *swan-2.6* has lower delay than *swan-orig* until 25 seconds approximately, but end-to-end real-time delay start to increase again under *swan-2.6* until approx. 50 seconds of simulation. Actually, average end-to-end delay is increasing from approx. 25 seconds to 50 seconds of simulation under both simulation environment. As we can observe, the delay under *swan-orig* is higher than *swan-2.6* from 50 to 140 seconds of simulation time. But, there is low end-to-end average real-time delay under *swan-orig* in the rest of simulation period. When simulation ends, average end-to-end real-time delay under *swan-2.6* is around 1.7 seconds and less than 1.5 seconds under *swan-orig*. We can see from figure that *swan-2.6* has lower delay than

swan-orig for only about 100 seconds throughout the total simulation period of 300 seconds, i.e 33% of simulation time.

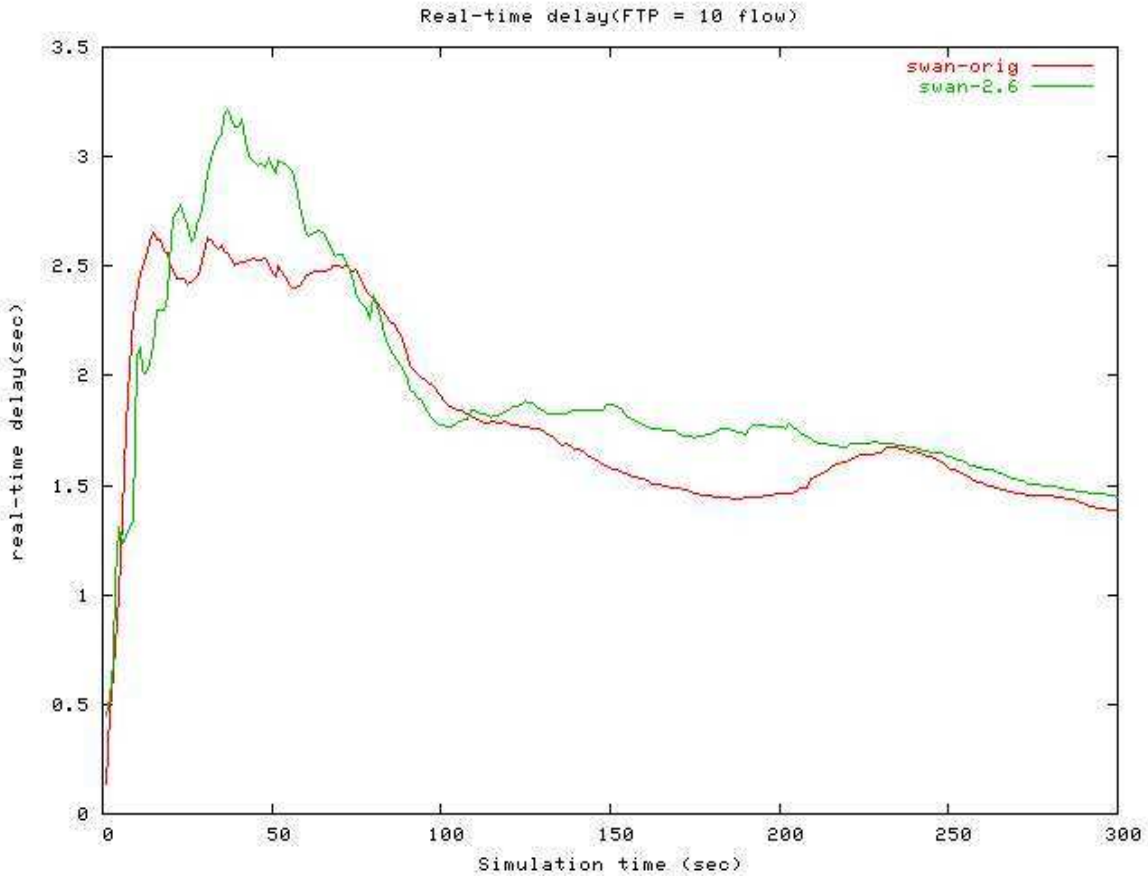

Figure 5.14: Average real-time delay,tcp flow = 10

Fig. 5.14 shows the average end-to-end delay of real-time traffic with background TCP traffic consisting 10 FTP flows. Here, *swan-orig* has lower delay than *swan-2.6* in the begining of simulation, but delay is increasing under both schemes. As shown in figure, the highest delay experienced under *swan-orig* is about 2.7 seconds, where as there is more than 3 seconds delay experienced under *swan-2.6*. In addition, the *swan-2.6* experienced higher real-time delay than *swan-orig* during simulation period.

The impact of growing number of background TCP flows on average end-to-end real-time traffic delay is illustrated in figure 5.15 on page 63. As shown in fig. 5.15 on page 63, the average end-to-end delay of the real-time

traffic in *swan-2.6* is about 1.58 seconds compare to *swan-orig*, where average end-to-end delay is about 1.39 seconds, when there is no any background TCP traffic flows.

We observed from simulation results that the highest delay experienced under *swan-2.6* is high as 1.6 seconds, where the experienced highet delay under *swan-orig* is just 1.4 seconds. The average end-to-end delay of the real-time traffic in *swan-2.6* grows from approx. 1.58 to 1.62 seconds as the number of TCP flows increases from 0 to 2 flows, respectively. In contrast, the average delay of real-time traffic in *swan-orig* is decreasing and remains around 1.39 to 1.25 seconds. In addition, the average delay of the real-time traffic remains consistently below 1.4 seconds in *swan-orig* while the average delay in *swan-2.6* grows above 1.6 seconds.

swan-2.6 actually shows higher level of performance in terms of real-time traffic throughput as shown by figure 5.12 on page 59, specially when there are less than 6 FTP flows. But, the average end-to-end real-time traffic delay is higher under *swan-2.6* than *swan-orig* for same number TCP background traffic flows. Also, *swan-orig* has constantly low average end-to-end real-time traffic delay compare to *swan-2.6* for growing number of best-effort TCP traffic flows. Therefore, *swan-2.6* or allocating bandwidth of 2.6 MB for real-time traffic, does not contribute to enhance SWAN model.

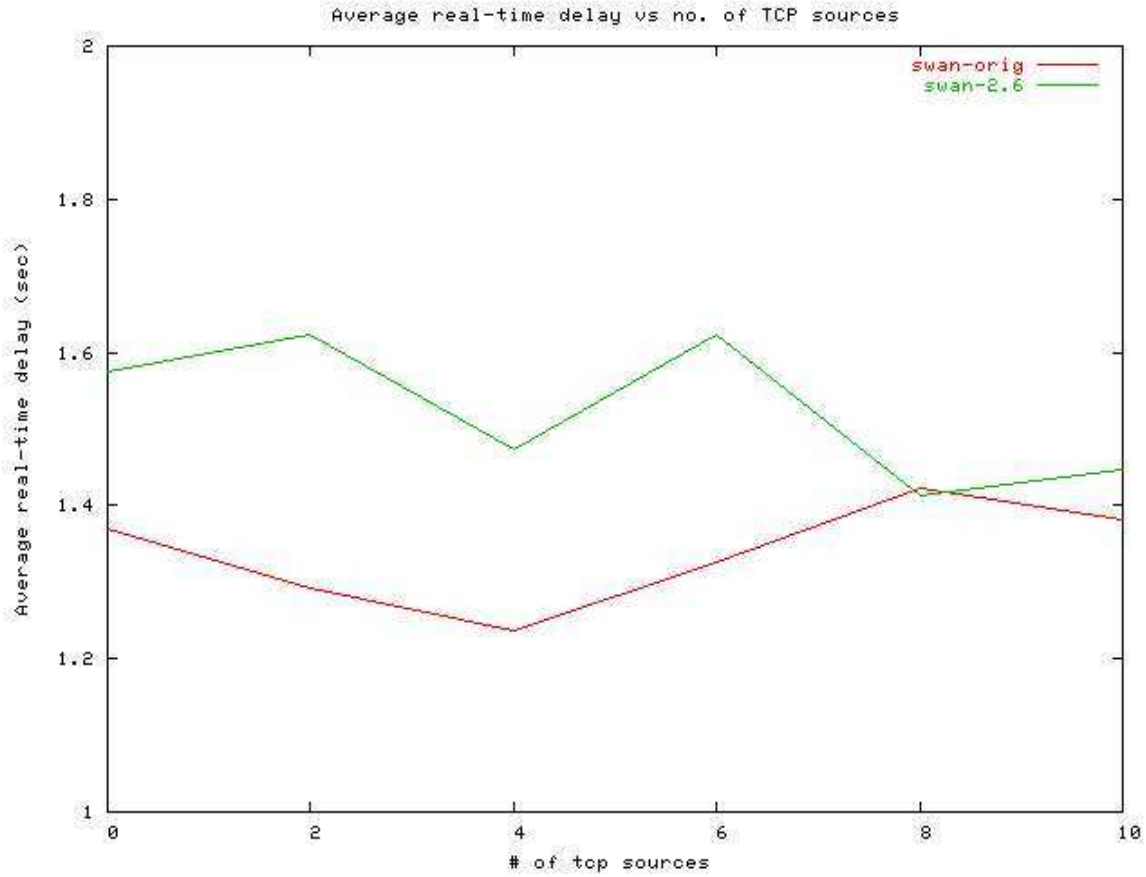

Figure 5.15: Average end-to-end real-time traffic delay vs. number of tcp flow

5.5.2 Impact of Admission Control Rate: 1.6 MB

In this section, we simulate multihop network consisting of 50 ad-hoc nodes with admitted real-time traffic bandwidth set to only 1600 Kbps.i.e 1.6 MB. The real-time traffic is modelled as 8 video flows at a rate of 200 Kbps. Also, the best-effort background traffic is modelled as a pair of 2, 4, 6, 8 og 10 FTP flows. AODV[1111] is used for routing in the simulation. Similarly, sender-receiver nodes does not lie within the transmission range of each other. So packets have to traverse average 2-5 hops to reach the destination.

Real-time traffic throughput

Figure 5.16 shows the average throughput for real-time traffic with 2 FTP flows representing best-effort TCP background traffic. As shown in figure, there is higher average real-time traffic throughput under *swan-1.6* during the first 50 seconds of simulation period, where the throughput under *swan-orig* drops down as low as zero during same period. But, from 50 seconds onwards, *swan-orig* has compartively higher real-time traffic throughput except at the end of simulation period. In addition, real-time throughput under *swan-1.6* drops also down to zero many times during the same period where *swan-orig* shows higher throughput.

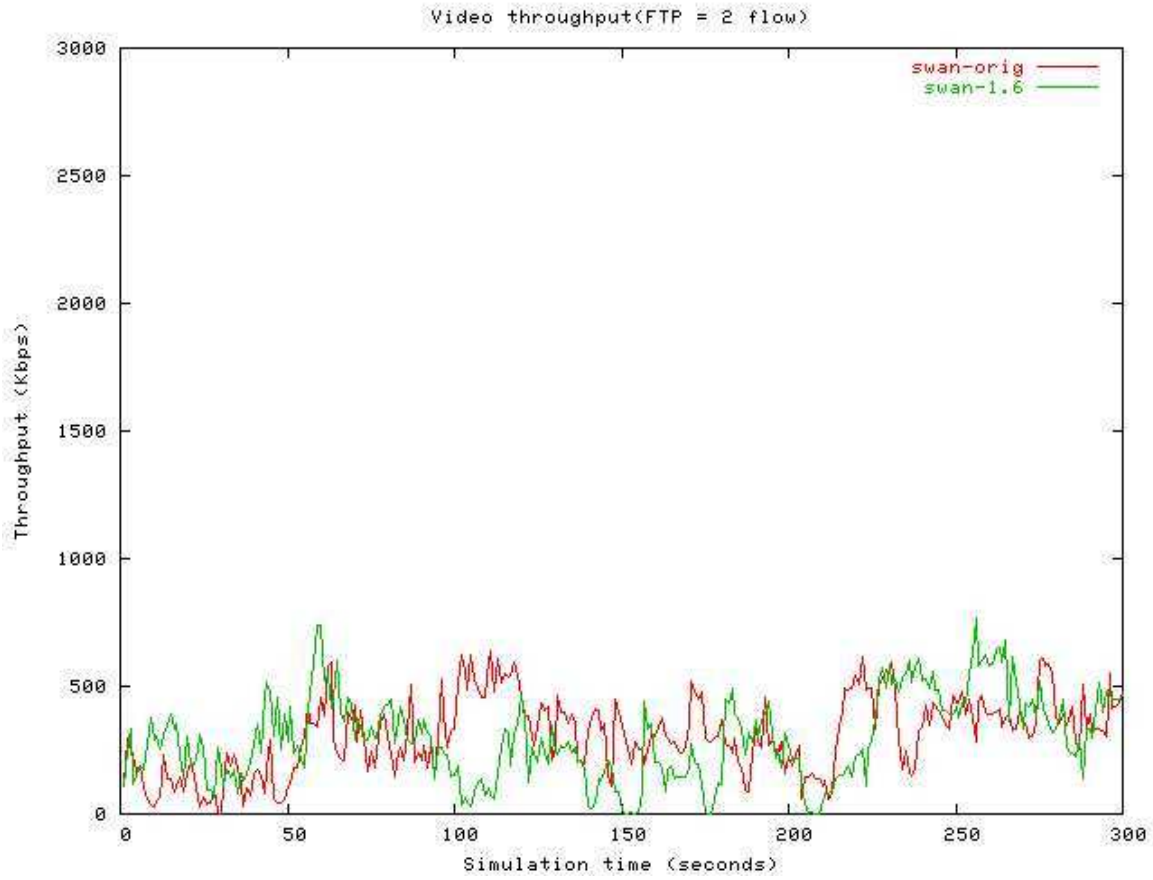

Figure 5.16: Real-time traffic throughput

Fig. 5.17 on the following page shows the real-time traffic throughput when there is 10 FTP flows representing best-effort TCP traffic. Here, in contrast to case when there is only 2 FTP flows, *swan-orig* has higher throughput from the begining of simulation period. By looking at figure **??** on page ??,

it is clear that *swan-orig* has higher throughput in more than 50% of simulation period.

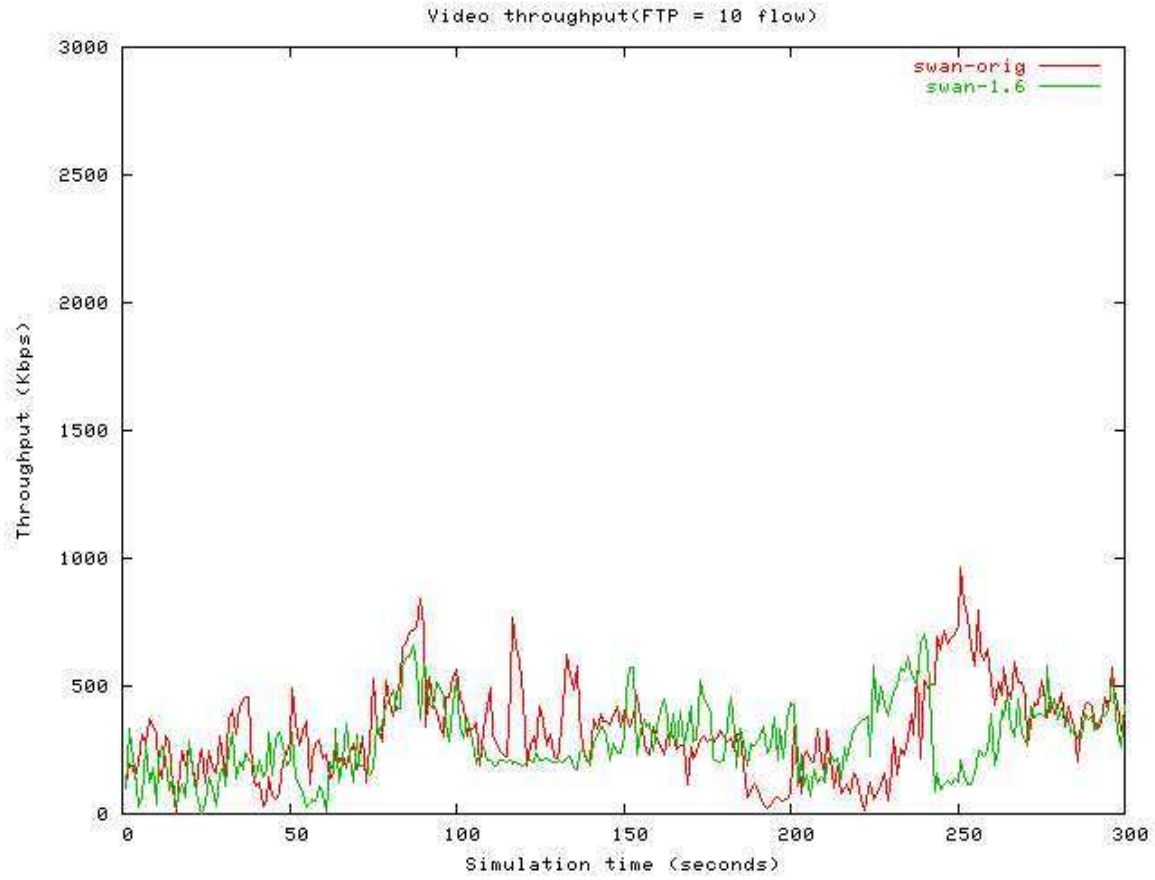

Figure 5.17: Real-time traffic throughput

The impact of growing number of FTP flows on aggregate real-time traffic throughput is illustrated in figure 5.18 on page 67. As shown in figure, *swan-1.6* has higher aggregate real-time traffic throughput than *swan-orig* when there is no any FTP flows present. Also, the aggregate real-time traffic increases, as the number of FTP flows increases from 2 to 6, under *swan-1.6*. This scenario changes as the FTP flow increases from 6 to 10 in which the real-time traffic throughput decreases in *swan-1.6*. In contrast, the *swan-orig* is insensitive to growing number FTP flows since real-time traffic throughput remains unchanged as shown by figure 5.18 on page 67. One of the main objective of SWAN model is to limit the best-effort trafic in order to provide sufficient bandwidth for real-time traffic. And this objective is better served by original proposed swan with only 2 MB

real-time bandwidth, i.e *swan-orig*.

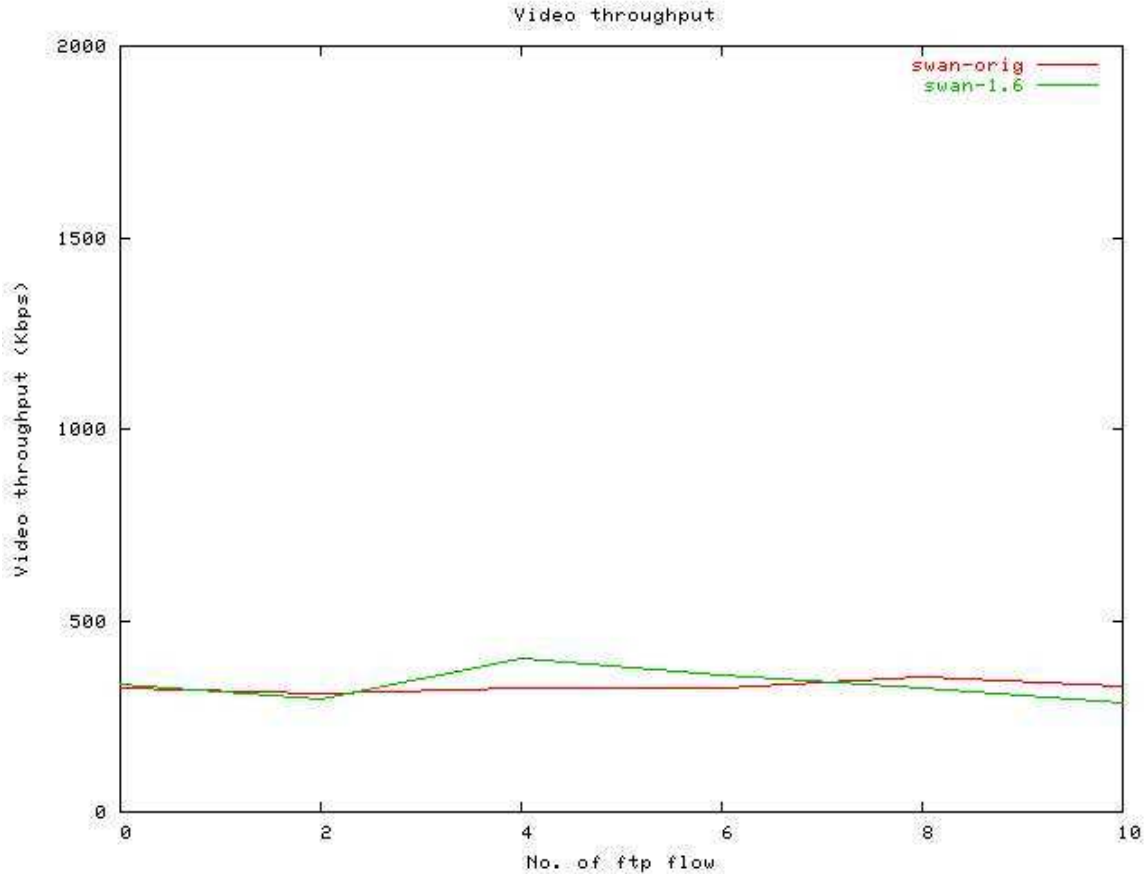

Figure 5.18: Average real-time throughput VS number of FTP flow

Average end-to-end real-time traffic delay :

In this section, we present the simulation results on average end-to-end real-time traffic under varied background best-effort TCP traffic flows.

Figure5.19 shows the traces of average end-to-end delay of real-time traffic with best-effort TCP background traffic represented by 2 FTP flows. Here, we see that *swan-1.6* has low real-time delay until 140 seconds of simulation time and at the same time, *swan-orig* experienced higher delay. But, *swan-orig* experience lower average end-to-end delay than *swan-1.6* in the rest of simulation period.
Vi have different performance scenario when there are 10 background best-effort TCP flows as shown by figure 5.20 on page 69. The *swan-orig* already shows the higher delay from the begining to end of simulation period, whereas *swan-1.6* shows lower end-to-end delay throughout the

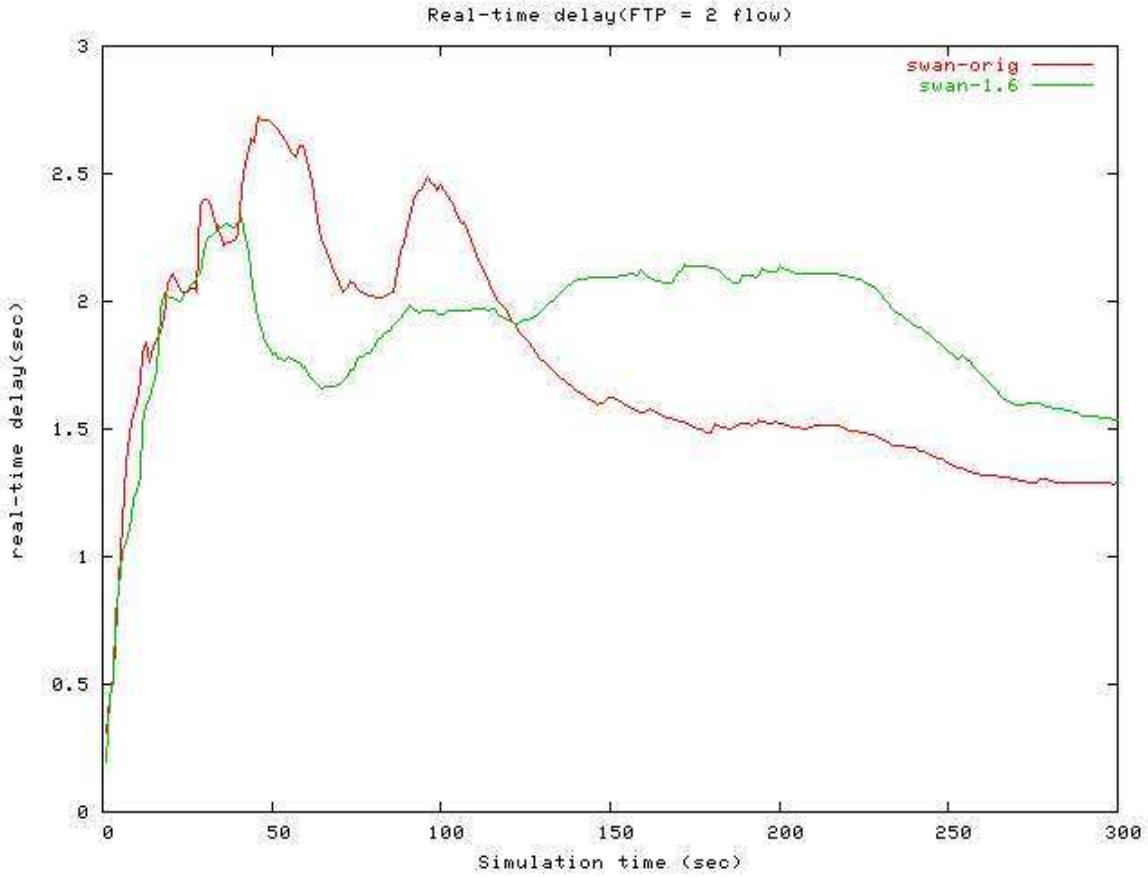

Figure 5.19: Average real-time delay, FTP flow = 2

simulation period. The maximum end-to-end real-time delay experienced under *swan-1.6* is about 2.0 seconds, and under *swan-orig* is more than 2.5 seconds. By looking at fig.5.19 we can assume that *swan-1.6* shows the better performance in terms of real-time traffic delay in the expense of low real-time bandwidth. I can not find the reason why *swan-1.6* shows the higher delay than *swan-orig* in cases where there are only 2 TCP background flow, but shows lower average end-to-end delay compare to situation when there are 10 TCP background flows.

The impact of number of background TCP flows on average end-to-end delay of real-time traffic is shown by figure 5.21 on page 70. The average end-to-end delay under both schemes ,i.e. *swan-orig* and *swan-1.6* is same when there is no any background TCP flows. The average end-to-end delay increases linearly when number of TCP flow increases from 0 to 2 under *swan-1.6*, where there is no any significant delay changes under *swan-orig*.

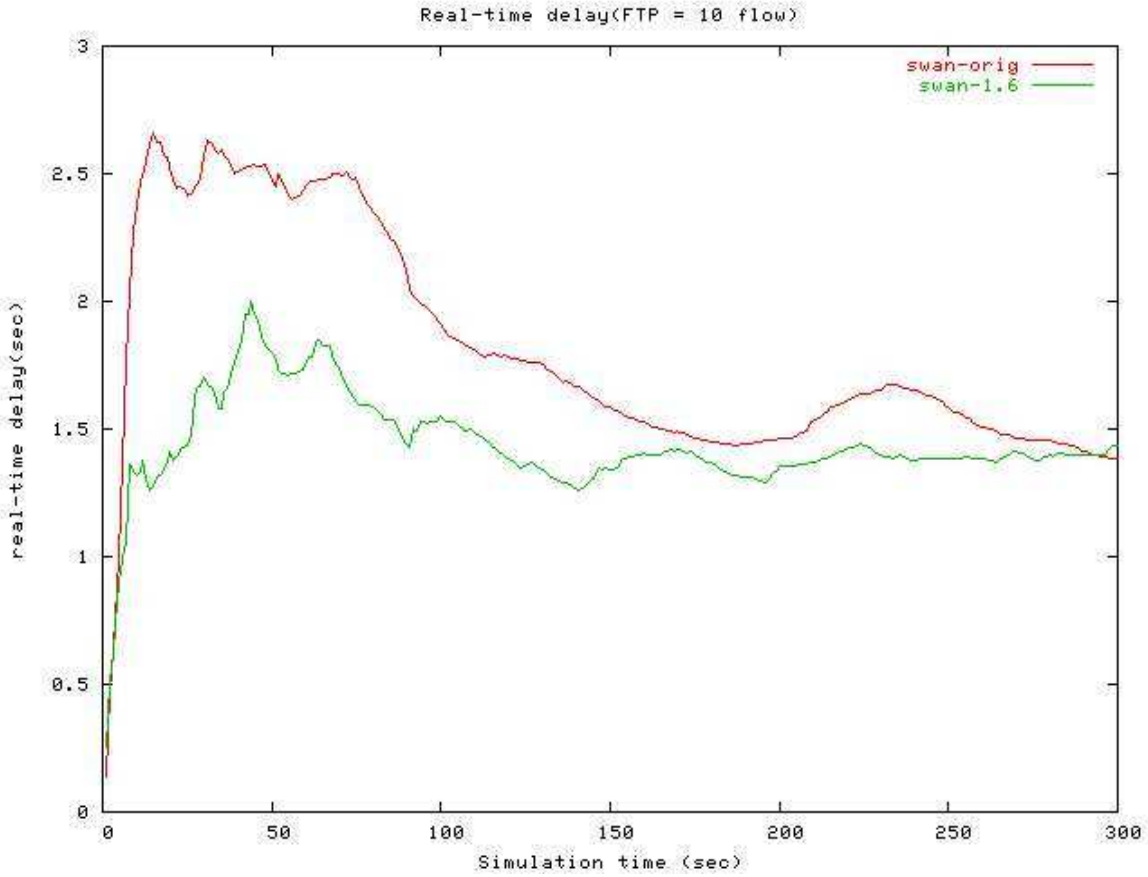

Figure 5.20: Average real-time delay, FTP flow = 10

Fig 5.21 on the following page also illustrate the fact that there is not too much delay variation under *swan-orig* as the number of TCP flows increases, but average delay fluctuates very highly under *swan-1.6*. In fact, there should be less congested network under *swan-1.6* due to limited allowed bandwidth for real-time traffic than *swan-orig* and hence low end-to-end delay .

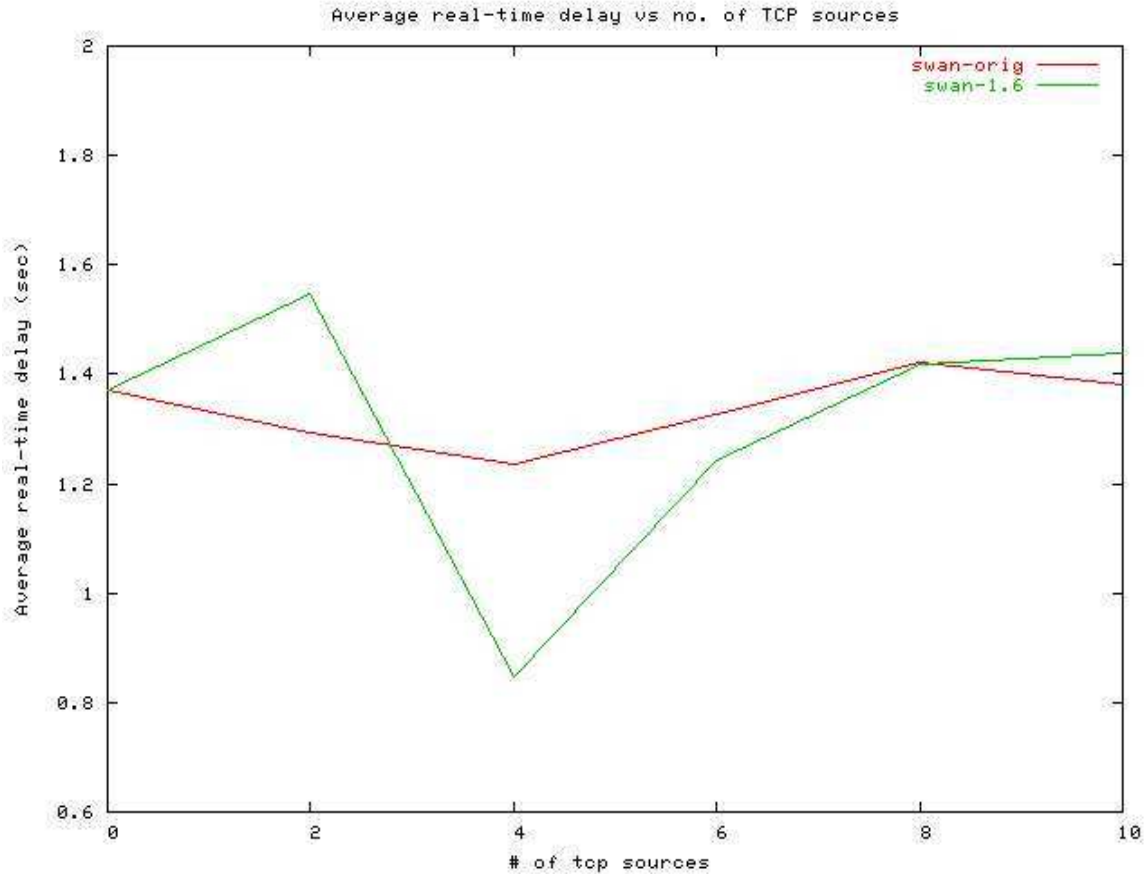

Figure 5.21: Average end-to-end real-time traffic delay vs. number of tcp flow

5.6 Performance of SWAN: 1600 Kbps, 2000 Kbps, 2600 Kbps

Here, we list the performance result of SWAN model in multi-hop scenario with admission control rate(real-time bandwidth) set respectively to 2.0, 2.6 and 1.6 Mbps. As mentioned before,the term *swan-orig*, *swan-2.6* and *swan-16* represent the scenarios where SWAN model is simulated respectively with 2.0(purposed), 2.6 and 1.6 MB as a allowed bandwidth for real-time traffic. Also , simulation is based on 10 , 13 and 8 video flows with varied FTP flows representing best-effort TCP traffic under *swan-orig*, *swan-2.6* and *swan-1.6* respectively .

The simulated multihop network composed of 50 mobile nodes moving with

maximum speed of 10 m/s. The background TCP traffic is modelled as a pair of 2, 4, 6, 8 and 10 FTP flows. The AODV used as routing protocol.

TCP traffic flows and real-time traffic delay :

We compare the results of all three scenarios on average real-time traffic throughput and average end-to-end real-time traffic delay in growing number of background TCP traffic flows. The figure 5.22 shows the traces

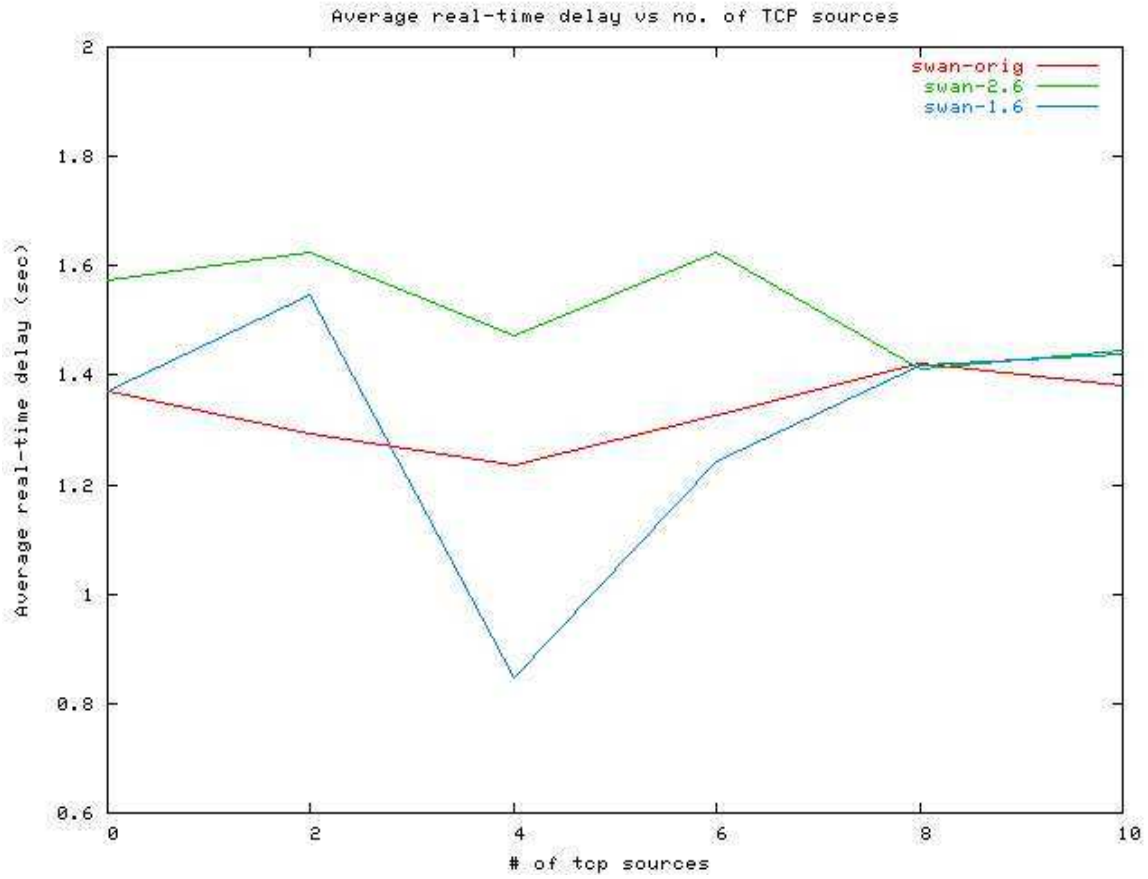

Figure 5.22: Average end-to-end real-time traffic delay vs. No. of FTP flow

of average real-time traffic delay under growing number of background best-effort TCP flows. The average delay is around 1.39 seconds under *swan-orig* and *swan-1.6* and 1.59 seconds under *swan-2.6*, when there are no any background TCP traffic present in the network. Average delay under *swan-1.6* highly fluctuates as we observe from figure that the that the average delay increases from 1.39 seconds to approximately 1.50

seconds when TCP flow increases from 0 to 2. *swan-2.6* exhibits higher delay than *swan-orig* as well as *swan-1.6*. *swan-orig* maintains the delay below 1.4 seconds regardless of background TCP flows. *swan-orig* shows average lower delay than *swan-2.6* and *swan-1.6*, when there are 10 TCP flows, where as *swan-2.6* and *swan-1.6* exhibits equal level of delay.

Average real-time traffic throughput :

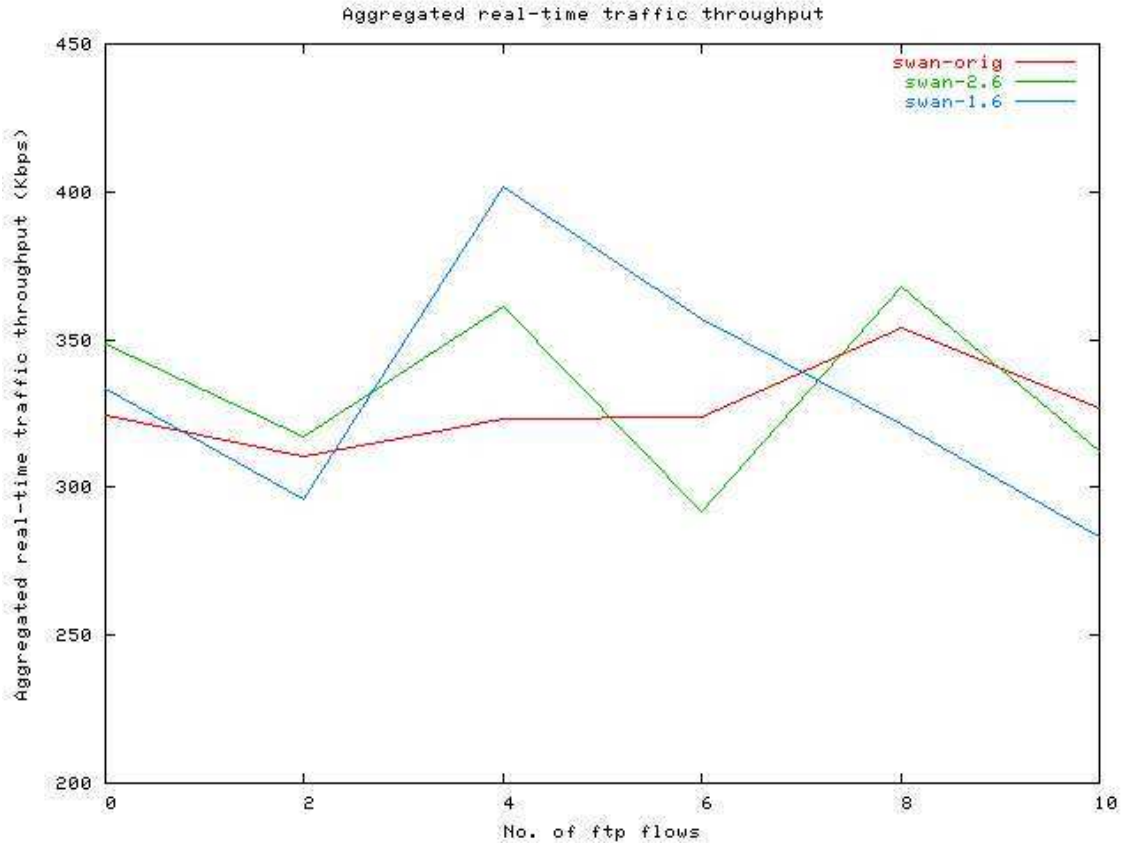

Figure 5.23: Aggregate real-time traffic throughput vs. No. of FTP flow

Fig. 5.23 shows the average real-time traffic throughput under the growing number of FTP flows representing best-effort background TCP traffic. As illustrated in figure 5.23, *swan-orig* has constantly higher aggregate real-time traffic throughput, where as the throughput fluctuates in both *swan-2.6* and *swan-1.6*. The throughput performance under *swan-1.6* varies from 400 Kbps as its highest to below 300 Kbps, where as the average throughput lies between 300 - 350 Kbps under *swan-orig* and

swan-2.6. Hence, the *swan-1.6* has very sharp variation on throughput performance. By looking at figure5.23, it becomes clear that *swan-orig* outperforms both *swan-2.6* and *swan-1.6* in terms of aggregate real-time traffic throughput , when there is higher presence of best-effort traffic. SWAN aims to provide better service to real-time traffic by restricting best-effort traffic. Given the facts above, it indicates that *swan-orig* shows better QoS support for real-time traffic.

Mobility and end-to-end real-time traffic delay :

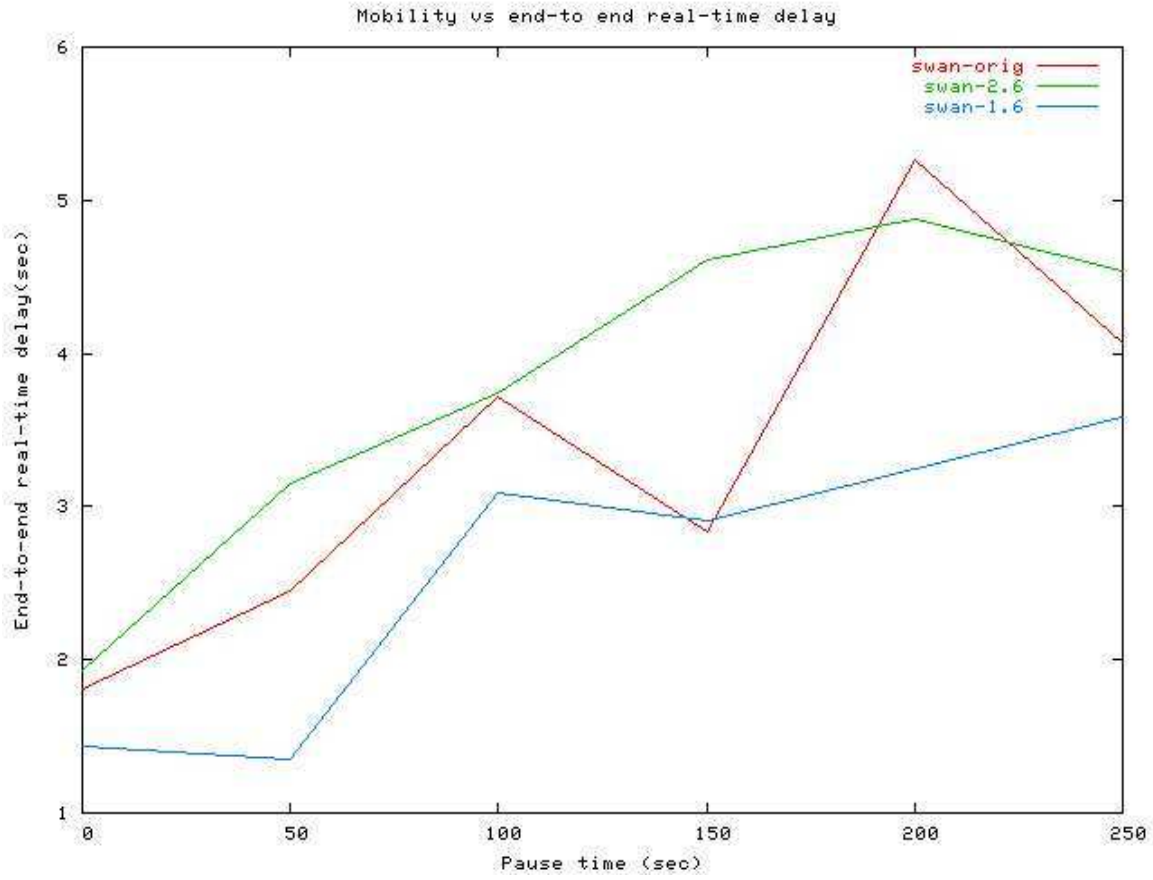

Figure 5.24: Average end-to-end delay of real-time traffic versus Mobility

Here, we show the impact of mobility on average end-to-end real-time traffic delay and average throughput of both real-time UDP traffic and best-effort TCP traffic under *swan-orig*, *swan-26* and *swan-16* respectively. The simulated network is the same as multihop scenario with the addition

of introduction of mobility. The movement of the nodes is defined by the random-way point model, where each mobile node selects a random destination and moves with a random speed up to a maximum speed of 10 m/sec and pauses for a given "pause time", when destination is reached. The simulation is based on fixed number of FTP flows representing best-effort TCP traffic with 10, 13 and 8 video flows in *swan-orig*, *swan-26* and *swan-16* respectively. The number of best-effort TCP flows comprises 10 FTP flows. The simulation is run for period of 300 seconds. All data traffic is simultaneously initiated at the start of the simulation and ended when the simulation ends. As depicted in figure 5.24, the average

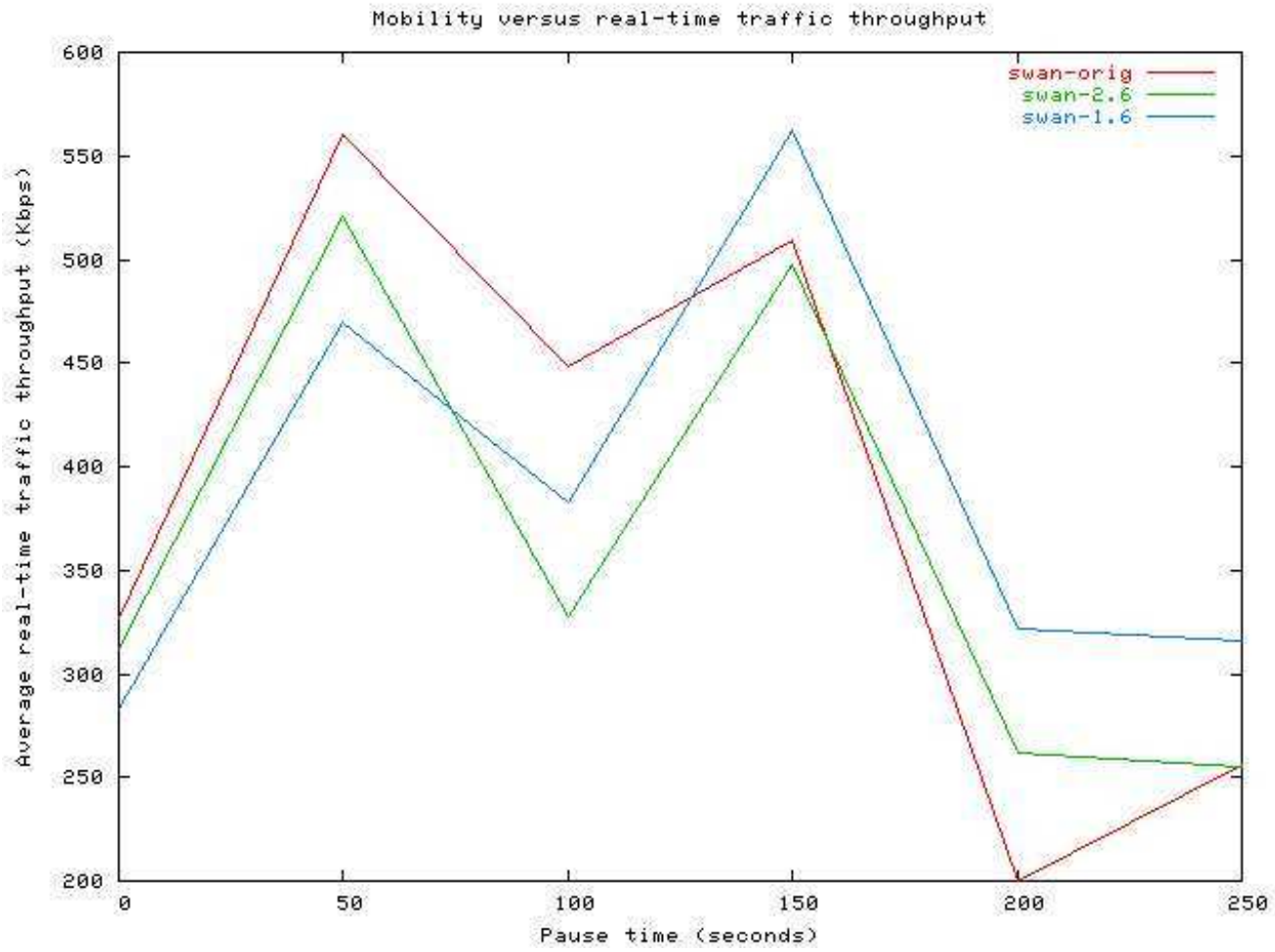

Figure 5.25: Aggregate real-time traffic throughput versus Mobility

end-to-end dealy of real-time traffic increases slowly as a mobility increases in all three simulations. *swan-1.6* has lower average real-time delay than

swan-2.6 and *swan-orig* regardless of mobility scenario. We also noticed that average real-time traffic delay under *swan-orig* fluctuates sharply compare to *swan-2.6* and *swan-1.6*. As shown in figure 5.24, the average end-to-end real-time delay in *swan-1.6* is around 1.5 seconds under the highest mobility, where as there is about 1.7 and 1.9 seconds of delay under *swan-orig* and *swan-26* respectively. Eventhough delay in *swan-orig* varies sharply, end-to-end delay experienced is lower than *swan-2.6*. The average end-to-end delay in *swan-1.6* remains around 2.5 seconds in all mobility scenarios, while the average end-to-end delay grows up to 4.0 and 5.0 seconds respectively in *swan-26* and *swan-orig*.
Both *swan-orig* and *swan-2.6* tends to have higher performance under higher mobility condition in contrast to the *swan-1.6* where the higher throughput performance is achieved in lower mobility condition. As illustrated in figure 5.25, the aggregate throughput rate in *swan-orig* is higher than *swan-2.6* and *swan-1.6* under higher mobility condition only. As the mobility decreases, the aggregate real-time throughput also decreases under *swan-orig* compare to *swan-orig* and *swan-2.6*. *swan-orig*

Chapter 6

Evaluation and Analysis

In this chapter, we present our analysis and conclusion as well as present an overview of scheme purposed to enhance original SWAN model.We also present our purposal as a bid of enhancing original SWAN model .

6.1 Analysis of admission control rate on Single Hop Scenario

The real-time traffic throughput performance is variable under both *swan-2.6* and *swan-orig*, when there is 2 background best-effort TCP flows present. But, the *swan-2.6* shows slightly higher real-time traffic throughput than *swan-orig* on aggregate in this simulation. When there are 10 TCP background flows present, there is almost no difference in throughput performance between *swan-2.6* and *swan-orig*. *swan-2.6* has higher aggregate real-time traffic throughput than *swan-orig* only when there are 2 or less TCP background flows present. When the TCP background flows increases, *swan-orig* exhibits higher aggregate throughput performance.

Given the fact that *swan-2.6* has been allocated more bandwidth for real-time traffic, it should have accomodated more flows thereby should have higher throughput performance. In another hand, admitting more flows also contribute to trigger ECN mechanism which in turn generate probe packets to sender node, and thus more traffic will be added to network. And this will affect the performance on throughput. Hence, the *swan-orig* is insensitive to the growing number of best-effort background traffic and has better QoS support for real-time flow than *swan-2.6*. By looking at figure 5.4, we can conclude that *swan-2.6* have aggregate

real-time traffic rate of 1700 Kbps, where aggregate real-time traffic throughput under *swan-orig* is around 1900 Kbps under growing number of TCP background traffic flows.

The average end-to-end real-time traffic delay is slowly increasing under both *swan-orig* and *swan-2.6*. But, *swan-2.6* shows consistently higher average real-time traffic delay than *swan-orig* for growing number of background TCP traffic flows. Moreover, *swan-2.6* experienced average delay of above 1.0 seconds, where the *swan-orig* experienced delay of approx. 0.57 seconds, when best-effort TCP traffic is not present in simulated network. The average end-to-end real-time traffic delay increased linearly in *swan-2.6* from 1.0 seconds to almost 1.7 seconds, when TCP flows increases from 2 to 10, respectively. In contrast, the average delay of real-time traffic in *swan-orig* remains around 0.58 to 1.5 seconds. The maximum average delay experienced in *swan-orig* and *swan-2.6* is around 1.5 seconds and above 1.7 seconds respectively.

Hence, *swan-orig* outperforms *swan-2.6* both in terms of aggregate real-time traffic throughput and average end-to-end real-time delay. The sum of the throughputs achieved by all nodes in the network can not exceed the theoretical maximum throughput of the nework [18]. As mentioned in [18], the theoretical maximum throughput of IEEE 802.11 network with nominal data rate lies around 2067 Kbps. So, assigning higher throughput which lies beyond the achievable throughput do not contribute to achieve good traffic performance. So, this fact explain the reason why *swan-2.6* show lower aggregate real-time traffic throughput than *swan-orig*. And this also explain why SWAN researchers choosen to have 2000 Kbps as maximum allowd bandwidth for real-time traffic.

swan-1.6 shows higer real-time traffic throughput than *swan-orig* when there is no any background best-effort TCP traffic present. As the number of best-effort TCP traffic flows increases, the *swan-1.6* has lower real-time traffic throughput than *swan-orig*. *swan-1.6* outperforms *swan-orig* both in real-time traffic throuhgput and end-to-end real-time delay, when there is no background TCP traffic is present in the network. In today's wireless network, both best-effort data traffic and real-time traffic are present, and *swan-1.6* demonstrates weakness in such a network environment. *swan-orig* has constant real-time traffic throughput regardless of growing background TCP flows than *swan-1.6*.

swan-2.6 has lower aggregate real-time traffic throughput and higer end-to-end real-time traffic delay. And *swan-1.6* lower real-time traffic

throughput and lower real-time delay. The real-time traffic throughput and end-to-end real-time traffic delay are equally important QoS parameters in the ad hoc networkss. Having better performance on either of these two parameters will partly provide QoS support.

6.2 Analysis of admission control rate on Multi Hop Scenario

swan-2.6 shows variation in aggregated real-time traffic throughput performance under growing number of FTP flows. Eventhough, *swan-2.6* higher throughput performance when there are less than 6 FTP flows present, it has weaker throughput performance when there are 10 FTP flows. Hence, *swan-2.6* shows better throughput performance when there is fewer best-effort TCP traffic flows present. This can be concluded by looking at figure 5.14 and 5.13, where *swan-2.6* shows better performance, when there are only 2 TCP flows present than in case where, there are 10 TCP flows present. The average end-to-end real-time traffic delay experienced by *swan-2.6* lies above or around 1.5 seconds, where as *swan-orig* maintains end-to-end delay below or equal to 1.4 seconds. Also, the *swan-2.6* is less scalable with respect to TCP traffic. The simulation results indicates that the increasing admission rate control upto 2600Kbps does not help to enhance purposed SWAN model (*swan-orig*).

swan-orig shows better real-time traffic througput than *swan-1.6*. Also, *swan-orig* is insensitive to the growing number of background TCP traffic flows, where the real-time traffic throughput fluctuates in *swan-1.6*. Also, the end-to-end real-time traffic delay fluctuates highly in *swan-1.6* than *swan-orig* under growing number of background TCP traffic flows. Both *swan-2.6* and *swan-1.5* demonstrates weaker performance than *swan-orig* in terms of real-time throughput and end-to-end real-time traffic delay in network scenario where higher number of TCPtraffic are present.

All three simulation results exhibits high variation on aggregate real-time throughput under different mobility condition. The aggregate real-time traffic throughput under *swan-orig* is higher than *swan-2.6* and *swan-1.6* under higher mobility condition only. At the lowest mobility condition, the *swan-1.6* outperforms both *swan-orig* and *swan-2.6* in terms of aggregate real-time throughput and average end-to-end delay.

The analysis of our simulations results revealed that neither *swan-2.6* nor *swan-1.6* outperforms purposed SWAN model (i.e. *swan-orig*) with respect to both real-time traffic throughput rate and real-time traffic delay implicating the fact that by increasing or decreasing admission rate control of real-time traffic does not enhance SWAN model. SWAN's researchers have choosen 2000 Kbps as maximum allowed bandwidth for real-time traffic in the network. According to results present in [18], this value is the highest achievable throughput in IEEE 802.11 networks. Our simulation results show that *swan-orig* has goog performance than other two simulations i.e. *swan-2.6* and *swan-1.6*. Given the facts mentioned above, we can assume that SWAN with admission control rate 2000 Kbps is optimal to provide QoS support for real-time traffic.

Since, best-effort traffic serves as a buffer zone for real-time traffic, this model does not work well in scenarios where most of the traffic is real-time in nature. An admitted real-time flow may encounter periodic violations in its bandwidth requirements. In the worst case, it may have to be dropped or be made to live with downgraded best-effort service. Hence, the local rate control of best-effort traffic mechanism alone may not be sufficient to fully support real-time traffic. Considering the fact that different real-time traffic have different bandwidth requirement, static bandwidth allocation mechanism is not effectiv solution to provide QoS support. SWAN model is vunerable to problems related to mobility and false admission, a new mechanism or functionality is needed to enhance orginal SWAN model. In the following section, we present overview of our purposal as well as other purposed mechanisms to enhance SWAN .

6.3 Our proposal

As we have mentioned earlier, the admission rate for real-time traffic is pre-defined value by SWAN researcher and based on empirically estimated available bandwidth at each node intermediate node. Having high value for admision rate contributes too poor performance of real-time flows while choosing low value may results in the denial of real-time flows. At the same time, the channel capacity is not optimally utilized.

As a enhancement to SWAN, we propose an adaptive admission rate control in which more real-time flows will be accepted for the same channel data rate, while the per hop delays are still guaranteed. According to this scheme, we set the targeted per hop delay as the threshold. When a real-time flow comes, if the measured available bandwidth is not enough,

the node will first check the current per hop delay. If the delay is below threshold, the node will increase its admission rate to allow the requesting flow being accepted, instead of rejecting the flow as SWAN did. If the experienced per hop delay has already reached the threshold , the node will set its current admission rate down.

6.4 Related work

This section presents a overview of a mechanism, called ESWAN [16], purposed to enhance congestion recovery of SWAN model. This mechanism aimed to decrease the frequent occurrence of congestion of real-time flows by using destination-based regulation. ESWAN[16] introduces the term *expired bandwidth*, which is the bandwidth consumed by highly delayed packets. According to this mechanism, destination based regulation relies on destination nodes to detect an increase in expired bandwidth, then regulating each flow accordingly, and thus restricting the wasting of network resources.

In ESWAN, the destination-based approach introduces **EBR(β)**, and **EDR(δ)** as two important parameters to measure RT flow quality, where **EBR(β)** is defined as the percentage of effective to received real-time bandwidth at a destination node for a specific real-time flow over a period of time **T** where as **EDR(δ)** is defined as the percenatage of average effective packet delays at a destination node to MAPD, over a period ot time T.*MAPD* is threshold packet delay value which triggers packets drops at destination application, and is flow specific value known to the destination node. **EBR** values close to 1 indicate high flow quality and values close to 0 indicate inefficient bandwidth usage, where as, **EDR** values close to 1 indicate a acceptable quality of a flow eventhough high delay and values close to 0 indicate higher QoS of a flow.

According to ESWAN scheme, if a sufficient number of packets arrive at a destination node with packet delays higher than *MAPD*, over a period of time T, the destination node detects a limited QoS condition ($\beta < \beta_i$), and issues a regulate message to the relevant source node. The destination node is also responsible for recovery behavior and it is implemented as follows: An intermediate node marks all RT packets wih CE using the ECN bits should the node experience congestion. And as soon as the condition ($\delta >= \delta_i$) is true at any destination nodes nodes have to issue regulate message. At the same time all other destination nodes have to wait for time T, if packets keep arriving marked with CE. Eventually, congestion is

resolved by removing flows with higher relative delay δ first .

6.5 Future work

During the work on this thesis, we have identified several aspects and issues around SWAN model, which would be interesting to look into as a continuation of our work :

- In SWAN, end-to-end delays are not considered in measuring the quality of real-time flows, but only the per link delays. The real-time flows becomes more sensitive the more apart sender-receiver pairs are. SWAN could be implemented which incoporates end-to-end delays while providing QoS service to real-time flows.
- There exists many different type of data traffic with different characteristics and different requirements in terms of delay, bandwidth etc. SWAN treats all UDP traffic with equal priority and enqueues all real-time packets into single FIFO queue without any distinction, which is not sufficient for a complete QoS solution for the network. Hence, there is need of new advance scheduling mechanism in SWAN.
- We also find that SWAN's bandwidth estimation is not accurate due to hidden and exposed terminal phenomenon. SWAN's rate measurement is based on the listening of the traffic generated by neighbouring nodes, but rate measurement did not take account of bandwidth consumed by hidden terminal, and also estimate available real-time traffic rate incorrectly due to exposed terminal. In the recent SWAN model, there is no mechanism to include hidden traffic into consideration. Hence, any new QoS model for wireless ad-hoc networks that has bandwidth estimation mechanism should incorporate this issue.
- SWAN's QoS scheme is based on real-time and best-effort. It will be feasible to implement SWAN as a class-based QoS model, which could be able to incorporate any kind of traffic with varying QoS requirement. Admission control mechanism in SWAN is based on probe message, which adds extra traffic in the network and utilizes already limited bandwidth. SWAN provides soft QoS service and SWAN could be implemented with adaptibale bandwidth provisioning instead of static bandwidth provisioning. This would certainly help SWAN to seen as a hard QoS model. SWAN's admission control mechanism add extra traffic due to probe message in the network.

- The congestion control mechanism in SWAN is based on ECN algorithm, which is not fairly implemented. Also, bandwidth consumed by delayed packets also contributes to effect QoS service for real-time flows.

List of Figures

Appendix A

TCL Script files for SWAN ns-2 Simulation

```
#
# ======================================================================
# Traffic Model #
# ======================================================================
set pareto1 [new RandomVariable/Pareto]
$pareto1 set avg_ 10
set pareto2 [new RandomVariable/Pareto]
$pareto2 set avg_ 1.0
$pareto2 set shape 1.2
set uniform [new RandomVariable/Uniform]
$uniform set min_ 0
$uniform set max_ 1
set udp_cnt 0
set tcp_cnt 0

global ns_ node_ opt uniform udp_cnt
set j [expr $stime + [$uniform value]
set udp_($udp_cnt) [new Agent/UDP]
$ns_ attach-agent $node_($src) $udp_($udp_cnt)
set null_($udp_cnt) [new Agent/Null]
$ns_ attach-agent $node_($dst) $null_($udp_cnt)
set cbr_($udp_cnt) [new Application/Traffic/CBR]
$cbr_($udp_cnt) set packetSize_ 80
$cbr_($udp_cnt) set interval_ 0.02
$cbr_($udp_cnt) set random_ 1
$cbr_($udp_cnt) set maxpkts_ 15000
```

```
$cbr_($udp_cnt) attach-agent $udp($udp_cnt)
$ns_ connect $udp_($udp_cnt) $null_($udp_cnt)
$ns_ at $j "$cbr_($udp_cnt) start"
$ns_ at 1000 "$cbr_($udp_cnt) stop"
puts "at $j SRC($src) DST($dst) Voice($udp_cnt)"
incr udp_cnt}
proc create-video-connection { src dst stime {proc create-voice-connection {
src dst stime} {
global ns_ node_ opt uniform udp_cnt
set j [expr $stime + [$uniform value]
set udp_($udp_cnt) [new Agent/UDP]
$ns_ attach-agent $node_($src) $udp_($udp_cnt)
set null_($udp_cnt) [new Agent/Null]
$ns_ attach-agent $node_($dst) $null_($udp_cnt)
set cbr_($udp_cnt) [new Application/Traffic/CBR]
$cbr_($udp_cnt) set packetSize_ 80
$cbr_($udp_cnt) set interval_ 0.02
$cbr_($udp_cnt) set random_ 1
$cbr_($udp_cnt) set maxpkts_ 15000
$cbr_($udp_cnt) attach-agent $udp($udp_cnt)
$ns_ connect $udp_($udp_cnt) $null_($udp_cnt)
$ns_ at $j "$cbr_($udp_cnt) start"
$ns_ at 1000 "$cbr_($udp_cnt) stop"
puts "at $j SRC($src) DST($dst) Voice($udp_cnt)"
incr udp_cnt}
proc create-video-connection { src dst stime {
 global ns_ node_ opt uniform udp_cnt
 set j [expr $stime + [$uniform value]
set udp_($udp_cnt) [new Agent/UDP]
$ns_ attach-agent $node_($src) $udp_($udp_cnt)
set null_($udp_cnt) [new Agent/Null]
$ns_ attach-agent $node_($dst) $null_($udp_cnt)
set cbr_($udp_cnt) [new Application/Traffic/CBR]
$cbr_($udp_cnt) set packetSize_ 512
$cbr_($udp_cnt) set interval_ 0.02
$cbr_($udp_cnt) set random_ 1
$cbr_($udp_cnt) set maxpkts_ 15000
$cbr_($udp_cnt) attach-agent $udp_($udp_cnt)
$ns_ connect $udp_($udp_cnt) $null_($udp_cnt)
$ns_ at $j "$cbr_($udp_cnt) start”
$ns_ at 1000 "$cbr_($udp_cnt) stop"
```

```
puts "at $j SRC($src) DST($dst) Video($udp_cnt)"
incr udp_cnt
}
proc create-ftp-connection { src dst stime} {
 global ns_ node_ opt uniform tcp_cnt
 set j [expr $stime + [$uniform value] ]
 set tcp_($tcp_cnt) [new Agent/TCP/Reno]
 $ns_ attach-agent $node_($src) $tcp_($tcp_cnt)
set null_($tcp_cnt) [new Agent/TCPSink]
$ns_ attach-agent $node_($dst) $null_($tcp_cnt)
 set ftp_($tcp_cnt) [new Application/FTP]
 $ftp_($tcp_cnt) set packetSize_ 512
$ftp_($tcp_cnt) set window_ 32
$ftp_($tcp_cnt) set windowInit_ 16
 $ftp_($tcp_cnt) set maxpkts_ 1000000
$ftp_($tcp_cnt) attach-agent $tcp_($tcp_cnt)
$ns_ connect $tcp_($tcp_cnt) $null_($tcp_cnt)
 $ns_ at $j "$ftp_($tcp_cnt) start"
 $ns_ at 1000 "$ftp_($tcp_cnt) stop"
 puts "at $j SRC($src) DST($dst) FTP($tcp_cnt)"
incr tcp_cnt
}
#
# Generate Traffic #
create-video-connection 0 1 0
create-video-connection 2 3 0
create-video-connection 4 5 0
create-video-connection 6 20 0
create-video-connection 7 8 0
create-video-connection 9 27 0
create-video-connection 11 26 0
create-video-connection 10 15 0
create-video-connection 12 16 0
create-video-connection 13 25 0
create-video-connection 14 22 0
create-video-connection 18 33 0
create-video-connection 17 28 0
create-video-connection 44 45 0
create-video-connection 46 42 0
```

create-ftp-connection 19 34 0
create-ftp-connection 21 31 0
create-ftp-connection 23 35 0

```
#
# SWAN NS-2 Simulator
#

# ======================================================================
# Define options
# ======================================================================
set opt(chan)           Channel/WirelessChannel
set opt(prop)           Propagation/TwoRayGround
set opt(netif)          Phy/WirelessPhy
set opt(mac)            Mac/802_11
set opt(ifq)            Queue/DropTail/PriQueue
set opt(ll)             LL
set opt(ant)            Antenna/OmniAntenna

set opt(x)              170             ;# X dimension of the topography
set opt(y)              170             ;# Y dimension of the topography
set opt(cp)             "./traffic.tcl"
set opt(sc)             "./MOBILITY/test.tcl"
#set opt(sc)            ./mobility.tcl
set opt(progress)       4               ;# progress markers

set opt(ifqlen)         50              ;# max packet in ifq
set opt(nn)             50              ;# number of nodes
set opt(seed)           0.0
set opt(stop)           300.0           ;# simulation time
set opt(tr)             "/opt/SANU-16/NOHOP/P0-ftp10.tr"        ;# trace file
set opt(nam)            "/opt/SANU-16/NOHOP/P0-ftp10-demo.nam"
set opt(rp)             AODV            ;# routing protocol script
set opt(lm)             "off"           ;# log movement
set opt(energymodel)    EnergyModel     ;
set opt(initialenergy)  100             ;# Initial energy in Joules

# ======================================================================

set opt(swan_rc)        "ON"            ;# rate controller ON/OFF
set opt(swan_ac)        "ON"            ;# admission controller ON/OFF
set opt(dir)            "/opt/SANU-16/"  ;# result directory
set opt(band)           "100Kb"         ;# initial rate
set opt(ssthresh)       "1Mb"           ;# slow start threshold
set opt(segment)        "50Kb"          ;# increment segment
set opt(mdrate)         "50"            ;# decrement rate
set opt(gap)            "1.2"           ;# gap control
set opt(minband)        "100kb"         ;# minimum rate
set opt(acrate)         "1600Kb"        ;# admission control rate
set opt(thrate)         "4000Kb"        ;# threshold rate

# ======================================================================
#
set AgentTrace                  OFF
set RouterTrace                 OFF
set MacTrace                    OFF

LL set mindelay_                50us
LL set delay_                   25us
LL set bandwidth_               0       ;# not used
LL set off_prune_               0       ;# not used
LL set off_CtrMcast_            0       ;# not used

Agent/Null set sport_           0
Agent/Null set dport_           0

Agent/CBR set sport_            0
Agent/CBR set dport_            0

Agent/TCPSink set sport_        0
Agent/TCPSink set dport_        0

Agent/TCP set sport_            0
Agent/TCP set dport_            0
Agent/TCP set packetSize_       1460

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 11*1e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set bandwidth_ 11e6

# Initialize the 802.11 MAC
Mac set bandwidth_ 11e6

# ======================================================================
```

```
#
# nodes: 50, pause: 0.00, max speed: 10.00  max x = 170.00, max y: 170.00
#
$node_(0) set X_ 38.701396808003
$node_(0) set Y_ 58.239362254689
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 9.724079303585
$node_(1) set Y_ 161.562936556314
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 45.089097239201
$node_(2) set Y_ 45.227765975840
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 164.412549016487
$node_(3) set Y_ 32.584916138477
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 154.915277178487
$node_(4) set Y_ 128.086650153162
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 129.942664450052
$node_(5) set Y_ 19.918163920049
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 59.457416363792
$node_(6) set Y_ 130.977769688092
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 84.219319067807
$node_(7) set Y_ 114.017393045504
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 87.063812212849
$node_(8) set Y_ 130.619575201737
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 43.383299853589
$node_(9) set Y_ 101.859121570992
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 128.813477324602
$node_(10) set Y_ 34.528745310842
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 85.796986451785
$node_(11) set Y_ 32.318190942096
$node_(11) set Z_ 0.000000000000
$node_(12) set X_ 63.917652236966
$node_(12) set Y_ 68.377105708336
$node_(12) set Z_ 0.000000000000
$node_(13) set X_ 140.345568533958
$node_(13) set Y_ 89.206658407344
$node_(13) set Z_ 0.000000000000
$node_(14) set X_ 166.230940422052
$node_(14) set Y_ 147.911686251320
$node_(14) set Z_ 0.000000000000
$node_(15) set X_ 124.233276380813
$node_(15) set Y_ 107.720966729326
$node_(15) set Z_ 0.000000000000
$node_(16) set X_ 52.731259171409
$node_(16) set Y_ 117.177562579264
$node_(16) set Z_ 0.000000000000
$node_(17) set X_ 14.862976051285
$node_(17) set Y_ 59.761804191036
$node_(17) set Z_ 0.000000000000
$node_(18) set X_ 24.256021376991
$node_(18) set Y_ 129.792364695298
$node_(18) set Z_ 0.000000000000
$node_(19) set X_ 84.138355462527
$node_(19) set Y_ 18.961199949013
$node_(19) set Z_ 0.000000000000
$node_(20) set X_ 74.024720133082
$node_(20) set Y_ 133.115672862171
$node_(20) set Z_ 0.000000000000
$node_(21) set X_ 52.795916742541
$node_(21) set Y_ 102.833095528019
$node_(21) set Z_ 0.000000000000
$node_(22) set X_ 108.788770481028
$node_(22) set Y_ 38.445298533566
$node_(22) set Z_ 0.000000000000
$node_(23) set X_ 129.379017669902
$node_(23) set Y_ 96.554847714353
$node_(23) set Z_ 0.000000000000
$node_(24) set X_ 30.323756003599
$node_(24) set Y_ 97.237481513386
$node_(24) set Z_ 0.000000000000
$node_(25) set X_ 90.277040835383
$node_(25) set Y_ 41.715243430056
$node_(25) set Z_ 0.000000000000
$node_(26) set X_ 65.321955063140
$node_(26) set Y_ 110.363201744749
$node_(26) set Z_ 0.000000000000
$node_(27) set X_ 70.574241531883
$node_(27) set Y_ 103.885897683973
$node_(27) set Z_ 0.000000000000
$node_(28) set X_ 141.226337564895
$node_(28) set Y_ 99.754152854174
$node_(28) set Z_ 0.000000000000
$node_(29) set X_ 141.106937713978
$node_(29) set Y_ 115.121566468218
$node_(29) set Z_ 0.000000000000
$node_(30) set X_ 83.350287330537
$node_(30) set Y_ 72.966030707335
$node_(30) set Z_ 0.000000000000
$node_(31) set X_ 169.431915809828
$node_(31) set Y_ 63.462437274909
$node_(31) set Z_ 0.000000000000
$node_(32) set X_ 147.795636079156
$node_(32) set Y_ 133.088105775492
$node_(32) set Z_ 0.000000000000
$node_(33) set X_ 167.190894436954
$node_(33) set Y_ 140.840937289157
$node_(33) set Z_ 0.000000000000
$node_(34) set X_ 117.078698806551
$node_(34) set Y_ 17.668901066094
$node_(34) set Z_ 0.000000000000
$node_(35) set X_ 150.201113268694
$node_(35) set Y_ 158.852549977910
$node_(35) set Z_ 0.000000000000
$node_(36) set X_ 111.845229958139
$node_(36) set Y_ 166.811800200225
$node_(36) set Z_ 0.000000000000
$node_(37) set X_ 4.169896446946
$node_(37) set Y_ 114.826362471535
$node_(37) set Z_ 0.000000000000
$node_(38) set X_ 72.001371515516
$node_(38) set Y_ 6.789941879061
$node_(38) set Z_ 0.000000000000
$node_(39) set X_ 108.265236962859
$node_(39) set Y_ 128.736051171343
$node_(39) set Z_ 0.000000000000
$node_(40) set X_ 104.273474293035
$node_(40) set Y_ 106.028820666018
```

Appendix B

Scripts used for Calculation

```
##############################
 Perl script file for analysis
##############################
#!/usr/bin/perl
@start_tid = () ;
@stop_tid = () ;
@node_delay = () ;
$flow_nr = 0;
$delay = 0;
$start = 0;
$sum = 0;
$LAMO = 0;
$average_delay =0;
print "which delay file ?";
$tracefile = <STDIN>;
chop($tracefile);
open(INFILE,"<$tracefile");
@lines = <INFILE> ;
foreach $line (@lines) {
@fromline = split(s+/, $line) ;
@node_id = split(/_/,$fromline[2]);
$nodeid = $node_id[1];
$flow_id = 1* $fromline[5];
if ($fromline[0] eq 's') {
if (($fromline[3] eq 'AGT') && ($fromline[6] eq 'cbr')) {
$start_tid[$flow_id] = $fromline[1];
}
}
```

```
elsif ( $fromline[0] eq 'r' ) {
if (($fromline[3] eq 'AGT' ) && ($fromline[6] eq 'cbr')) {
$stop_tid[$flow_id] = $fromline[1];
}
}
}
foreach $flow (@start_tid) {
if ($stop_tid[$flow_nr] == ()) {
$stop_tid
                                        $flow_nr] = 0;
$node_dela[$flow_nr] = 0;
}
elsif ($start_tid[$flow_nr] == ()) {
$start_tid[$flow_nr] = 0;
$node_delay[$flow_nr] = 0;
}
else {
$node_delay[$flow_nr] = (( $stop_tid[$flow_nr] - $start_tid[$flow_nr]) *
1000) ;
}
$flow_nr++;
}
foreach $delay (@node_delay) {
$sum = $sum + $delay
}
$average_delay = $sum  $LAMO;
```

```
#!/usr/bin/env python

import sys, re, math, os, shutil
try:
    fil1 = sys.argv[1]
    fil2 = sys.argv[2]
#    fil3 = sys.argv[3]

except:
    print 'Usage:', sys,argv[0], '20-vithr-ftpflow.txt  16-vithr-ftpflow.txt '
    sys.exit(1)
case = "haldi"

f=open(case + '.gnuplot', 'w')
f.write ("""
set title "Aggregate real-time traffic throughput  vs  No. of TCP sources";
set yrange [0:3000] ;
set ylabel 'Aggregate real-time traffic throughput(Kbps)'
set xlabel '# of TCP sources'
set term png small color;
set output 'swan_orig_26_ftpflow_video.png' ;
""")

f.write ("plot  '%s' title 'swan-orig' with lines, '%s' title 'swan-2.6' with lines; \n" %(fil1,fil2))
f.close();
cmd = "gnuplot -geometry 800x600 -persist " +case + ".gnuplot"
failure = os.system(cmd)
if failure:
        print "gnuplot failed"; sys.exit(1)
```

Appendix C

SWAN code

```
#include <stdlib.h>
#include "swan_ac.h"

int hdr_swan::offset_;
static class SWANHeaderClass : public PacketHeaderClass {
public:
  SWANHeaderClass() :
    PacketHeaderClass("PacketHeader/SWAN", sizeof(hdr_swan)) {
      bind_offset(&hdr_swan::offset_);
  }
} class_swan_hdr;

static class SWAN_ACClass : public TclClass {
public:
        SWAN_ACClass() : TclClass("Agent/SWAN_AC") {}
        TclObject* create(int, const char*const*) {
                return (new SWAN_AC);
        }
} class_swan_ac;

SWAN_AC::SWAN_AC() : Agent(PT_SWAN)
{
        bind_bw("acrate_", &acrate_);
        bind_bw("thrate_", &thrate_);
        rtrate_ = 0;
        num_bit = 0;
        ce_src = 0;
        ce_dst = 0;
        ce_hop = 0;
        ce_by_pending = 0;
        fp = 0;
        swan_status = SWAN_NEW;
        target_ = 0;
}

int SWAN_AC::command(int argc, const char*const* argv)
{
        if (argc == 3) {
                if (strcmp(argv[1], "target") == 0) {
                        target_ = (NsObject*) TclObject::lookup(argv[2]);
                        assert(target_);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "down-target") == 0) {
                        downtarget_ = (NsObject*) TclObject::lookup(argv[2]);
                        assert(downtarget_);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "monitor") == 0) {
                        monitor(atoi(argv[2]));
                        return TCL_OK;
                }
                if (strcasecmp (argv[1], "ip-addr") == 0)
                {
                        net_id = atoi(argv[2]);
                        return TCL_OK;
                }
        }
        else if (argc == 4) {
               if (strcmp(argv[1], "util-trace") == 0) {
                 char fname[64];
                 sprintf(fname, "%s/util%s", argv[3], argv[2]);
                 if ((fp = fopen(fname, "w+")) == 0)
                 {
                     printf("Error opening %s\n", fname);
                     return TCL_ERROR;
                 }
                 return TCL_OK;
               }
        }
        return TCL_OK;
}

void SWAN_AC::recv(Packet* p, Handler* h)
{
  struct hdr_cmn *cmn = HDR_CMN(p);
  struct hdr_ip  *ip = HDR_IP(p);

  float ctime = Scheduler::instance().clock();

  if (cmn->ptype_ == PT_SWAN)
    {
      process_swan_message(p, h);
      return;
    }

  if (cmn->ptype_ == PT_CBR)
    {
      if (ip->src_.addr_ == net_id)
        {
          ip->ecn_ = 1;
          ip->ce_ = 0;
          switch(swan_status)
            {
            case SWAN_NEW:
              send_probe_request(p);
              ip->tos_ = 0;
              swan_status = SWAN_PROBING;
              break;
            case SWAN_PROBING:
              ip->tos_ = 0;
              if (probe_timer < ctime)
                {
                  swan_status = SWAN_ADMITTED;
                  send_probe_request(p);
                }
              break;
            case SWAN_REPROBING:
              ip->tos_ = 1;
              if (probe_timer < ctime) send_probe_request(p);
              break;
            case SWAN_PENDING:   ip->tos_ = 2;
              if (pending_timer < ctime)
                {
                  swan_status = SWAN_ADMITTED;
                  ip->tos_ = 1;
                }
              break;
            case SWAN_ADMITTED:  ip->tos_ = 1; break;
            case SWAN_REJECTED:  ip->tos_ = 0; break;
            }
        }
      else if (ce_hop && ip->tos_ && ip->ecn_)
        {
          mark_ecn(p);
```

```
/* -*-  Mode:C++; c-basic-offset:8; tab-width:8; indent-tabs-mode:t -*- */
/*
 * Copyright (c) 1996-1997 The Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the Network Research
 *      Group at Lawrence Berkeley National Laboratory.
 * 4. Neither the name of the University nor of the Laboratory may be used
 *    to endorse or promote products derived from this software without
 *    specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include "swan_rc.h"

static class SWAN_RCClass : public TclClass {
public:
        SWAN_RCClass() : TclClass("SWAN_RC") {}
        TclObject* create(int /* argc */, const char*const* /* argv */) {
                return (new SWAN_RC);
        }
} class_swan_rc;

SWAN_RC::SWAN_RC() : dynamic_(0), itq_(0)
{
        bind_bw("bandwidth_", &bandwidth_);
        bind_time("delay_", &delay_);

        bind_bw("ssthresh_", &ssthresh_);
        bind_bw("segment_", &segment_);
        bind_bw("mdrate_", &mdrate_);
        bind_bw("gap_", &gap_);
        bind_bw("minband_", &minband_);

        on_ = 0;
        fp1 = 0;
        fp2 = 0;
        num_pkt = 0;
        num_delayed_pkt = 0;
        num_byte = 0;
        sum_delay = 0;
        avg_delay = 0;
}

int SWAN_RC::command(int argc, const char*const* argv)
{
        if (argc == 2) {
                if (strcmp(argv[1], "isDynamic") == 0) {
                        dynamic_ = 1;
                        itq_ = new PacketQueue();
                        return TCL_OK;
                }
        }

        if (argc == 3) {
                if (strcmp(argv[1], "adjust") == 0) {
                        adjust(atoi(argv[2]));
                        return TCL_OK;
                }
        }

        if (argc == 4) {
                if (strcmp(argv[1], "delay-trace") == 0) {
                        char fname[64];
                        net_id = atoi(argv[2]);
                        sprintf(fname, "%s/delay%d", argv[3], net_id);
                        if ((fp1 = fopen(fname, "w+")) == 0) {
                                printf("Error opening %s\n", fname);
                                return TCL_ERROR;
                        }
                        return TCL_OK;
                }
                if (strcmp(argv[1], "rate-trace") == 0) {
                        char fname[64];
                        net_id = atoi(argv[2]);
                        sprintf(fname, "%s/rate%d", argv[3], net_id);
                        if ((fp2 = fopen(fname, "w+")) == 0) {
                                printf("Error opening %s\n", fname);
                                return TCL_ERROR;
                        }
                        return TCL_OK;
                }
        }

        return Connector::command(argc, argv);
}

void SWAN_RC::recv(Packet* p, Handler* h)
{
        double txt = txtime(p);
        Scheduler& s = Scheduler::instance();
        if (dynamic_) {
                Event* e = (Event*)p;
                e->time_= txt + delay_;
                itq_->enque(p); // for convinience, use a queue to store packets
 in transit
                s.schedule(this, p, txt + delay_);
        } else {
                s.schedule(target_, p, txt + delay_);
        }
```

Bibliography

[1] The network simulator ns-2 [http://www.isi.edu.nsnam/ns].

[2] *On Demand Distance Vector (AODV) Rounting*, november 1998. Internet-Draft.

[3] *A Two-bit Differentiated Services Architecture for the Internet*, juli 1999. RFC-2638.

[4] Gahng-Seop Ahn, Andras Veras Andrew T.Campbell, and Li-Hsiang Sun. "supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks". *IEEE Transaction on mobile computing*, september 2002.

[5] Harpreet Arora, Lloyd Greenwald, Usha Rao, and John Novatnack. *Performance Comparison and Analysis of two QoS schemes: SWAN and Diffserv.* Department of Computer Science, Drexel University, Philadelphia, PA 19104.

[6] Imrich Chlamtac, Marco Conti, and Jennifer J-N Liu. *Mobile ad hoc networking : imperatives and challenges.* Elseview B.V., 2003. Ad Hoc Networks I.

[7] Zeinalipour-Yazti Demetrios. A glance at quality of services in mobile ad-hoc networks. Seminar on mobile ad hoc, University of California, 3201 Canyon Crest Dr, Riverside CA, 2001.

[8] Yixin Dong and Dimitrios Makrakis. *Effective Bandwidth Calculation for QoS Routing in IEEE 802.11 DCF Ad hoc Networks.* University of Ottawa, Broadband Wireless and Internetworking Research Laboratory.

[9] Michael Gerharz, Christian Vogt, and Christian Waal de. Current approaches towards improved quality-of-service provision in mobile ad-hoc networks. Technical report, Rheinische Friedrich Wilhelm University, Bonn, Germany, march 2003.

[10] Gang Hua and Feifei Lou. Performance evaluation of quality of service schemes in ad-hoc networks.

[11] Andreas Kassler, Davide Mandator Teodora Guenkova-Luy, and Peter Schoo. Enabling mobile heterogenerous networking environments with end-to-end user perceived qos. Technical report, 2000.

[12] Nahrstedt Klara, Samarth H. Shah, and Kai Chen. Cross-layer architectures for bandwidth management in wireless networks. In *RESOURCE MANAGEMENT IN WIRELESS NETWORKING*. Kluwer Academic Publishers, 2004.

[13] J. Manner, A. Laukkanen, M. Kojo, and K Raatikainen. *A QUALITY OF SERVICE ARCHITECTURE FRAMEWORK FOR MOBILE NETWORKS*. Department of Computer Scienc, FIN-00014 University of Helsinki, Finland.

[14] Jukka Manner, Alberto Lopez, Andrej Mihailovic, Hector Velayos, Eleanor Hepworth, and Youssef Khouaja. A study on qos provision for ip-based radio access networks. Technical report, IST project IST-1999-10050 BRAIN, 1999.

[15] Prasant Mohapatra, Jian Li, and Chao Gui. Qos in mobile ad hoc networks. Technical report, University of California, Department of Computer Science, Davis, CA, December 2002.

[16] Yasser L. Morgan and Thomas Kunz. Enhancing swan qos model by adopting destination-based regulation. Technical report, Carleton University, 2004.

[17] Navid Nikaein and Christian Bonnet. A glance at quality of service models for mobile ad hoc networkss. Technical report, Institut Eure'com, Sophia Antipolis-France, 2002.

[18] North Carolina State University. *Theoretical Maximum Throuhput of IEEE 802.11 and its Applications*, april 2003. The 2nd IEEE International symposium on Network Computing and Application.

[19] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[20] Dmitri D. Perkins and Herman Hughes. *A survey on quality-of-service support for mobile ad hoc networks*, chapter 2, pages 503 – 513. John Wiley and Sons, Ltd, 2002.

[21] Sivakumar Raghupathy, Prasun Sinha, and Vadurvur Bharghavan. *CEDAR: A Core-Extraction Distributed Ad hoc Routing Algorithm*, chapter volum 17, pages 1454 – 1464. In IEEE on Selected Areas in Communications, Special Issues on Ad Hoc Mobile Networks, 1999.

[22] T. Bheemarjuna Reddy and B.S. Manoj I. Karthigeyan. Quality of service provisioning in ad hoc wireless networks: a survey o f issue and solutions. *Ad Hoc networks 4*, 2006.

[23] Pedro M. Ruiz and Emilio Garcia. *IMPROVING USER-PERCEIVED QOS IN MOBILE AND WIRELESS IP NETWORKS USING REAL-TIME ADAPTIVE MULTIMEDIA APPLICATIONS.* Agora Systems, S.A., Aravaca, 28040 Madrid, Spain.

[24] Kui Wu and Janelle Harms. Qos support in mobile ad hoc networks.

[25] Yaling Yang and Robin Kravets. *Distributed QoS Guarantees for Realtime Traffic in Ad Hoc Networks.* Department of Computer Science, University of Illinois at Urbana-Champaign.