ENABLING THEORETICAL MODEL BASED TECHNIQUES FOR SIMULATING LARGE SCALE NETWORKS

BY

HWANGNAM KIM

B.E., Pusan National University, 1992
M.S., Seoul National University, 1994

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

Abstract

Modern data communication networks are extremely complex and do not lend well to theoretical analysis. It is not unusual that network analysis can be rigorously made after leaving out several subtle details that cannot be easily captured in the analysis. As a result, packet mode, event driven simulation studies are usually resorted to better study the performance of network components, protocols, and their interaction. The major obstacle in packet mode simulation is, however, the vast number of packets that have to be simulated in order to produce accurate results, especially in large scale networks. What seems to be a reasonable solution is really to incorporate theoretical modeling into packet mode simulation.

The notion of *fluid model based simulation* is recently proposed to alleviate the computational overhead in packet mode simulation. Conceptually, a fluid model is developed and incorporated into the simulation engine. In the course of simulation, a sequence of closely-spaced packets are abstracted into a fluid information, and the fluid model is used to determine its behavior. As the first theme, we investigate whether or not the fluid model based simulation is effective in simulating IEEE 802.11-operated wireless LANs, and to develop a fast simulation framework to expedite simulation, while not compromising the fidelity of simulation results.

In spite of its effectiveness in terms of reducing the execution time, fluid model based simulation is not well-suited for studying the network behavior under light and/or sporadic traffic, as it is built upon the assumption of a large number of active flows in the network. To address the issue, we contrive *network calculus based simulation* as another main theme in the thesis. We firstly characterize how TCP congestion control interacts with AQM strategies with network calculus theory, and then determine a set of scheduling rules to regulate TCP traffic, and finally incorporate the rules into a network simulation engine to improve simulation performance.

Although both fluid model based simulation and network calculus based simulation indeed give encouraging results, they cannot provide the packet level dynamics, such as the instantaneous queue length and packet dropping probability, due to the use of abstract simulation units, i.e., fluid rate and traffic amount. In order to address this issue, we propose hybrid simulation techniques, called *mixed mode simulation*, to discover packet level details of one packet mode, foreground flow, approximating all the other flows with theoretical model based, background flows. In the mixed mode framework, packet mode simulation co-exists with theoretical model based simulation within one simulation framework, and therefore analytical models of specifying the interactions should be devised.

Lastly, we also contrive a new *rescaling simulation methodology (RSM)* to simulate large scale IP networks with TCP and/or UDP traffic. Even though mixed mode simulation can produce packet level details, there exist the cases that all the network behaviors, inclusive of all the flows and all the networking points, should be inspected. The underlying idea of RSM based simulation is to reduce the computational cost by scaling down the network to tractable one that can be simulated for a short time interval to produce sufficient results at packet mode, and then to extrapolate the results expected from the original network with the obtained those. In order to give a guideline for both down- and up-scaling or to explain how to preserve the network properties unique in the original network within the down-scaled network, a rescaling model is contrived and presented.

To my mother

Acknowledgments

This thesis has been written with the help of my adviser Professor Jennifer C. Hou. Professor Hou has guided, stimulated, encouraged, and supported me to finish my Ph.D. study with all the aspects. I would like to appreciate Professor Bruce Hajek, Professor Lui Sha, and Professor Klara Nahrsted for their invaluable advices and devoted service on my thesis committee. I also would like to give a sincere gratitude to my fellow graduate students, Wei-Peng Chen, Guanghui He, Chunyu Hu, Ning Li, Lu-chuan Kung, Ahmed Sobeih, Honghai Zhang, Rong Zheng and Dennis Y. Chi for research discussion, interest, and comment. Especially I am obliged to Sung-Il Pae, Kihwal Lee, Dr. Hyuk Lim, Youngihn Kho, and Dr. Jong-Kwon Lee, for technical discussion and argument, personal talk and comment, friendship and a lot of coffee. Last but certainly not least, I would love to pay uncountably many thanks and respects to my parents and brothers for their constant support, love, and confidence for me over the years from the bottom of my heart.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern data communication networks are extremely complex and do not lend well to theoretical analysis. With computer/network entities and techniques interacting and interfering with one another, optimization problems do not have a simple and regular structure that allows us to neatly fit it into the framework of established optimization theories. As a result, it may be more feasible to carry out simulation to study and evaluate the performance of network entities and protocols, and interaction among them. The major obstacle in packet mode network simulation is, however, the vast number of packets that have to be simulated in order to produce accurate results, especially in large scale networks. Each packet will generate a number of events (e.g., arrival of a packet at the router, its departure, and its queuing, just to name a few) on the path from the source to the destination and each event has to be executed at some specified time point. As the CPU time required is roughly proportional to the number of events that have to be executed, packet mode simulation easily becomes computationally expensive, if not infeasible, when the network size and/or the traffic amount is extremely large.

What seems to be reasonable is to combine theoretical modeling with packet mode simulation so as to take advantage of both simulation, while not suffering significantly from their individual drawbacks. The main intent in this thesis is thus to investigate the feasibility of leveraging theoretical models in simulating large scale networks. As the specific domain to which theoretical models are applied, we choose IEEE 802.11-compliant wireless LANs (WLANs) and TCP and/or UDP traffic over IP-operated, packet-switched networks. We also investigate the performance achieved as well as the error incurred, for each developed theoretical model based technique in order to evaluate it. As for the error evaluation, we compare the results obtained from each technique to those obtained from its corresponding packet mode simulation. As for the performance evaluation, we adopt the execution time since (i) it encompasses the loading time of simulation objects as well as the execution time necessary for simulating given networks for a given simulation time, and (ii) it alleviates the additional overhead necessary to define other measures and to implement the measuring scheme within simulation engine. Specifically, we carried out research along the following research thrusts.

Fluid model based simulation for IEEE 802.11-compliant WLANs: As the first empirical trial, we investigate how applicable and feasible fluid model based simulation is for WLANs. In fluid model based simulation, a cluster of closely-spaced packets is modeled as a fluid chunk at a specific time point or a flow rate during a time interval, and mathematical models (fluid models) are used to specify the behaviors for the abstract units [22, 28, 31, 35, 45, 47]. Network simulator equipped with fluid models then keeps track of fluid chunks and their rate changes at each network component on the communication path from the source to the destination. As a large number of packets are abstracted as a single fluid chunk, the computational overhead is expected to be lessoned. The fluid model based simulation currently has been used to study the throughput behavior of TCP and congestion control algorithms [21, 32, 37, 39, 42] in the steady state, but it is not clear whether or not the same technique can be applied to analyze system behaviors in IEEE 802.11-compliant WLANs.

To conduct this experiment, we first derive the analytical model that characterizes data transmission activities in IEEE 802.11-compliant WLANs with/without the RTS/CTS mechanism. All the control overhead incurred in the physical and MAC layers, as well as system parameters specified in the IEEE 802.11 standard [24] are faithfully figured in. In the model, all the analytical components are described with the attempt rate that is the rate to place the frames on the wireless medium, and the system throughput is also defined as a function of the rate. The attempt rate is a function of the number of active nodes and their respective backoff window sizes. We validate the model with simulation in the cases in which the network is and is not saturated. We then implement, with the use of the time stepping technique [45], fluid model based simulation for WLANs in *ns-2* [4], and conduct a simulation study to evaluate the framework in terms of speed-up and errors incurred under a variety of network configurations and traffic types.

Network calculus based simulation for TCP networks: One drawback of utilizing fluid models in network simulation is that fluid models may not be well-suited to model light, sporadic traffic [37, 42]. To address this issue, we contrive *network calculus based simulation*, in which analytical models built upon *network calculus* are used as an alternative to fluid models. Specifically, we examine the feasibility of incorporating network calculus based models in simulating TCP/IP networks. Network calculus is grounded on the mathematical theory of *Min-Plus* (or *Max-Plus* algebra) [1, 9, 13]. By exploiting properties in the network calculus, we characterize how TCP congestion control, additive increase and multiplicative decrease (AIMD) and *slow start*, regulates TCP flows within the simulation engine, without the assumption of existence of a large number of flows which is usually made in fluid model based simulation.

Following the same line of approaches as in fluid model based simulation, we first divide the time axis into intervals, each of which consists of multiple round-trip times, and derive a TCP throughput model which derives the attainable throughput for a TCP flow, given the number of losses in an interval. Then based on the derived throughput model, we define a set of network calculus based theorems that give upper and lower bounds on the attainable TCP throughput in each interval. Finally, we implement network calculus based simulation in *ns-2*, conduct a simulation study to evaluate the network calculus based simulation, and present the performance gain in terms of the execution time thus reduced and the error discrepancy in terms of the discrepancy between the simulation results and corresponding ones in packet mode simulation.

Mixed mode simulation for IEEE 802.11-compliant WLANs and TCP networks: Although both fluid model based and network calculus based simulation indeed give encouraging results, they cannot provide packet level dynamics, such as the instantaneous queue length and the packet dropping probability. In some sense, such the theoretical model based simulation trades some degree of accuracy and packet level dynamics for simulation performance. If packet level details are of concern to users, the best approach seems to simulate foreground traffic (whose packet dynamics are of interest) in the packet mode, and model the background traffic (that comprise of possibly hundreds of flows) with a model based simulation.

For the purpose, we propose the notion of *mixed mode simulation* to combine the performance gain of theoretical model based simulation with the accuracy afforded by packet mode simulation. The important problem in mixed mode simulation is to specify how one foreground flow in packet mode simulation interacts with other flows in theoretical model based simulation at the point of interaction, such as the wireless medium in a WLAN or the routing buffer in a bottleneck link. We develop two models of interaction: the one is for IEEE 802.11-operated WLAN and the other for TCP-operated networks. According to each developed model of interaction, we implement two mixed mode simulations in *ns-2*. We then conduct comprehensive simulation studies to evaluate each mixed mode simulation with respect to accuracy, which is error discrepancy in simulation results from both packet mode and theoretical model based simulation, and efficiency, which is the performance scale-up in conducting simulation.

Rescaling simulation model based simulation: Even though the mixed mode simulation can produce packet level details for one foreground flow, we cannot approximate some or all flows into background flows with theoretical models when all the details for all the flows are of concern. For example, we might have to investigate all the effects of the flows passing through one point of interest. In order to deal with such the cases, we take a dramatic departure from the above approaches and contrive a new rescaling simulation methodology (RSM) for simulating large scale IP networks, inclusive of UDP and TCP traffics.

In packet mode simulation, the computational cost increases accordingly due to a vast number of packet events as the number of events increases with the network size and the amount of traffic. Therefore, if we can scale down the network to one that can be simulated at the packet mode in a short time interval to produce sufficient results, and then extrapolate, without loss of accuracy, results for the original network, we can significantly reduce the computational cost and yet preserve the network dynamics. The key issue is how to preserve the properties that characterize the original network in the downscaled network so that accurate results can be easily extrapolated after the packet mode simulation in the smaller networks. In particular, the network property of interest should remain invariant in the process of scaling down and up the network. As for the invariant, we use the *bandwidth-delay* product as the network property to be preserved, since it represents the capacity of the "pipe," i.e., the amount of data packets that can be transmitted without waiting for the acknowledgment. By preserving this product invariant during the down/up scaling operations, the network capacity as perceived by each TCP connection is preserved, and we can formally prove that the queue dynamics, the RTT dynamics, the TCP window size dynamics, and even the effect of TCP-friendly flows remain unchanged in the operations.

Contributions: The major contribution of this research is a rigorous, in-depth investigation of whether or not, and to what extent, fluid models, network calculus models, mixed mode models, and rescaling models may be leveraged to facilitate large scale network simulation. In summary, we are the first to carry out and validate the use of fluid model based simulation for simulating IEEE 802.11-operated WLANs. We are also the first to apply network calculus theory to large scale network simulations, in which we present that the network calculus theory can be employed to improve network simulation in addition to its existing application area in the traffic regulation. We then contrive how to integrate packet mode and theoretical model based simulation within one simulation framework. The proposed mixed mode simulations provide packet level details for one foreground flow of concern and interest while it keeps the performance improvement by approximating other flows with theoretical models. Finally, we contrive rescaling simulation methodology to keep the packet level details for all the flows in large scale IP networks and at the same time to improve simulation performance by simulating down-scaled networks and estimating the results for the original networks.

Road-map: The thesis is organized as follows. In Chapter 2, we give a succinct summary of related work in three perspectives: the one pertains to existing techniques developed to speed up network simulations, such as fluid model based simulation and simulation technique based on down-scaling; another one introduces existing analytical models for both IEEE 802.11 DCF and TCP congestion control; and the other looks over the development in network calculus system theory (that Chapter 4 exploits). In Chapter 3, we derive the analytical model that characterizes the data transmission activities in IEEE 802.11-operated WLANs, discuss how we implement fluid model based simulation, and evaluate the simulation in comparison with packet mode simulation. In Chapter 4, we establish a set of regulation rules, i.e. a set of theorems, to schedule TCP traffic in a simulation engine, based on network calculus theory. We then explain the structure of network calculus based simulation and its performance gain and error discrepancy. Following that, we derive two interaction models between one foreground, packet mode flow and the other background, theoretical model based flows at the point of wireless medium and congested link. With the models, in Chapter 5, we implement mixed mode simulation for IEEE 802.11-compliant WLANs and evaluate it, compared to both packet mode simulation and fluid model based simulation. We also develop mixed mode simulation for TCP-operated networks in Chapter 6 in the same procedure of Chapter 5. In Chapter 7, we propose the rescaling simulation methodology for large scale IP networks with TCP and/or UDP traffic. With the methodology, we prove that the property of the original network is preserved in down-scaled network, and we therefore exploit the latter network to estimate the results expected from the former network. Finally, we make conclusions in Chapter 8.

Chapter 2

Related Work

In this Chapter, we summarize existing research efforts that pertain to the study, and categorize them into five thrusts:

Fluid model based simulation: Several research efforts have focused on fluid model based simulation. Liu *et al.* [31] demonstrated the fundamental performance gain in fluid model based simulation over, rather than a realistic network with detailed network protocols, simple network components. Milidrag *et al.* [35] presented various sets of differential equations that describe the behaviors of network components in the continuous time domain. They showed that as long as the behavioral characteristics in the continuous time domain can be exactly specified, fluid simulation gives results with reasonable error bounds. Wu *et al.* [45] studied the error behavior that simulation results exhibit in a simple $M/D/1$ network configuration.

Fluid models have also been used to study the throughput behavior of TCP and congestion control algorithms, together with active queue management in the steady state [37, 39, 42]. Liu *et.al.* [32] and Gu *et al.* [21] solve fluid models with the numerical Runge-Kutta method, and incorporate numeric results in the simulation of large scale IP networks. As indicated in Chapter 1, fluid model based simulation may not render satisfactory performance (in terms of the discrepancy between results obtained in packet level simulation and fluid model based simulation) in the case of light and/or sporadic traffic, as it relies on the assumption of existence of a large number of flows [32, 37, 42]. It is also limited to show kinetic transient behaviors of the network.

Simulation based on scaling down the network: Pan *et al.* [40] introduce two SHRiNK methods to sample, simulate, and predict the network behavior. The first method deals with IP networks with long-lived flows and the other with a mixture of long-lived and short-lived flows. A proportion of the traffic flows are independently sampled and collected. The original network is reduced to a downscaled one and fed with the sampled traffic. In the first method, the downscaling operation is performed by reducing the number of flows, the link capacity, the maximum buffer sizes, and AQM parameters, while *keeping the end-to-end delay invariant.* In order to keep the end-to-end delay invariant, the queue dynamics has to be approximated (by certain linear functions) and may respond differently to each TCP connection. As a result, the network capacity as perceived by each TCP connection during its interaction with the network is changed. Moreover, in cases where the queue dynamics cannot be adequately approximated with linear functions, e.g., when Drop Tail is used as the AQM strategy, SHRiNK cannot operate correctly. Although similar to RSM, the second SHRiNK method was presented without theoretical reasoning. In addition, both SHRiNK methods rely heavily on the assumption that each input flow is a Poisson process, and moreover, they assume that all the traffic are TCP flows. Compared with SHRiNK, RSM can be universally applied to long-lived, shorted-lived, and the mixture of long-lived and short-lived TCP connection; deal with the interference of TCP friendly and unresponsive UDP flows; the down/up scaling operations in RSM do not change the queue dynamics (which will be rigorously proved); and its correctness does not rely on any assumption about input traffic. The detailed comparison is referred to Chapter 7.

Analytical models for the IEEE 802.11 MAC DCF protocol: Bianchi [6] models the behavior of the binary backoff counter at one tagged station as a discrete Markov chain model. It determines the transmission probability (τ) and analyzes the saturation throughput under the assumption that in each transmission attempt, regardless of the number of retransmissions, each packet collides with constant and independent probability p. It is intuitive that this assumption is more accurate when W and n get larger. Although the model does not consider the case in which the backoff counter stays at the current value when the medium is sensed busy due to the data transmission activities (initiated by other stations), it motivates a significant amount of subsequent analysis work.

Calí *et al.* [11] derive a theoretical throughput bound by approximating IEEE 802.11 with a p-persistent model of IEEE 802.11. Based on the analytical model, they observe that the system throughput only relies on the value of p and the number of active stations, N. They also show that with the current parameter settings of IEEE 802.11, it can hardly achieve the theoretical capacity bound. As such, they suggest to incorporate a parameter tuning method in IEEE 802.11 so as to achieve the capacity bound. They only deal with IEEE 802.11 DCF without the RTS/CTS mechanism under the assumptions that the length of data frame follows a geometric distribution, and that all the stations always have packets ready for transmission (i.e., under the *asymptotic* condition).

Carvalho *et al.* [12] propose an analytical model which computes the average service time and jitter per frame in a saturated IEEE 802.11 ad-hoc network. They show that the existing binary backoff scheme is not appropriate for supporting delay constraints, and that use of a large and constant contention window size is more efficient than binary backing off the window size. This suggests that the initial contention window size CW_{min} should be set to a large enough value to avoid excessive backoff.

Foh and Zukerman [18] analyze, by leveraging the throughput analysis by Bianchi [6], the saturation throughput with a Markov chain with a single server. They assume that the number of active stations increases according to a Poisson process and decreases according to the state dependent service process. Wu *et al.* [44] also exploit the analysis by Bianchi to modify IEEE 802.11 DCF for reliable transport protocol over IEEE 802.11 WLANs. They extend the Markov chain model in [6] to incorporate the frame retransmission limit, and hence the revised model achieves better accuracy in characterizing the transmission activities of IEEE 802.11 DCF.

Xiao [46] extends Bianchi's Markov chain model [6] to accommodate the case of multi traffic classes, and incorporate three tunable parameters into the model: the initial contention window size, the retry limit, and the backoff window-increasing factor. However, the author does not figure in the effect of AIFS values. With the use of the model, the performance of IEEE 802.11e in terms of saturation throughput, saturation delay and frame dropping probability is analytically derived.

Ge and Hou [19], on the other hand, extend the work by Calí *et al.*, devise an analytical model for a multi-class, *p-persistent* version of IEEE 802.11 DCF, and derive the optimal values of the probability, p_i, that a class-i frame attempts for transmission in a slot under the asymptotic condition. As the analysis in [7] indicates, there is an one-on-one correspondence between the value of p in the p-persistent version of IEEE 802.11 DCF and the contention window size, CW, in the legacy IEEE 802.11 DCF. Hence the results derived in [19] can be readily applied to tune the contention window size in the legacy IEEE 802.11 DCF, so as to optimize the protocol capacity in the case of multiple traffic classes.

Analytical models for TCP congestion control: Kelly [25, 27, 26] analyzes the relationship between network rate control and user payment, and proposes an analytic model to optimize both the user utility and the network price in a distributed manner. The model assumes that network traffic can adjust its rate and provide constraints and prices for users, while each user attempts to maximize his utility function at his current rate. After the author formulate the model, he divides the principal optimization problem into two subproblems of optimization: network side and user side optimization, and exploits the standard optimization techniques to solve them. This work is referred as a milestone in many fluid model based analysis for TCP congestion control interacting with networks.

Based on duality theory, Low [33, 34] also proposes an optimization network model, in which the source rate can be adjusted in response to congestion notification and the network determines the level of congestion with respect to source rates. The overall analytic model is to determine source rates to maximize the aggregate benefit with the current set of source rates, subject to the network capacity constraints or the congestion indication.

Misra *et al.*[37, 36] analyze the TCP throughput behavior based on TCP feedback, such as retransmission timeout and triple duplicated ACKs, and a stochastic loss process model [37]. The authors approximate the TCP window size and the loss behavior with a set of stochastic differential equations, and then show how to employ the derived equations to predict the TCP throughput in steady state. Their subsequent work [36] extends the previous work to include the interaction between TCP flows and network routers equipped with active queue management (AQM) strategy. In addition to the previous set of stochastic differential equations, the authors derive several differential equations to describe AQM policy. With all the equations, the authors estimate the network behavior as well as the TCP behavior within a unified framework.

Shakkotai and Srikant [42] focus on justification of the use of deterministic differential equations in order to model TCP congestion control. They prove that the delay differential equations for TCP congestion control can produce an appropriate throughput limit of real TCP traffic, as far as a large number of flows exist in the network. They also show that the difference between simulation results and analytical results comes from the packet discretization effect, i.e., the real congestion control system (or the corresponding simulation) does not operate continuously but handle packets in discrete time.

Network calculus: Based on that data streams that enter the network and satisfy "burstiness constraints", Cruz analyzes network delays [15, 16]. Such an analysis lays its base on *Min-Plus* algebra [1, 9, 13] and has been recognized as constrained traffic regulation problems. The fundamental components developed in the system theory are *shaper* (or *regulator*, or *buffered leaky bucket controller*) for traffic regulation and *service curve* (or *f-server*) for provisioning of service guarantees. The results derived in the theory enable one to understand fundamental properties of, and hence better deploy, flow control, multimedia smoothing, delay control, and integrated service provisioning. In addition, by extending the time invariant theory to the time varying case, dynamic traffic regulation or service guarantee can be considered in the same framework [9, 14]. Finally, by introducing packetizers, the effect of variable length packets can be analyzed using the theory developed under network calculus [8, 13]. The interested readers are referred to [1, 9, 13] for a detailed account of network calculus.

Although network calculus has been mainly used in the context of QoS provisioning, it has been recently applied to other domains. In particular, Baccelli and Hong [2] show that key feature of TCP operated networks can be expressed via a linear dynamical system in the max-plus algebra (max-plus algebra is the dual form of min-plus algebra).

Chapter 3

Fluid Model Based Simulation for IEEE 802.11-operated WLANs

In this chapter, we derive a mathematical model that fully characterizes the data transmission activities in IEEE 802.11-operated WLANs. Then, we incorporate the model in network simulation, and carry out the so-called *fluid model based* simulation[22, 31, 35, 45, 47]. We investigate to what extent (in terms of practicality and feasibility), fluid model based simulation can be used to simulate IEEE 802.11-operated WLANs.

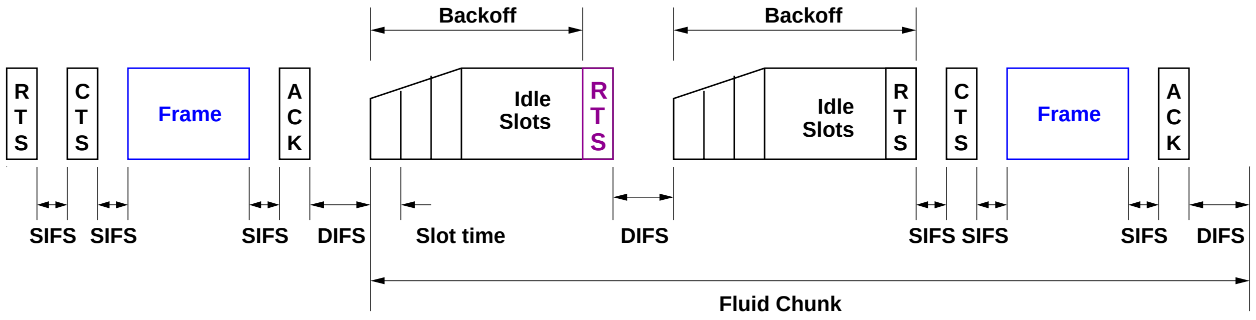
3.1 Operations of IEEE 802.11

As the standard medium access control (MAC) and physical layer, IEEE 802.11 [24] provides two access methods: (i) the *Distributed Coordination Function* (DCF), also known as the basic access method, is a carrier sense multiple access with collision avoidance (CSMA/CA) protocol; and (ii) the Point Coordination Function (PCF) is an access method similar to a polling system and uses a point coordinator to arbitrate the access right among stations. In addition, the standard includes a floor acquisition mechanism, called request to send/clear to send (RTS/CTS), to partially resolve the hidden terminal problem or to decrease the overhead caused by collisions. As DCF is the basic access method in both wireless infrastructure and infrastructure-less environments and is directly related to the work, we look over DCF in the IEEE 802.11 protocol, neglecting PCF.

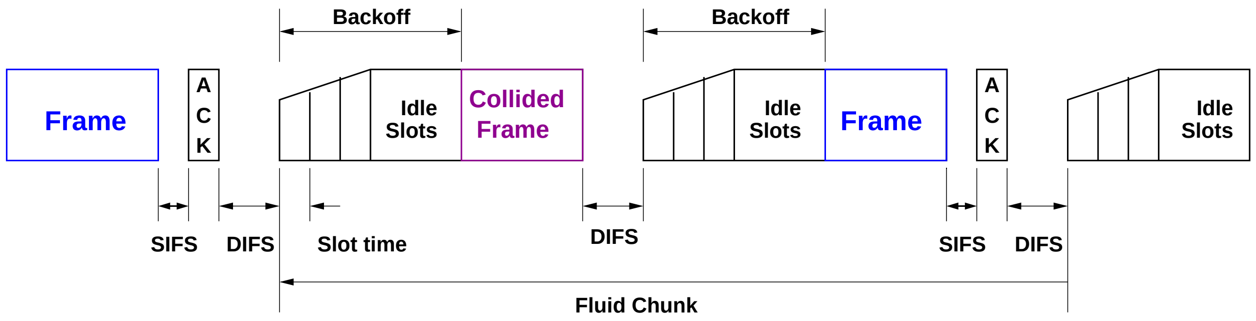
Distributed Coordination Function: DCF operates as follows (Figure. 3.1). Before the data transmission takes place, a station senses the channel to determine whether or not another station is transmitting. If the medium is sensed idle for a specified time interval, called the *distributed* *inter-frame space* (DIFS), the station is allowed to transmit. If the medium is sensed busy, the transmission is deferred until the ongoing transmission terminates. A slotted binary exponential backoff technique is used to arbitrate the access: a random backoff interval value is uniformly chosen in $[0, \widehat{CW} - 1]$ and used to initialize the backoff timer, where $\widehat{CW}$ is the current maximum contention window. The backoff timer is decreased as long as the channel is sensed idle, stopped when data transmission (initiated by other stations) is in progress, and reactivated when the channel is sensed idle again for more than DIFS. The time immediately following an idle DIFS is slotted, with each slot equal to the time needed for any station to detect the transmission of a packet (in the IEEE 802.11 term, MAC Service Data Unit (MSDU)) from any other station. When the backoff timer expires, the station attempts for frame transmission at the beginning of the next slot time. Finally, if the data frame is successfully received, the receiver transmits an acknowledgment frame after a specified interval, called the *short inter-frame space* (SIFS), that is less than DIFS.[1] If an acknowledgment is not received, the data frame is presumed to be lost, and a retransmission is scheduled. The value of $\widehat{CW}$ is set to $CW_{min}(= 32)$ in the first transmission attempt, and is doubled at each retransmission up to a pre-determined value $CW_{max}(= 1024)$. According to the standard, a maximum number of retransmissions (=7 as default) are allowed before the frame is dropped.

In the case that the floor acquisition RTS/CTS mechanism is used, the same procedure is conducted except that RTS/CTS handshake operations proceeds the data/acknowledgment exchange (Figure 3.1(a)). A station sends a RTS frame to the destination before transmitting any MSDU. The destination then responds with a CTS once it has correctly received a RTS. The source can then send the MSDU after receiving the corresponding CTS response. Both the RTS and CTS frames contain the information on the duration of the MSDU/ACK transmission. Based on this information all the surrounding stations can update their internal timers called Network Allocation Vector (NAV) and defer any transmission until this timer expires. Even if a hidden station cannot hear the RTS from the source station, it will be able to receive the CTS response from the destination station and update its NAV accordingly. This mechanism guards transmission between stations against unexpected transmission from hidden stations.

[1] The necessity of returning an acknowledgment is due to the inability of WLANs to listen while transmitting, since usually only one antenna is available for both transmitting and receiving.



(a) DCF in conjunction with the RTS/CTS mechanism



(b) DCF

Figure 3.1: The timing structure of a MAC fluid in IEEE 802.11.

3.2 Derivation of Analytical Model

3.2.1 An Overview of Analysis

We have two important analytical components: *fluid chunk* and *MAC fluid*. We define a *fluid chunk* as a sequence of collision periods followed by a successfully transmission frame, where a collision period is composed of (i) idle slots (due to binary exponential backoff) and (ii) a collided frame or a RTS, depending on whether or not the floor acquisition mechanism is used. Figure 3.1 gives the timing structure of a fluid chuck. In particular, Figure 3.1 (a) and (b) show, respectively, how IEEE 802.11 DCF operates with and without the RTS/CTS mechanism. A *MAC fluid* consists of a sequence of fluid chunks. Let P_t represent the probability with which a node sends a frame (note that this probability depends on the backoff timer of the node), and M the number of active nodes. The condition under which a frame can be successfully transmitted is that only one node attempts to send its frame. Hence, the probability of successful transmission in the WLAN, P_s,

Table 3.1: IEEE 802.11 system parameters.

Channel Bit Rate	1 Mb/s
Slot Time	20 μsec
SIFS	10 μsec
DIFS	50 μsec
CW_{min}	32
CW_{max}	1024
Phy preamble	144 bits
Phy header	48 bits
MAC header & FCS	224 bits
ACK	112 bits
RTS	160 bits
CTS	112 bits

can be expressed as

$$P_s = M \cdot P_t \cdot (1 - P_t)^{M-1},$$

and the probability that there are no transmission activities (idle periods), P_i, is

$$P_i = (1 - P_t)^M.$$

If we assume that M is large enough, then by using the approximation $(1-x)^y \approx e^{-xy}$, the time till the next transmission attempt can be approximated as a random variable with the exponential distribution. The parameter λ for the assumed exponential distribution is determined as follows. The current backoff window size, B_i, determines the waiting time until which each node, i, should delay its transmission. Each node employs the identical binary backoff mechanism specified in the IEEE 802.11 standard, and independently chooses a random value of B_i from the uniform distribution $[0, \widehat{CW}_i]$, where $\widehat{CW}_i$ is the current maximum contention window of node i. (Therefore, the waiting time at each node is i.i.d.) Let $\overline{b} = E(B_i)$ denote average backoff window size (average waiting time) for one node in a MAC fluid, and then average rate for one node is $\frac{1}{\overline{b}}$. The average waiting time in system wide therefore is

$$\frac{1}{\lambda} = \frac{1}{\sum_{i=1}^{M} \frac{1}{\overline{b}}},$$

and the average rate in system wide is

$$\lambda = M \cdot \frac{1}{\bar{b}}. \tag{3.1}$$

As will be seen, we will express the analytic components — the fluid chunk and the MAC fluid — as functions of the parameter, λ, which in turn is a function of the number of active nodes in Eq. (3.1) (The details on how to derive the attempt rate, λ, will be given in Section 3.3.). Therefore, the proposed fast simulation framework is devised in the way that it only keeps track of the current number of active nodes (see Figure 3.5) since the information is sufficient to estimate the expected throughput and delay in the proposed model.

3.2.2 Derivation of the Frame Service Time

As described in subsection 3.2.1, a MAC fluid is composed of a sequence of fluid chunks, each of which in turns consists of zero or more collision periods followed by a successful frame transmission. We define the length of a fluid chunk, i.e., the time it takes to successfully transmit a frame, as the *frame service time*, Y. The frame service time plays an important role for the overall analysis, and is depicted in Figure 3.2. To facilitate analysis of the frame service time Y, we define below several random variables and express their relation in both cases in which the RTS/CTS mechanism is and is not employed.

- F: the r.v. representing the total length of collision periods in a fluid chunk;
- N: the r.v. representing the number of collisions in a frame service time or a fluid chunk;
- N': the r.v. representing the total number of idle periods in a frame service time or a fluid chunk; note that $N' = N + 1$;
- C_i: the r.v. representing the ith collision period;
- CW_i: the r.v. representing the number of idle slots before the ith collision or the successful transmission;
- CF: the r.v. representing the size of a collided frame;
- X: the r.v. representing the time it takes to successfully transmit a frame;

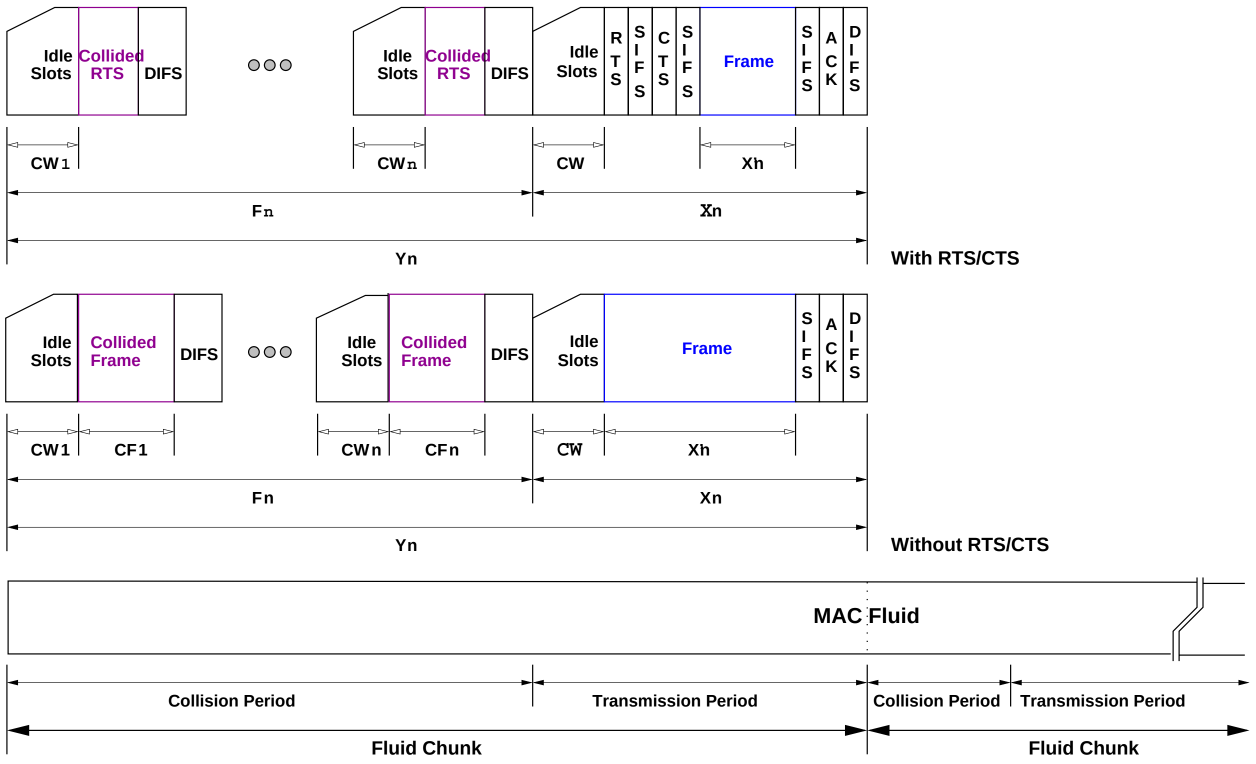


Figure 3.2: Frame service time and its components.

- X': the r.v. representing the size of a frame (note that the distribution of CF is the same as that of X');
- $DIFS$ and $SIFS$: the system parameters whose values are obtained from Table 3.1.
- t_{ack}: another system parameter defined as $t_{ack} = ACK/1(Mb/s)$.

In the case that DCF employs the RTS/CTS mechanism, we have

$$\begin{aligned} Y &= F + X, && (3.2)\\ F &= \sum_{i=1}^{N} C_i, && (3.3)\\ C_i &= CW_i + t_{rts} + DIFS, \text{ and} && (3.4)\\ X &= CW_{N'} + t_{rts} + SIFS + t_{cts} + SIFS + \\ &\quad X' + SIFS + t_{ack} + DIFS, && (3.5) \end{aligned}$$

where t_{rts} and t_{cts} are obtained from system parameters specified in Table 3.1 as $t_{rts} = RTS/1(Mb/s)$, and $t_{cts} = CTS/1(Mb/s)$. In the case that RTS/CTS mechanism is not employed, all the above equations remain valid except Eqs. (3.4)–(3.5) which should be modified as follows:

$$\begin{aligned} C_i &= CW_i + CF + DIFS, \text{ and} && (3.6) \\ X &= CW_{N'} + X' + SIFS + t_{ack} + DIFS. && (3.7) \end{aligned}$$

Note that we implicitly include the physical layer overhead (preamble and header) and MAC header in the above equations (as indicated in Table 3.1).

Under both cases, we have (i) F and X are independent of each other, (ii) C_i, $i \geq 1$, are independent of each other, and (iii) CW_i and X' are independent of each other. Let $Y^*(s)$, $F^*(s)$, $C^*(s)$, $X^*(s)$, $CW^*(s)$, and $X'^*(s)$ denote, respectively, the Laplace transform of the probability density function associated with Y, F, C, X, CW, and X'. Then, we have

$$\begin{aligned} Y^*(s) &= F^*(s) \cdot X^*(s), && (3.8) \\ X^*(s) &= CW^*(s) \cdot X'^*(s) \cdot \\ & \quad e^{-(t_{rts}+SIFS+t_{cts}+SIFS+SIFS+t_{ack}+DIFS)s} && (3.9) \end{aligned}$$

in the case that the RTS/CTS mechanism is used, or

$$X^*(s) = CW^*(s) \cdot X'^*(s) \cdot e^{-(SIFS+t_{ack}+DIFS)s} \quad (3.10)$$

in the case that the RTS/CTS mechanism is not used, and

$$\begin{aligned} F^*(s) = E[e^{-s\sum_{i=1}^{N} C_i}] &= \sum_{n=0}^{\infty} E[e^{-sC_1}] \cdots E[e^{-sC_n}] \cdot P[N=n] \\ &= \sum_{n=0}^{\infty} [C^*(s)]^n \cdot P[N=n]. \end{aligned} \quad (3.11)$$

Note that the last equality of Eq. (3.11) results from the fact that C_i, $i \geq 1$, are mutually independently of one another. Since the z transform of N is $N(z) = \sum_{n=0}^{\infty} P[N=n] \cdot z^n$, Eq. (3.11) can be rewritten as

$$F^*(s) = N(C^*(s)). \tag{3.12}$$

Now what is left is to derive $C^*(s)$. By Eq. (3.4), we have

$$C^*(s) = CW^*(s) \cdot e^{-(t_{rts}+DIFS)s} \tag{3.13}$$

in the case that the RTS/CTS mechanism is used, or

$$C^*(s) = CW^*(s) \cdot CF^*(s) \cdot e^{-DIFSs} \tag{3.14}$$

in the case that the RTS/CTS mechanism is not used. Since $\overline{c^k} = (-1)^k \cdot C^{*(k)}(s)|_{s=0}$, we have

$$\overline{c} = -C^{*(1)}(s)|_{s=0} = \overline{cw} + t_{rts} + DIFS \tag{3.15}$$

in the case that the RTS/CTS mechanism is used, or

$$\overline{c} = -C^{*(1)}(s)|_{s=0} = \overline{cw} + \overline{cf} + DIFS \tag{3.16}$$

in the case that the RTS/CTS mechanism is not used, where $\overline{cw}$ and $\overline{cf}$ are, respectively, the expectation of CW and CF. Note that the distribution of CF is the same as that of X' and is given.

To derive the distribution of CW, we note that whether or not each node attempts to transmit in a time slot is determined by its current backoff window size. Each slot is either idle or used, but the number of consecutive idle slots is bounded by the maximum contention window size, CW_{max} (Table 3.1) within each frame service time. In other words, the number, CW, of idle slots before each collision or a successful transmission in a fluid chunk cannot exceed CW_{max}. We will derive CW based on this observation.

Given that the number of slots till a transmission attempt is approximated as a random variable with exponential distribution (whose rate is determined by Eq. (3.1)), the probability that there is no transmission attempt in k slots is $e^{-\lambda k}$. Given that the maximum contention window size of

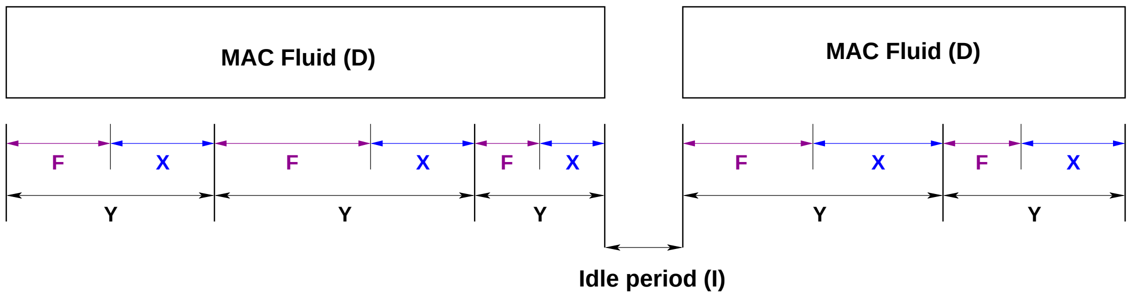


Figure 3.3: MAC fluid structure.

$CW_{max} = m$, we can derive the expected number, $\overline{cw}$, of idle slots before each collision or successful transmission as

$$\begin{aligned}\overline{cw} &= E[P(CW \leq y | CW \leq m)] \\ &= \frac{1}{1-e^{-\lambda m}} \cdot \int_0^m y\lambda e^{-\lambda y} dy = \frac{1-(m\lambda+1)e^{-\lambda m}}{\lambda(1-e^{-\lambda m})}.\end{aligned} \tag{3.17}$$

3.2.3 Derivation of the Length of a MAC Fluid

Recall that a fluid chunk is made up of zero or more collision periods followed by a successful frame transmission, and a sequence of fluid chunks constitute a MAC fluid. Figure 3.3 gives the structure of a MAC fluid.

Total number of idle slots in a fluid chunk

We first derive the total number, $CW_T \triangleq \sum_{i=1}^{N'} CW_i$, of idle slots in a frame service time. Since the number, CW_i, of idle slots before the ith collision or the successful frame transmission is independently and identically distributed with each other, the Laplace transform, $CW_T^*(s)$, of the probability density function associated with CW_T can be expressed as

$$\begin{aligned}CW_T^*(s) &= E[e^{-s\sum_{i=1}^{N'} CW_i}] \\ &= \sum_{n=0}^{\infty} E[e^{-sCW_1}] \cdots E[e^{-sCW_n}] \cdot P[N'=n] \\ &= \sum_{n=0}^{\infty} [CW^*(s)]^n \cdot P[N'=n].\end{aligned} \tag{3.18}$$

Additionally, since the z transform, $N'(z)$, of N' is $\sum_{n=0}^{\infty} P[N' = n] \cdot z^n$, we have

$$CW_T^*(s) = N'(CW^*(s)). \tag{3.19}$$

The expected value of CW_T can be obtained as follows.

$$\begin{aligned} \overline{cw_t} &= (-1) \cdot CW_T^{*(1)}(s)|_{s=0} \\ &= (-1) \cdot N'^{*(1)}(CW^*(0)) \cdot CW^{*(1)}(0) = \overline{n'} \cdot \overline{cw}. \end{aligned} \tag{3.20}$$

Length of a MAC fluid

Let D denote the random variable that represents the length of a MAC fluid, Y_i the random variable of the ith frame service time, $CW_{T,1}$ the total number of idle slots in Y_1, and K_1 the total number of transmission attempts in Y_1. To derive $E[e^{-sD}]$, we first consider $E[e^{-sD}|Y_1 = y, CW_{T,1} = \ell, K_1 = k]$. The condition that totally $K_1 = k$ attempts for transmission in Y_1 implies that there will be *at least* $K_1 = k$ fluid chunks by the end of this MAC fluid. During the execution of a subsequent fluid chunk, new attempts may be made, thus "spawning off" more fluid chunks.

Let the frame service time in which the jth packet $(1 \le j \le k)$ is successfully transmitted be denoted as $Y_{(j)}$, and the number of transmission attempts in $Y_{(j)}$ be denoted as $K_{(j)}$. The fact that $K_{(j)}$ transmissions are attempted implies another $K_{(j)}$ fluid chunks are generated. Let the sum of $Y_{(j)}$ and the frame service times that are spawned off as a result of $K_{(j)}$ transmission attempts in $Y_{(j)}$ be denoted as T_j. It can be shown that T_j, $j = 1, \ldots, k$ has the same distribution of D. Moreover, conditioned on $Y_1 = y$, $CW_{T,1} = \ell$, and $K_1 = k$, we have $D = y + T_1 + \cdots + T_k$. Hence,

$$\begin{aligned} &E[e^{-sD}|Y_1 = y, CW_{T,1} = \ell, K_1 = k] \\ &\quad = E[e^{-s(y+T_k+T_{k-1}+\ldots+T_1)}] \\ &\quad = E[e^{-sy}] \cdot E[e^{-sT_k}] \cdots E[e^{-sT_1}] = e^{-sy} \cdot (D^*(s))^k. \end{aligned} \tag{3.21}$$

Under the observation that the number of transmission attempts is Poisson distributed (see Eq. (3.1) and related explanation), we have

$$\begin{aligned}
&E[e^{-sD}|Y_1 = y, CW_{T,1} = \ell] \\
&= \sum_{k=0}^{\infty} E[e^{-sD}|Y_1 = y, CW_{T,1} = \ell, K_1 = k] \cdot P[K_1 = k] \\
&= \sum_{k=0}^{\infty} e^{-sy} \cdot (D^*(s))^k \cdot \frac{(\lambda \cdot \ell)^k}{k!} \cdot e^{-\lambda \cdot \ell} \\
&= e^{-sy} e^{-\ell(\lambda - \lambda D^*(s))}.
\end{aligned} \tag{3.22}$$

Unconditioning on $CW_{T,1} = \ell$, we have

$$\begin{aligned}
E[e^{-sD}|Y_1 = y] &= e^{-sy} \cdot \int_0^{\infty} e^{-\ell(\lambda - \lambda D^*(s))} dCW_T(\ell) \\
&= e^{-sy} \cdot CW_T^*(\lambda - \lambda D^*(s)).
\end{aligned} \tag{3.23}$$

Finally, unconditioning on $Y_1 = y$, we have

$$\begin{aligned}
D^*(s) = E[e^{-sD}] &= \int_0^{\infty} e^{-sy} \cdot CW_T^*(\lambda - \lambda D^*(s)) dY(y) \\
&= CW_T^*(\lambda - \lambda D^*(s)) \cdot Y^*(s).
\end{aligned} \tag{3.24}$$

Recall that $Y^*(s) = F^*(s) \cdot X^*(s)$, and hence

$$\begin{aligned}
D^*(s) &= Y^*(s) \cdot CW_T^*(\lambda - \lambda D^*(s)) \\
&= F^*(s) \cdot X^*(s) \cdot N(CW(\lambda - \lambda D^*(s))).
\end{aligned} \tag{3.25}$$

As $\overline{d^{(k)}} = (-1)^k \cdot D^{*(k)}(0)$, we have

$$\begin{aligned}
\overline{d} &= (-1) \cdot D^{*(1)}(0) \\
&= (-1) \cdot \Big[Y^{*(1)}(s) \cdot CW_T^*(\lambda - \lambda D^*(s)) \\
&+ Y^*(s) \cdot CW_T^{*(1)}(\lambda - \lambda D^*(s)) \cdot (-\lambda D^*(s)) \Big]_{s=0},
\end{aligned} \tag{3.26}$$

where $Y^{*(1)}(0) = F^{*(1)}(0) \cdot X^*(0) + F^*(0) \cdot X^{*(1)}(0) = -(\overline{f} + \overline{x})$ (the second equality results from $X^*(0) = 1$ and $F^*(0) = 1$), $CW_T^*(\lambda - \lambda D^*(s))|_{s=0} = CW_T^*(0) = N(CW^*(0)) = 1$, and

$CW_T^{*(1)}(\lambda - \lambda D^*(s))|_{s=0} = CW_T^{*(1)}(0) \cdot (-\lambda D^{*(1)}(0)) = (-1) \cdot \overline{cw_t} \cdot \lambda \cdot \overline{d}.$

Finally, we have

$$\overline{d} = (\overline{f} + \overline{x}) + \lambda \cdot \overline{d} \cdot \overline{cw_t}. \tag{3.27}$$

Rearranging Eq. (3.27), we have

$$\overline{d} = \frac{\overline{f} + \overline{x}}{1 - \lambda \cdot \overline{cw_t}} = \frac{\overline{f} + \overline{x}}{1 - \lambda \cdot \overline{cw} \cdot \overline{n'}}. \tag{3.28}$$

The terms needed in Eq. (3.28), i.e., the expected frame service time $\overline{f}$, the expected total number of idle slots in a frame service time $\overline{n'}$, and the expected time to successfully transmit a packet $\overline{x}$, can be obtained as follows:

$$\begin{aligned} \overline{f} &= (-1) \cdot F^{*(1)}(0) = (-1)(N^{(1)}(C^*(0))) \cdot C^{*(1)}(0) \\ &= \overline{n} \cdot \overline{c}, \end{aligned} \tag{3.29}$$

where $\overline{c}$ is given in Eqs. (3.15) or (3.16):

$$\begin{aligned} \overline{x} &= -X^{*(1)}(0) \\ &= \overline{cw} + t_{rts} + SIFS + t_{cts} + SIFS + \overline{x'} \\ &\quad + SIFS + t_{ack} + DIFS, \end{aligned} \tag{3.30}$$

in the case that the RTS/CTS mechanism is employed, or

$$\begin{aligned} \overline{x} &= -X^{*(1)}(0) \\ &= \overline{cw} + \overline{x'} + SIFS + t_{ack} + DIFS, \end{aligned} \tag{3.31}$$

in the case that the RTS/CTS mechanism is not used, and

$$\overline{n'} = \overline{n} + 1, \tag{3.32}$$

where $\overline{n} = N^{(1)}(1) = \frac{(1-e^{-\lambda}-\lambda e^{-\lambda})}{\lambda e^{-\lambda}}$.

With all the expressions properly plugged in, the expected length of a MAC fluid can be expressed as

$$\overline{d} = \frac{\overline{n} \cdot \overline{c} + \overline{x}}{1 - \lambda \cdot \overline{cw}(\overline{n} + 1)}. \tag{3.33}$$

3.2.4 Length of an Idle Period

An idle period separates consecutive MAC fluids. Since each MAC fluid is triggered by one or more transmission attempts and the time till a transmission attempt is exponentially distributed with rate λ (Eq. (3.1)), we estimate an idle time between two consecutive MAC fluid as follows:

$$\overline{i} = \frac{1}{\lambda}. \tag{3.34}$$

3.2.5 Throughput

Let $\overline{n}_f$ denote the number of frame service times in a MAC fluid. Then, the expected throughput, T, can be expressed as

$$T = \frac{\overline{n}_f \times \overline{x'}}{\overline{d} + \overline{i}}, \tag{3.35}$$

where $\overline{n}_f$ can be approximated as

$$\overline{n}_f = \frac{\overline{d}}{\overline{f} + \overline{x}}. \tag{3.36}$$

We will elaborate in the next section on how we derive the attempt rate, λ, needed as the input to the model.

3.3 Derivation of The Attempt Rate

To calculate the numerical results from the model derived in Section 3.2, we need to know the value of the attempt rate λ. In this section, we explain how we determine the attempt rate, λ.

Recall that in Eq. (3.1), we express $\lambda = M/\overline{b}$, where M is the number of nodes and $\overline{b}$ is the average backoff window size. To determine λ, we take an iterative approach to determine $\overline{b}$ based on M. Succinctly, in each iteration we calculate, based on the average backoff window size $\overline{b}$ calculated from the previous iteration, the probability of collision p. This probability is then used to calculate (along with M) the new average backoff window size to be used in the next iteration. In what follows, we first derive the relationship between $\overline{b}$ and p. Then we describe in detail how the iterative algorithm operates.

3.3.1 Relationship between $\overline{b}$, p and λ

Let $\widetilde{CW}_i$ be the contention window after the ith collision occurs, c_i the probability that the current contention window size is $\widetilde{CW}_i$, and m the index for the maximum contention window size. Then, given the probability, p, of collision, we have

$$c_i = \begin{cases} c_0 \cdot p^i & 1 \leq i \leq m-1, \\ c_0 \cdot \sum_{k=m}^{\infty} p^k = \frac{c_0 \cdot p^m}{1-p} & i = m. \end{cases} \tag{3.37}$$

Since $\sum_{k=0}^{m} c_k = 1$, we have

$$\begin{aligned} & c_0 \cdot (1 + p + p^2 + \cdots + p^{m-1} + \frac{p^m}{1-p}) = 1, \text{ or} \\ & c_0 = 1 - p. \end{aligned} \tag{3.38}$$

Let b_i be the average backoff window when the contention window is $\widetilde{CW}_i$. Then, $b_i = \frac{1}{\widetilde{CW}_i} \sum_{i=0}^{\widetilde{CW}_i - 1} i = \frac{\widetilde{CW}_i - 1}{2}$. In addition, we have $\widetilde{CW}_i = 2^i \cdot \widetilde{CW}_0$ for $1 \leq i \leq m$. Thus, given the probability of collision, p, the average backoff window size, $\overline{b}$, can be expressed as

$$\begin{aligned} \overline{b} &= b_0 \cdot c_0 + b_1 \cdot c_1 + \cdots + b_m \cdot c_m \\ &= \frac{\widetilde{CW}_0 (1-p)}{2} \cdot \left\{ 1 + 2p + \cdots + (2p)^{(m-1)} + \frac{(2p)^m}{1-p} \right\} \\ &\quad - \frac{1}{2} \cdot \{(1-p) + (1-p)p + \cdots + p^m\} \\ &= \frac{\widetilde{CW}_0}{2(1-2p)} \cdot (1 - p - p \cdot (2p)^m) - \frac{1}{2}. \end{aligned} \tag{3.39}$$

At the point, we have derived the expression of $\overline{b}$ as a function of p. Conversely, given the value of $\overline{b}$ (and hence the aggregate attempt rate $\lambda = M/\overline{b}$), the probability, p, of collision can be determined to be

$$p = 1 - e^{-\lambda} - \lambda \cdot e^{-\lambda}. \tag{3.40}$$

3.3.2 The Iterative Algorithm

The iterative algorithm constructs the sequence $\{\ \overline{b}^{(i)}, i = 0, 1, 2, \ldots\ \}$ as follows. The initial backoff window size $\overline{b}^{(0)}$ is set to $\frac{E(\widehat{CW}^{(0)})-1}{2}$, where the average contention window size $E(\widehat{CW}^{(0)})$ is determined by leveraging Goodman's work [20]. Specifically, Goodman *et al.* proved that the expected number of collisions in a binary backoff algorithm grows asymptotically with $O(logM)$ (Lemma 3 and Theorem 4 in [20]). Thus, the average contention window size, $E(\widehat{CW}^{(0)})$, is bounded by

$$E(\widehat{CW}^{(0)}) < CW_{min} \cdot 2^{K \cdot logM}, \tag{3.41}$$

where $CW_{min}(= CW_0)$ is the initial window size (whose default value is 32 in IEEE 802.11) and K is some arbitrary constant $(K > 0)$ [2].

Given the value of $\overline{b}^{(0)}$ calculated above, the attempt rate $\lambda^{(0)}$ and the probability, of collision, $p^{(0)}$, can be calculated using Eqs. (3.1) and (3.40), respectively. Then, the average backoff window size, $\overline{b}^{(1)}$, for the next iteration can be calculated as the arithmetic mean of $\overline{b}^{(0)}$ and the result, $\overline{b}^{(0)'}$, calculated in Eq. (3.39). The rationale behind using the arithmetic mean is as follows; the input $\overline{b}^{(0)}$ is the worst case backoff window size as we use the bound in Eq. (3.41) as the initial contention window size. On the other hand, the result, $\overline{b}^{(0)'}$, is the best case backoff window size as the worst case backoff window size, $\overline{b}^{0}$, renders the smallest value of λ and p and as a result $\overline{b}^{(0)'}$ is the best case backoff window size. Hence, we use the arithmetic mean of both values. The iteration repeats until the difference between the average backoff window sizes in two consecutive iterations satisfies $|\overline{b}^{(i+1)} - \overline{b}^{(i)}| < \epsilon$, for some pre-determined value of ϵ, or there is no more convergence, which is $|\overline{b}^{(i+1)} - \overline{b}^{(i)}| \geq |\overline{b}^{(i)} - \overline{b}^{(i-1)}|$. We prove in Theorem 1 the iteration algorithm converges.

Theorem 1: The iterative algorithm always converges.

Proof: All the obtained backoff windows are bounded by both the minimum and maximum contention windows, and hence the sequence of all computed backoff window sizes, $\{\ \overline{b}^{(i)}, i = 0, 1, 2, \ldots\ \}$, does not diverge infinitely.

Suppose that the initial backoff window size at the beginning of the ith iteration is $\overline{b}^{(i)}$, the newly generated backoff window size is $\overline{b}^{(i)'}$, and the backoff window size at the end of the ith iteration is

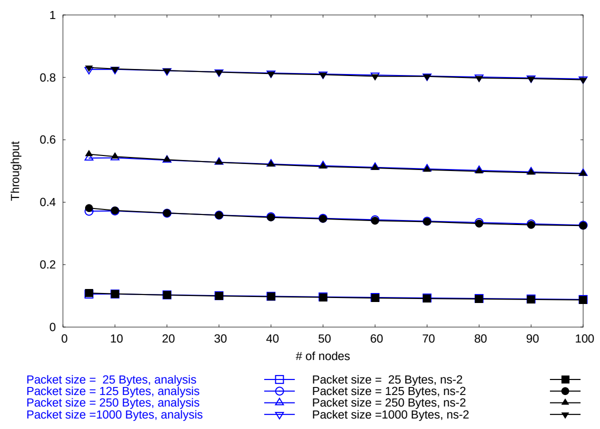
[2] In our extensive simulation study, we found that the value of K does not affect the performance of the iterative algorithm considerably.

$\overline{b}^{(i+1)} = \frac{\overline{b}^{(i)}+\overline{b}^{(i)'}}{2}$. If $\overline{b}^{(i)}$ is larger than $\overline{b}^{(i)'}$, $\overline{b}^{(i+1)}$ is less than $\overline{b}^{(i)}$ but larger than $\overline{b}^{(i)'}$. Otherwise, $\overline{b}^{(i+1)}$ is less than $\overline{b}^{(i)'}$ but larger than $\overline{b}^{(i)}$. Additionally, if $\overline{b}^{(i)}$ be less than $\overline{b}^{(i+1)}$, then, $\lambda^{(i)}$ is larger than $\lambda^{(i+1)}$, and consequently $p^{(i)}$ is larger than $p^{(i+1)}$ and $\overline{b}^{(i)'}$ is larger than $\overline{b}^{(i+1)'}$. This is because the backoff window size is reversely proportional to the collision probability since the attempt rate is reversely proportional to the backoff window size and the collision probability is proportional to the rate (Eqs. (3.1), (3.39) and (3.40)). Additionally, when a smaller backoff window is given, the iteration step produces a larger attempt rate and a larger collision probability, and consequently computes a larger backoff window. Therefore, the difference between $\overline{b}^{(i)}$ and $\overline{b}^{(i)'}$ is larger than the difference between $\overline{b}^{(i+1)}$ and $\overline{b}^{(i+1)'}$, and moreover, the difference between $\overline{b}^{(i)}$ and $\frac{\overline{b}^{(i)}+\overline{b}^{(i)'}}{2}$ (which is equal to $\overline{b}^{(i+1)}$) is also larger than the difference between $\overline{b}^{(i+1)}$ and $\frac{\overline{b}^{(i+1)}+\overline{b}^{(i+1)'}}{2}$ (which is equal to $\overline{b}^{(i+2)}$). In the case that $\overline{b}^{(i)}$ is larger than $\overline{b}^{(i+1)}$, the same conclusion that the difference between $\overline{b}^{(i)}$ and $\overline{b}^{(i+1)}$ is larger than the one between $\overline{b}^{(i+1)}$ and $\overline{b}^{(i+2)}$ can be reached via similar reasoning. Therefore, the difference between $\overline{b}^{(i)}$ and $\overline{b}^{(i+1)}$ decreases at each iteration in the algorithm, and the sequence of backoff windows generated in the algorithm, $\{\overline{b}^{(i)}, i = 0, 1, 2, \ldots\}$, converges to a certain value. Additionally, the second termination condition, $|\overline{b}^{(i+1)} - \overline{b}^{(i)}| \geq |\overline{b}^{(i)} - \overline{b}^{(i-1)}|$, supports that the algorithm converges to a value range and then terminates. $\blacksquare$

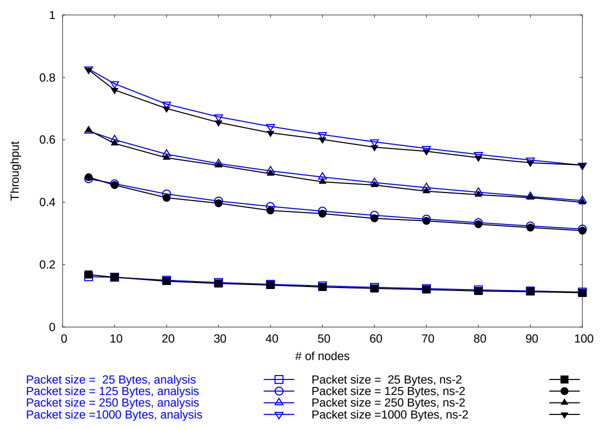
To use the equations derived in Section 3.3.1 in the iterative algorithm, one modification has to be made. Not all the nodes can send their frames in every time slot due to the binary backoff algorithm. Hence, we modulate the number, M, of nodes to transmit frame in a slot time as $M \leftarrow M \cdot (1 - e^{-\lambda})$, as $(1 - e^{-\lambda})$ is the probability that at least one node attempts to transmit in that slot.

3.4 Model Validation

We validated the analytical model presented in Section 3.2 and 3.3 with *ns-2* simulation. Figure 3.4 (a) and (b) depicts the throughput versus the number of nodes under IEEE 802.11 with and without the RTS/CTS mechanism, when the packet size is 25 bytes (10 time slots), 125 bytes (50 time slots), 250 bytes (100 slot times), and 1000 bytes (400 time slots). The numerical results calculated under the derived model agree extremely well with the simulation results obtained in *ns-2* simulation. We will further validate our model in Section 3.5 when we discuss how we incorporate the model in the



(a) IEEE 802.11 with RTS/CTS



(b) IEEE 802.11 without RTS/CTS

Figure 3.4: Comparison between analytical and simulation results under IEEE 802.11 with and without RTS/CTS.

proposed fast simulation framework.

3.5 Simulation Study

In this section, we present the fluid model based, fast simulation framework for simulating IEEE 802.11-operated WLANs. All the packets between the successful transmission of two consecutive frames are abstracted as a fluid chunk, and the simulation engine is instrumented to handle these fluid chunks in the time stepping fashion [45], according to the analytic models derived in Section 3.2. Figure 3.5 shows the blueprint of the proposed framework: the wireless channel provides the simulation engine with the number, M, of nodes that attempt to send frames at each time step, thus eliminating the simulation engine from the burden of simulating activities at the packet level, e.g., packet sending, receiving, collision, and backoff. The simulation engine then determines the values of λ and the system throughput using the results derived in Sections 3.3 and 3.2.

Modification to *ns-2*: To focus on the effect of data transmission related activities in the simulation study and to filter out other second-order effects, we deliberately leave out several protocol operations in IEEE 802.11, e.g., power saving, beaconing, association and re-association between wireless nodes and access points, and hidden terminal effects. In addition, several modifications have been made in *ns-2* [4] to accommodate fluid model based simulation for WLANs. First, we in-

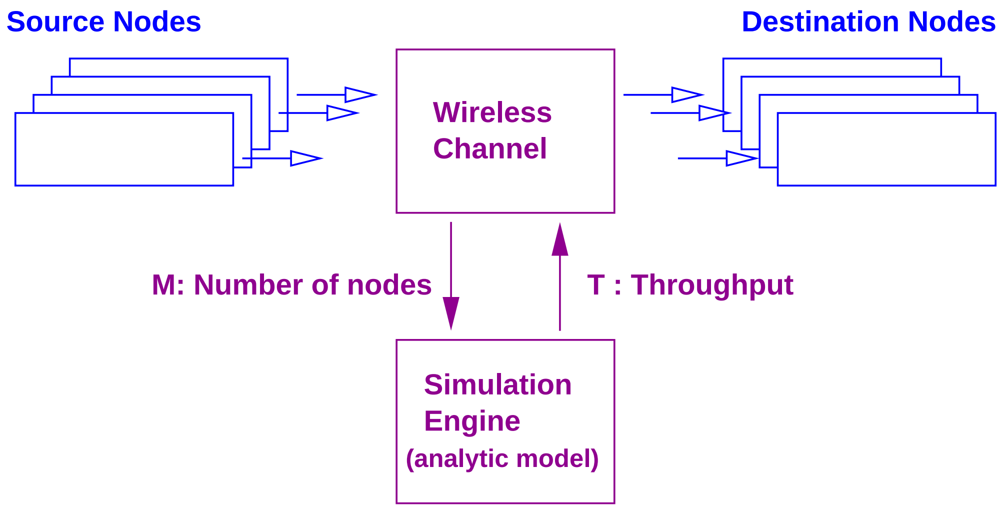
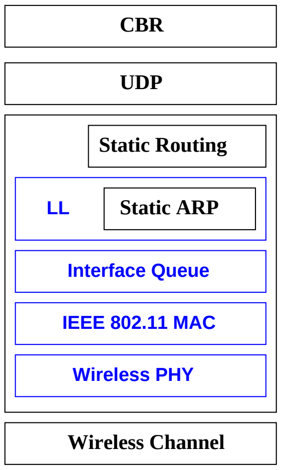


Figure 3.5: The blueprint on fast simulation framework for IEEE 802.11-operated WLANs.

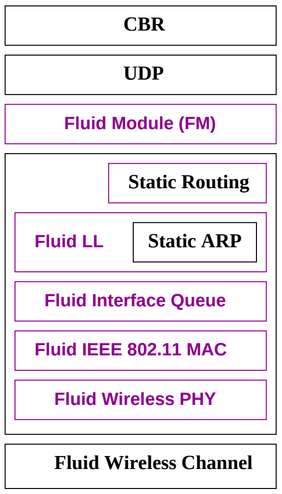
troduce a virtual wireless LAN node.[3] All the wireless nodes communicate with each other through this virtual node. This wireless LAN node uses a static routing algorithm, rather than ad hoc routing protocols (e.g., DSDV, TORA, AODV, and DSR), and also uses a static ARP table. Consequently, the routing and ARP control overheads are not considered in this simulation study. The control overhead is solely due to data transmission related activities and includes RTS/CTS/ACK packets. Second, we introduce a new packet type, called the *fluid rate* packet that describes the fluid chunk of a packet flow within a time period. We also include in *ns-2* (i) several new protocol modules that correspond to the fluid simulation version of existing link layer, MAC layer, physical layer, and channel modules and (ii) a new module, called *fluid module*, that translates a cluster of packets into a *fluid rate* packet or vice versa, or that just deliver *fluid rate* packets when applications directly create them.

As shown in Figure 3.6, we include in *ns-2* two network protocol stacks, one for packet level simulation (Figure 3.6 (a)) and the other for the proposed simulation framework (Figure 3.6 (b)). Except for the above differences, the two simulation models share several common modules: each wireless node has (i) two UDP (or one TCP and one TCP Sink) modules, one as the sender and the other as the receiver, and (ii) one CBR or FTP module that generates packets (destined for the destination node specified by the node identifier). The number of UDP (TCP) flows is equal

[3]This corresponds to the existing virtual LAN node in *ns-2*, and virtually exists to construct a LAN among nodes [4].



(a) For packet level simulation



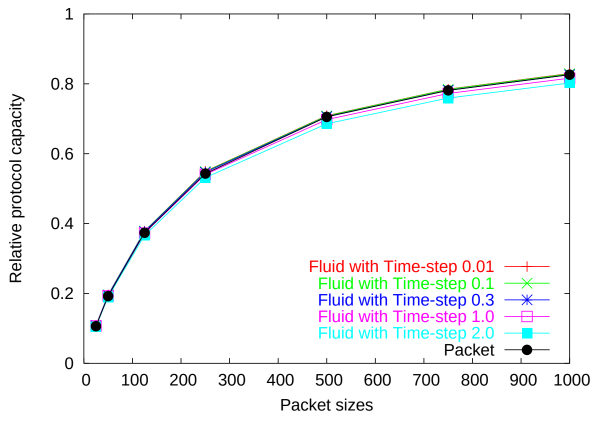
(b) For the proposed simulation framework

Figure 3.6: The network protocol stacks for the packet level simulation and the proposed fluid model based simulation framework.

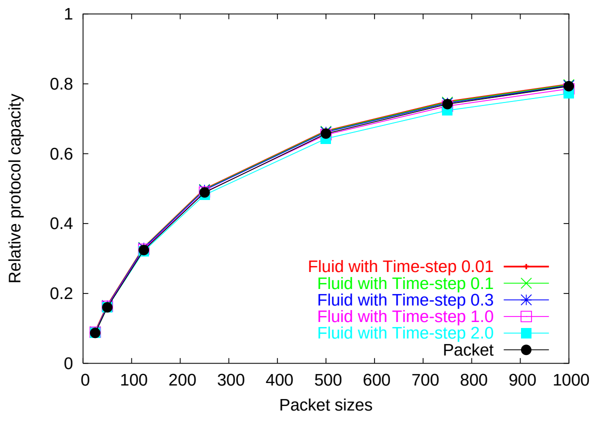
to the number of wireless nodes.

Simulation study: We have conducted a performance study to evaluate the performance of the proposed simulation framework against that of packet level simulation in IEEE 802.11-operated WLANs with respect to the system throughput under a variety of network configurations. Additionally, we also examine how the time step value affects the performance in terms of simulation speed-up and accuracy.

We carry out experiments under two different IEEE 802.11 operational modes: one with the RTS/CTS mechanism and the other without, and with two types of traffic: UDP and TCP traffic. All the simulations are conducted on Linux 2.4.18 on a Pentium 4-1.9 *Ghz* PC with 1 *GBytes* memory memory and with 2 *GBytes* swap memory. We use *ns-2.1b9a*, but upgrade the code of the IEEE 802.11 MAC layer with that available in *ns-2.26*, and carry out each simulation for 60 simulation seconds. We report simulation results with UDP traffic and TCP traffic in Sections 3.5.1 and 3.5.2, respectively.



(a) The number of nodes is 10



(b) The number of nodes is 100

Figure 3.7: Relative protocol capacity and relative errors versus packet sizes, when the number of nodes is 10 and 100, respectively, with the RTS/CTS mechanism.

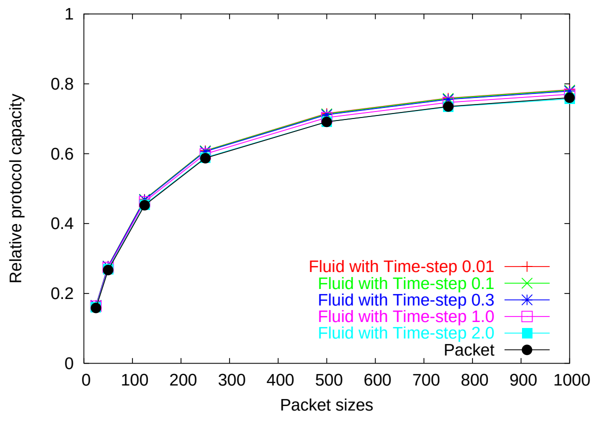
3.5.1 Performance under UDP Traffic

Performance in Terms of relative errors

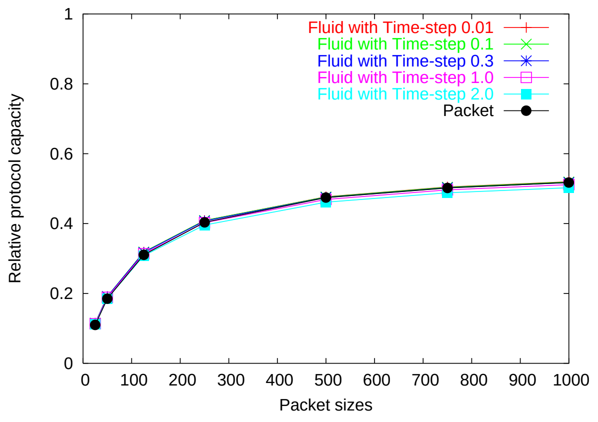
As mentioned in Chapter 1, fluid model based simulation trades accuracy for performance efficiency. Before we explore how scalable the proposed simulation framework is for WLANs equipped with IEEE 802.11, we quantitatively evaluate the discrepancy between results obtained in packet level simulation and in the proposed simulation framework. In both simulation modes, the throughput attained by each receiver node is measured and summed up to give the system throughput.

Figure 3.7 gives relative protocol capacity versus packet sizes, when the number of nodes is fixed at 10 ((a)) and 100 ((b)) respectively, and when the RTS/CTS mechanism is employed. (Note that relative capacity is computed from the total number of received packets counted at the receiving modules.) As shown in Figure 3.7, the relative errors — the differences between the protocol capacity measured in packet level simulation and that in fluid model based simulation, fall within about 2 % in all cases. The larger the time step value, the larger the relative error. Figure 3.8 shows the simulation results in the same configuration as in Figure 3.7, except that the RTS/CTS mechanism is not employed. The same observation can be made.

Figures 3.9–3.10 depict the relative protocol capacity as the number of nodes increases to 100 nodes, when the packet size is fixed at 25 (10, (a)) and 250 bytes (100 slot times, (b)), respectively, in both the cases with and without the RTS/CTS mechanism. The same observation can be made:



(a) The number of nodes is 10



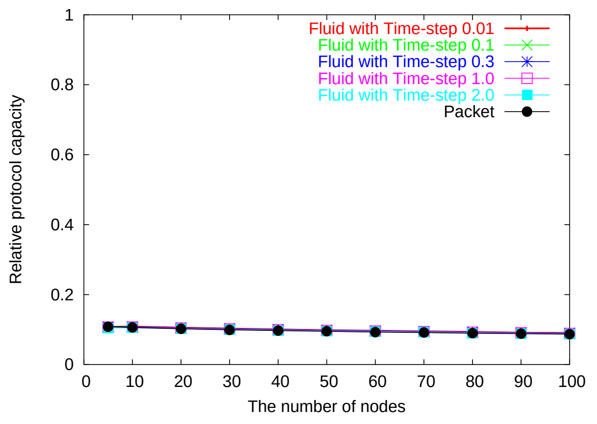
(b) The number of nodes is 100

Figure 3.8: Relative protocol capacity and relative errors versus packet sizes, when the number of nodes is 10 and 100, respectively, without the RTS/CTS mechanism.

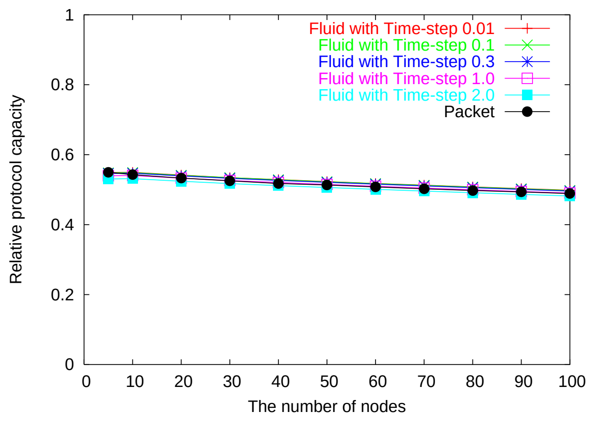
the discrepancy in system throughput under the two simulation modes falls within 2 %.

Results in WLANs of extremely large sizes: To study whether or not the relative errors increase as the network size further increases, we carry out simulation for WLANs of (unreasonably) large sizes. Figures 3.11–3.12 give the relative protocol capacity as the number of nodes increases to 1000 nodes with and without the RTS/CTS mechanism. The packet size is fixed at 25 (10, (a)) and 250 bytes (100 slot times, (b)) respectively. Again the relative errors fall within 2 %.

Results in the presence of varying loads imposed by applications: We have also conducted simulations to study whether or not the relative errors increase when the total aggregate load (imposed by applications) does not fully utilize the maximum bandwidth as allowed in wireless channel. Figures 3.13– 3.14 present the relative protocol capacity as the ratio of the aggregate load to the maximum bandwidth, 1 *Mb*, as allowed in the WLAN, increases (from 50% to 150%) in both the cases with and without the RTS/CTS mechanism. The packet size is fixed at 250 bytes (100 slot times, (a)), and the number of nodes is fixed at 50 ((b)), respectively. The time step value is set to 1.0 (s). As can be observed in both figures, the relative errors fall within 2 % when the WLAN is not saturated.

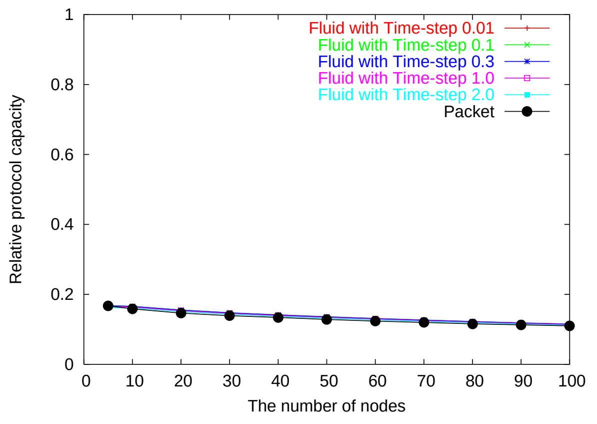


(a) 25 bytes (10 slot times)

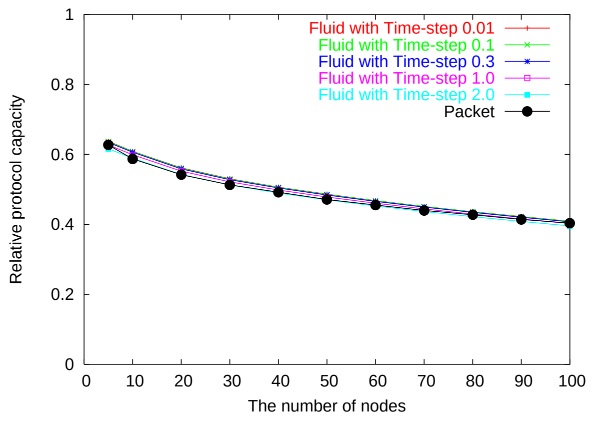


(b) 250 bytes (100 slot times)

Figure 3.9: Relative protocol capacity as the number of nodes increases to 100. The packet size is 25 (10) and 250 bytes (100 slot times), respectively (both with the RTS/CTS mechanism).

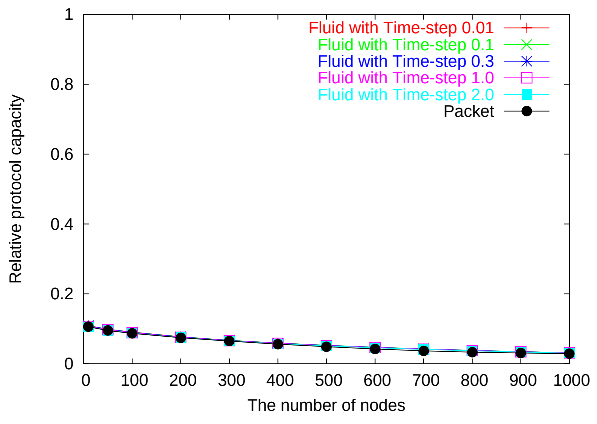


(a) 25 bytes (10 slot times)

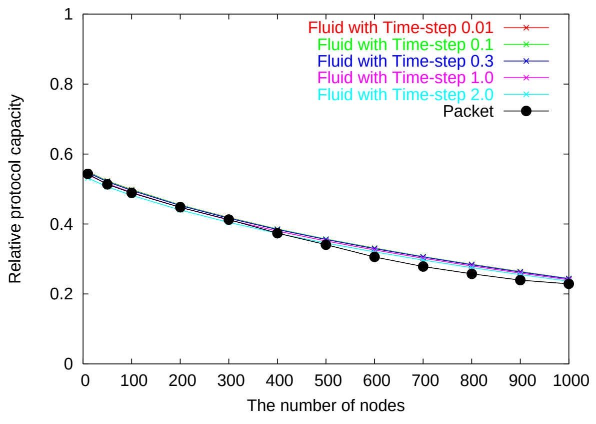


(b) 250 bytes (100 slot times)

Figure 3.10: Relative protocol capacity as the number of nodes increases to 100. The packet size is 25 (10) and 250 bytes (100 slot times), respectively (both without the RTS/CTS mechanism).

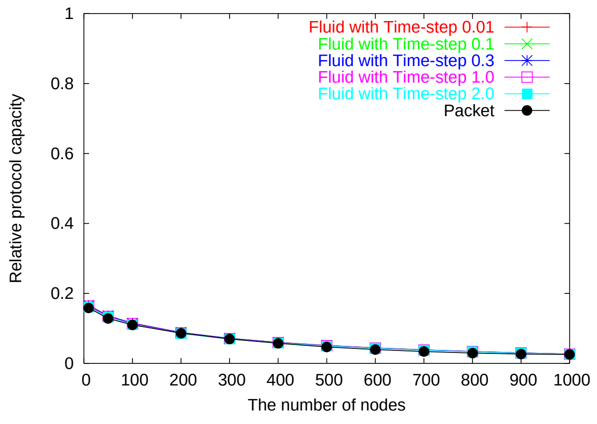


(a) 25 bytes (10 slot times)

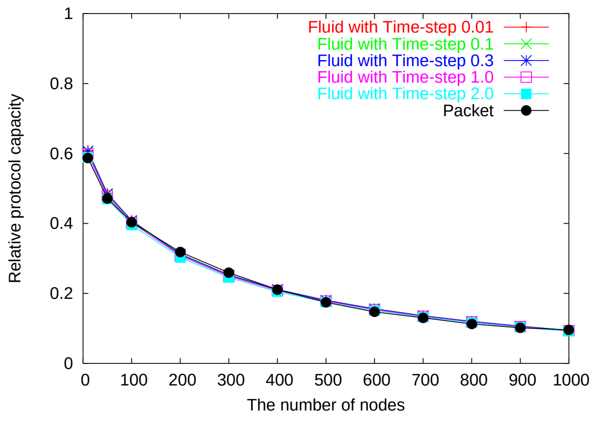


(b) 250 bytes (100 slot times)

Figure 3.11: Relative protocol capacity as the number of nodes increases up to 1000 nodes (with the RTS/CTS mechanism). The packet size is 25 (10) and 250 bytes (100 slot times), respectively.

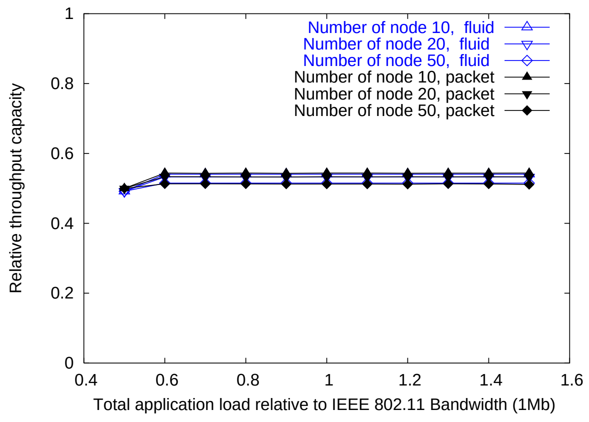


(a) 25 bytes (10 slot times)

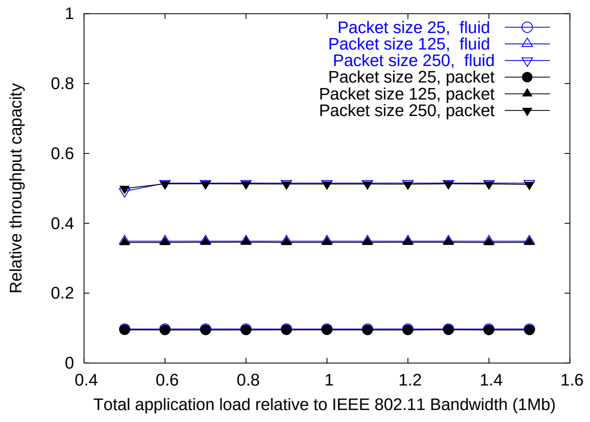


(b) 250 bytes (100 slot times)

Figure 3.12: Relative protocol capacity as the number of nodes increases up to 1000 nodes (without the RTS/CTS mechanism). The packet size is 25 (10) and 250 bytes (100 slot times), respectively.

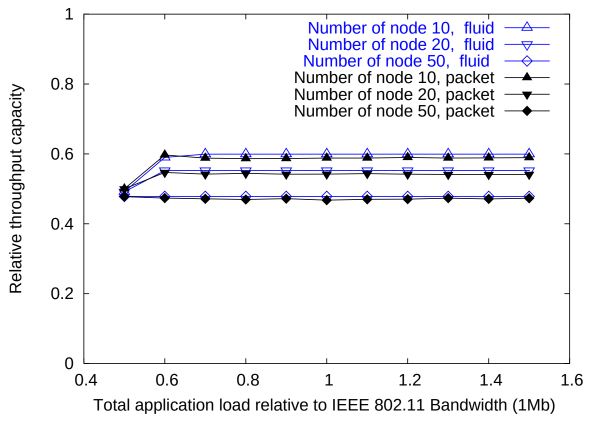


(a) Packet size is 250 bytes

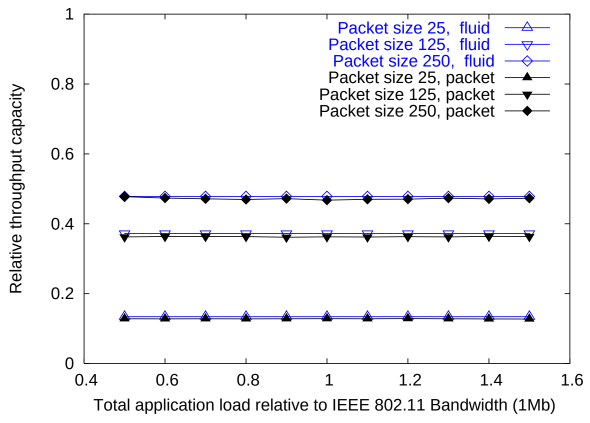


(b) The number of nodes is 50

Figure 3.13: Relative protocol capacity as the total application load increases (the RTS/CTS mechanism is used). The packet size is 250 bytes (100 slot times) in (a) and the number of nodes is 50 in (b), respectively.

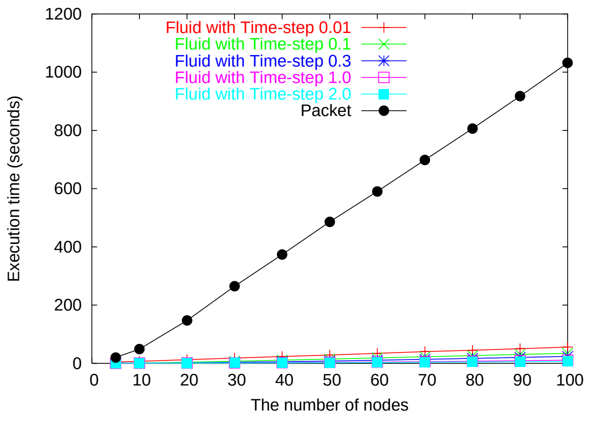


(a) Packet size is 250 bytes

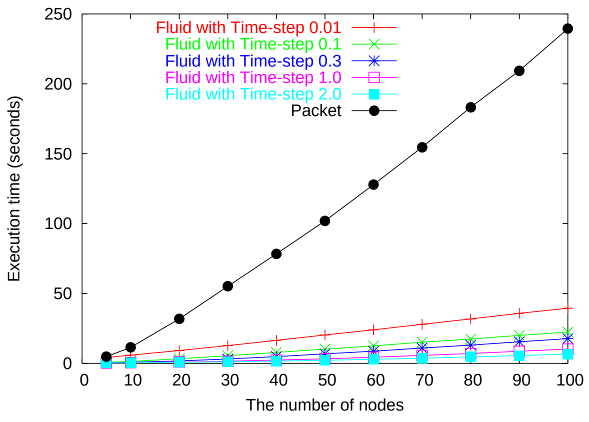


(b) The number of nodes is 50

Figure 3.14: Relative protocol capacity as the total application load increases (the RTS/CTS mechanism is not used). The packet size is 250 bytes (100 slot times) in (a) and the number of nodes is 50 in (b), respectively.



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

Figure 3.15: Execution time versus the number of nodes when the packet size is 25 (10) and 250 bytes (100 slot times), respectively (the RTS/CTS mechanism is used).

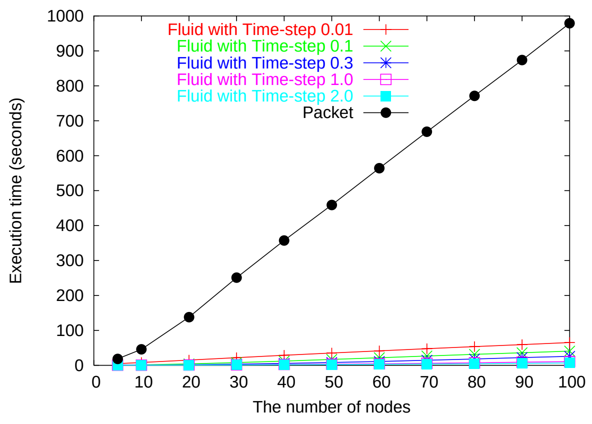
Performance in Terms of execution time

We now evaluate the performance gain that the fluid model based simulation framework can achieve as compared with packet level simulation.

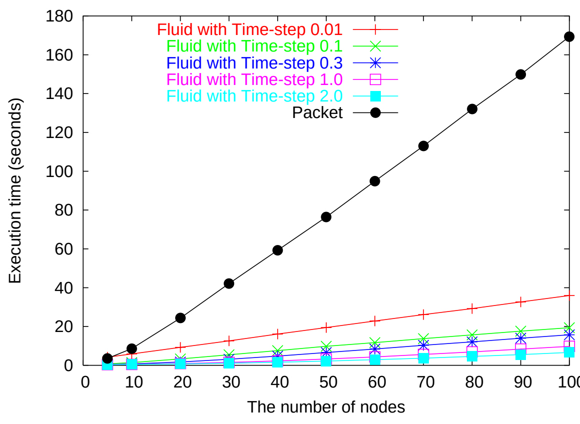
Figure 3.15 depicts the execution time versus the number of nodes when the packet size is fixed with 25 bytes (100 slot times, (a)) and 250 bytes (200 slot times, (b)), respectively in the case that the RTS/CTS mechanism is used. We observe approximately two orders of magnitude improvement. The improvement is especially pronounced when the number of nodes increases or the packet size decreases. This is because under these conditions packet level simulation generates more events to be processed. The same trend can be observed when the packet generation rate of the applications increases, even if the number of nodes is small and/or the packet size is large.

Figure 3.16 presents the simulation results obtained in the same configuration as in Figure 3.15 except that the RTS/CTS mechanism is not used. The level of improvement in the cases without RTS/CTS is a little bit lower than that in the cases with RTS/CTS. This results from the fact that events incurred in processing RTS/CTS frames themselves can be saved in the cases without RTS/CTS.

Results in the multiple-WLAN, multiple-application scenario: To study whether or not the performance improvement levels off as the network size further increases, we evaluate the performance for large scale networks in which multiple WLANs (each of which is made up of multiple



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

Figure 3.16: Execution time versus the number of nodes when the packet size is 25 (10) and 250 bytes (100 slot times), respectively (the RTS/CTS mechanism) is not used.

nodes) exist and are interconnected via bridge wireless nodes, each pair of which are connected by a wired link and which constitute a ring structure (Figure 3.17). In this configuration, both the number of applications per node and the number of WLANs may increase.

Figure 3.18 depicts the execution time versus the number of applications per node when the number of WLANs is 5 (each of which consists of 10 nodes) and when the packet size is fixed at 25 bytes (10 slot times, (a)) and 250 bytes (100 slot times, (b)), respectively. More than two orders of magnitude improvement can be observed in this scenario, and moreover, the improvement becomes more prominent as the number of applications, the number of nodes per WLAN, the number of WLANs, or the packet generation rate increases. Figure 3.19 presents the simulation results in th cases that the RTS/CTS mechanism is not employed. The results show a similar trend as those in Figure 3.18 does.

Results in WLANs of extremely large sizes: We also carry out simulations for WLANs of (unreasonably) large sizes. Figure 3.20 depicts the execution time versus the number of wireless nodes when the packet size is fixed at 25 (10 slot times, (a)) and 250 bytes (100 slot times, (b)), respectively. The speed-up in execution times as a result of using the proposed framework can now be as large as about 100 times. Additionally, the improvement in this case also becomes salient when the packet generation rate increases even in the case of large packet sizes. Figure 3.21 presents the results without the RTS/CTS mechanism. The same observation is observed as done

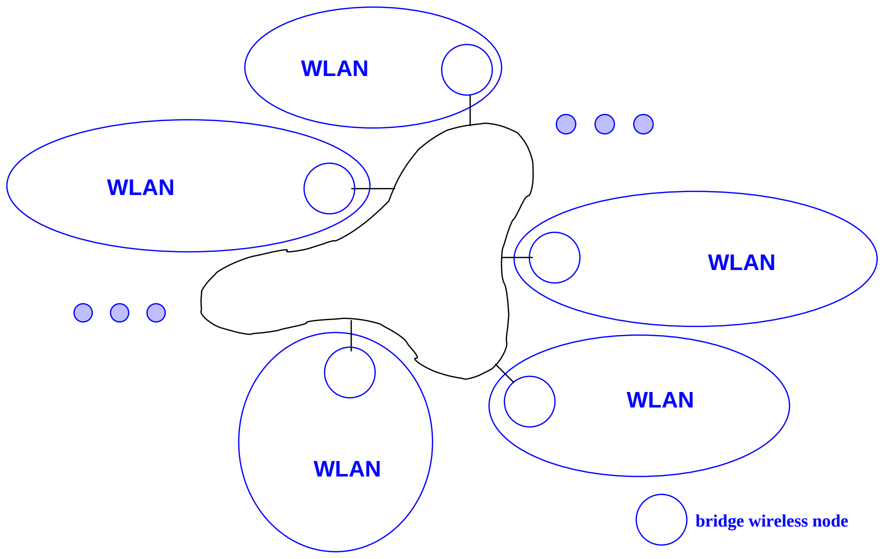


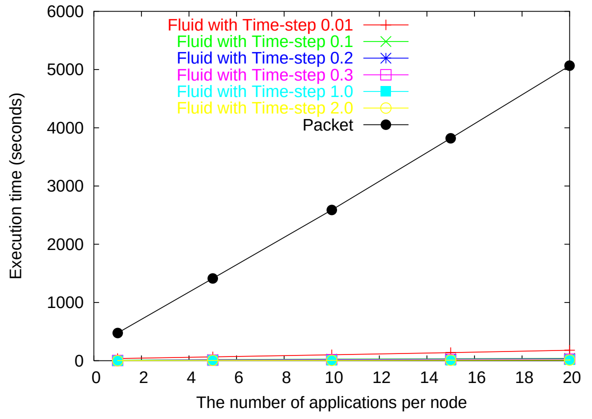
Figure 3.17: Multiple WLANs interconnected via bridge wireless nodes.

in Figure 3.20.

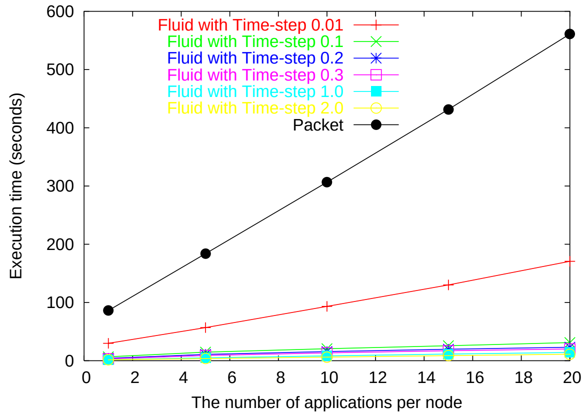
Effects of time step values

As the proposed simulation framework advances system states at every time step, the time step value may affect the performance speed-up and the simulation accuracy.

Figures 3.22–3.25 give the execution time and the relative errors (difference in relative protocol capacity) versus the time step values, in both cases with and without the RTS/CTS mechanism. Figures 3.22–3.23 depict the execution time and relative protocol capacity when the packet size is fixed at 25 bytes (10 slot times) and when the packet size is fixed at 250 bytes (100 slot times), respectively. Although a larger time step value reduces execution time, it also leads to larger relative errors, which suggests that an appropriate value of the time step has to be chosen to ensure fidelity in fluid model based simulation. Figures3.25 and 3.25 gives another set simulation results in the same configuration as in Figures3.22 and 3.23, respectively, except that the RTS/CTS mechanism is not used. The same observation can be made: the larger the time step values, the smaller the execution time, but the larger the relative errors.

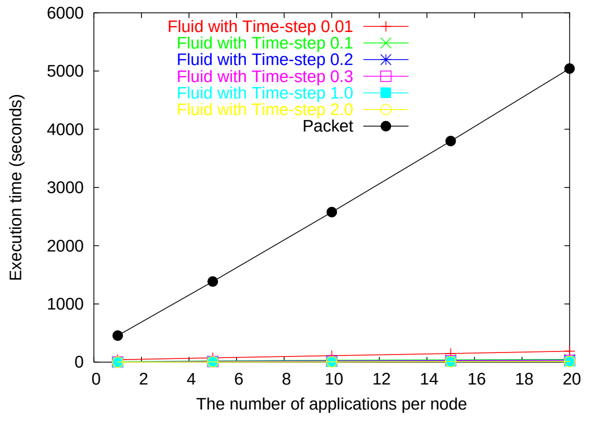


(a) 25 bytes (10 slot times)

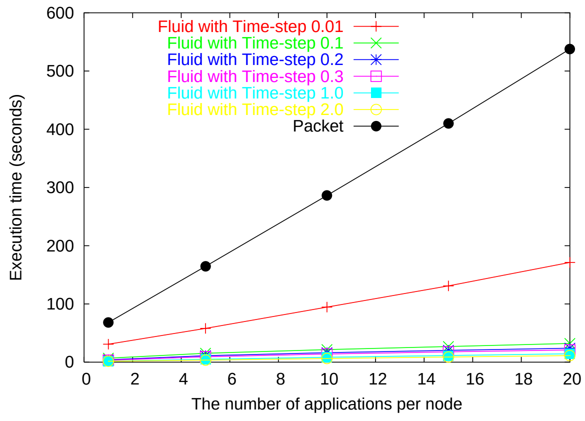


(b) 250 bytes (100 slot times)

Figure 3.18: Execution time versus the number of applications per node, when the number of WLANs is 5 (each of which is made up of 10 nodes) and packet size is 25 (10) and 250 bytes (100 slot times), respectively (the RTS/CTS mechanism is used).

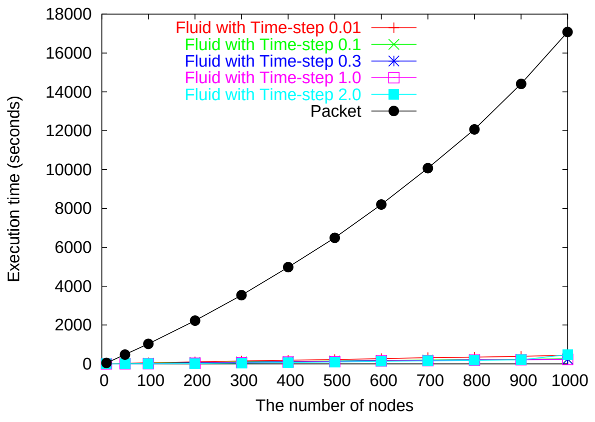


(a) 25 bytes (10 slot times)

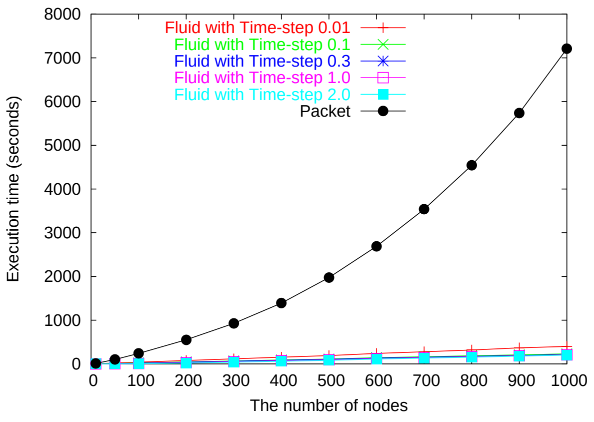


(b) 250 bytes (100 slot times)

Figure 3.19: Execution time versus the number of applications per node, when the number of WLANs is 5 (each of which is made up of 10 nodes) and packet size is 25 (10) and 250 bytes (100 slot times), respectively (the RTS/CTS mechanism is not used).

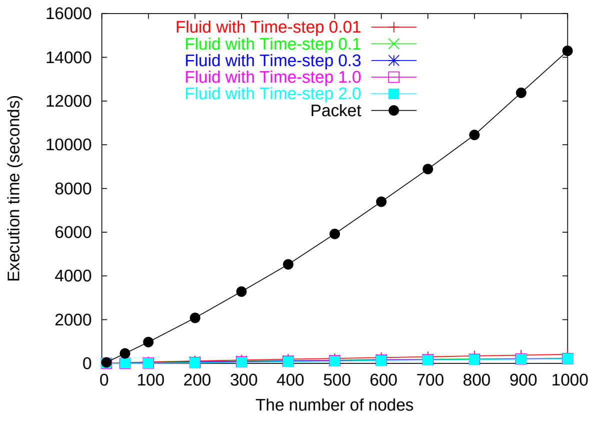


(a) 25 bytes (10 slot times)

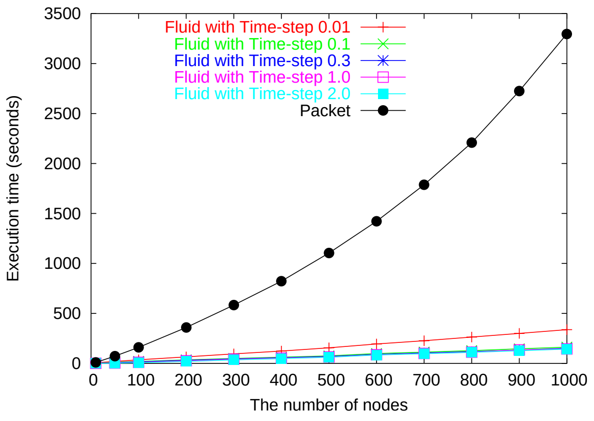


(b) 250 bytes (100 slot times)

Figure 3.20: Execution time versus the number of nodes (up to 1000) when packet size is 25 (10) and 250 bytes (500 slot times), respectively (the RTS/CTS mechanism is used).

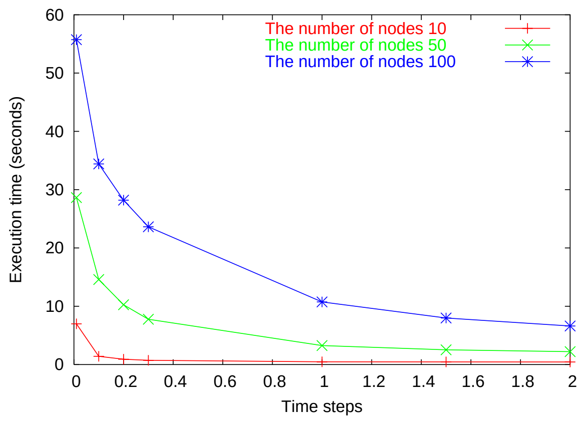


(a) 25 bytes (10 slot times)

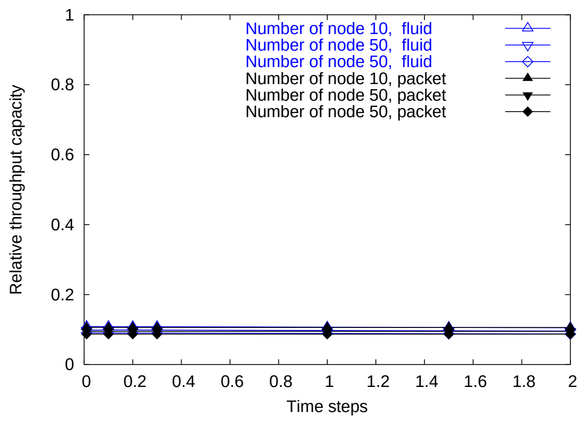


(b) 250 bytes (100 slot times)

Figure 3.21: Execution time versus the number of nodes (up to 1000) when packet size is 25 (10) and 250 bytes (500 slot times), respectively (the RTS/CTS mechanism is not used).

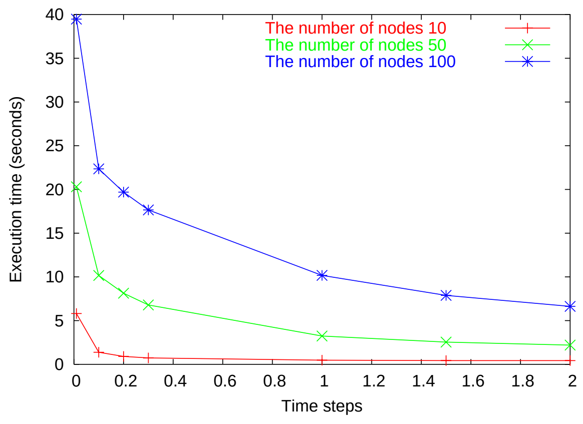


(a) Execution time

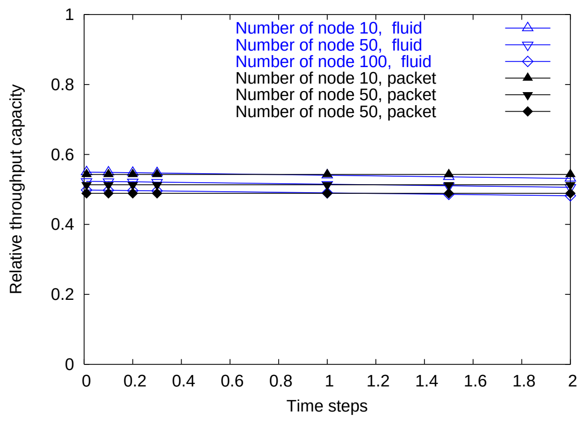


(b) Relative throughput capacity

Figure 3.22: Execution time and relative throughput capacity versus time step values, when packet size is 25 bytes (10 slot times) respectively (the RTS/CTS mechanism is used).

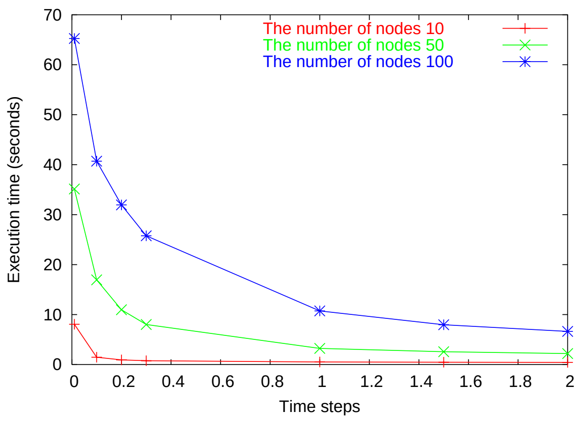


(a) Execution time

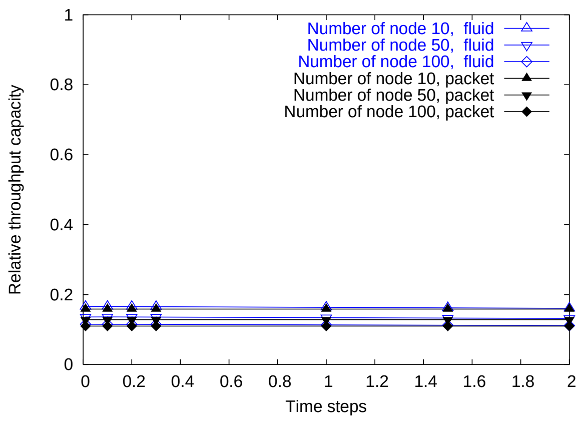


(b) Relative throughput capacities

Figure 3.23: Execution time and relative throughput capacity versus time step values, when packet size is 250 bytes (100 slot times) respectively (the RTS/CTS mechanism is used).

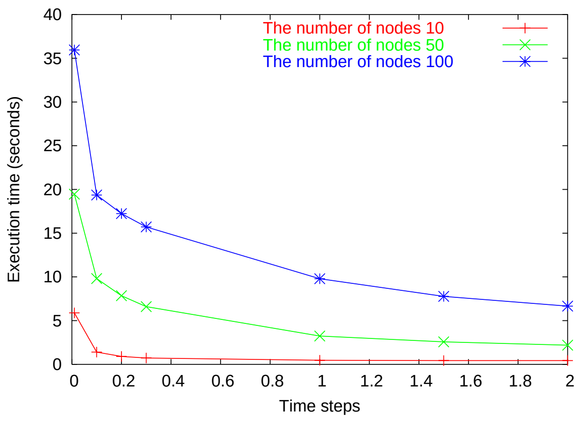


(a) Execution time

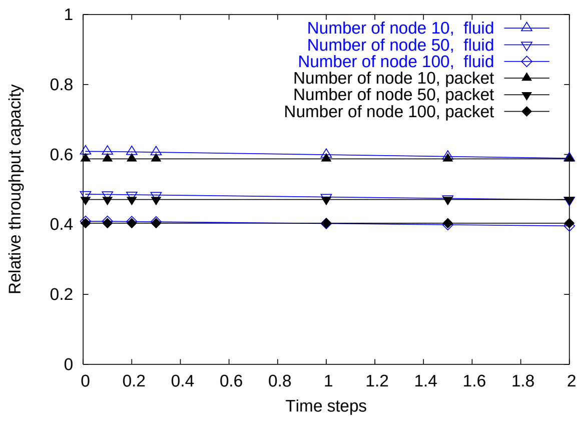


(b) Relative throughput capacity

Figure 3.24: Execution time and relative throughput capacity versus time step values, when packet size is 25 bytes (10 slot times) respectively (the RTS/CTS mechanism is not used).



(a) Execution time



(b) Relative throughput difference

Figure 3.25: Execution time and relative throughput capacity versus time step values, when packet size is 250 bytes (100 slot times) respectively (the RTS/CTS mechanism is not used).

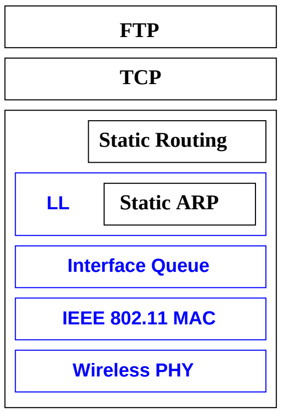
3.5.2 Performance under TCP Traffic

In addition to simulations with CBR traffic on top of UDP, we conduct experiments with FTP on top of TCP in order to verify whether or not fluid model based simulation is effective for different types of traffic.

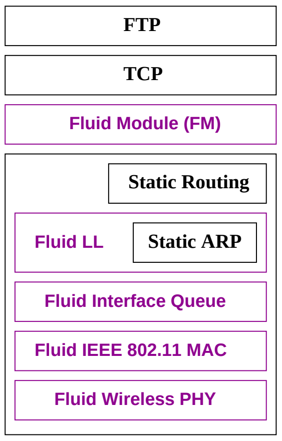
We employ *network calculus based simulation* [29] to simulate complex TCP specifics. As explained in Chapter 2, network calculus based simulation characterizes how TCP congestion control interacts with AQM strategies in the analytic model with the properties of network calculus [9, 13], and regulates TCP flows in a simulation engine with the derived model. As reported in [29], significant improvement can be made in expediting network simulation with TCP traffic, while keeping the error discrepancy acceptably small. The interested readers are referred to [29] for a detailed account of the network calculus models and how they are incorporated into network simulation.

Packet-level simulation employed in this part of study uses the same network configuration as that used in Section 3.5.1, except that now we lay FTP on top of TCP, rather than CBR on top of UDP (Figure 3.26). Fluid model based simulation for IEEE 802.11 is integrated with network calculus based simulation for TCP traffic. The configuration is given in Figure 3.27. *NC-TCP* modules approximate TCP dynamics with the use of network calculus models given in [29]. *Loss Manager* provides packet loss information that occurs in the wireless LAN to the sender and hence saves the execution time incurred in processing ACK packets that come from the TCP receiver sides [29].

Except for the different modules used to generate TCP traffic and to simulate IEEE 802.11 data transmission activities, the two modes of simulation share the common configuration: each wireless node in packet level simulation (network calculus based simulation) has (i) one TCP module and one TCP sink module (two NC-TCP modules), the former as the sender and the latter as the receiver, and (ii) one FTP module that generates FTP traffic on top of TCP (NC-TCP). The number of FTP flows is equal to the number of wireless nodes, and each simulation runs for 500 simulation seconds.



(a) For packet level simulation



(b) For the proposed simulation framework

Figure 3.26: The network protocol stacks for the packet level simulation and the proposed fluid model based simulation framework in the cases with TCP.

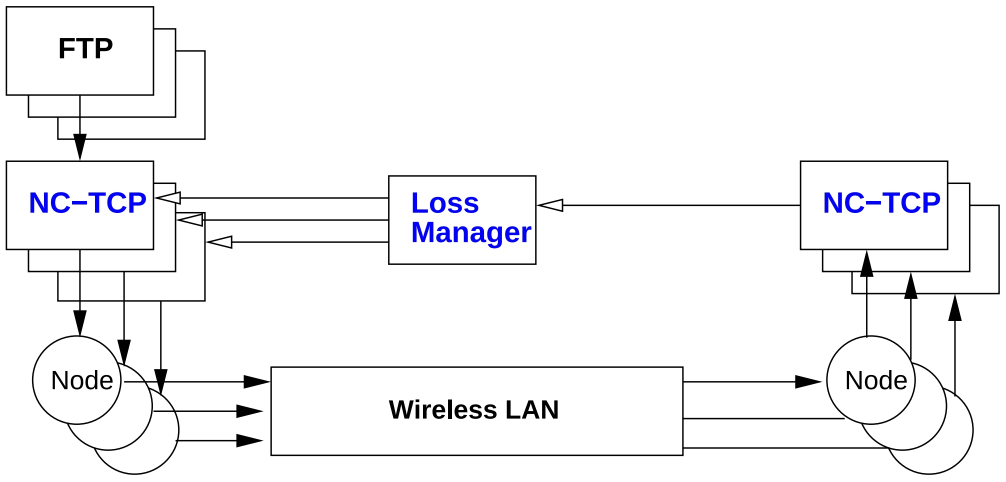
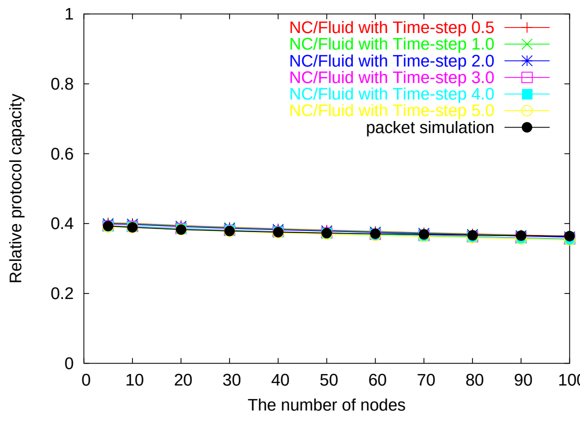
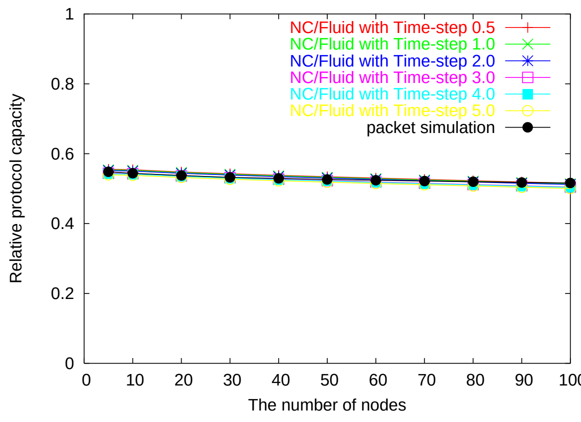


Figure 3.27: Fluid model based simulation for IEEE 802.11-operated WLANs integrated with network calculus based simulation for TCP traffic.



(a) TCP default packet size = 250 bytes (100 slot times)



(b) TCP default packet size = 500 bytes (200 slot times)

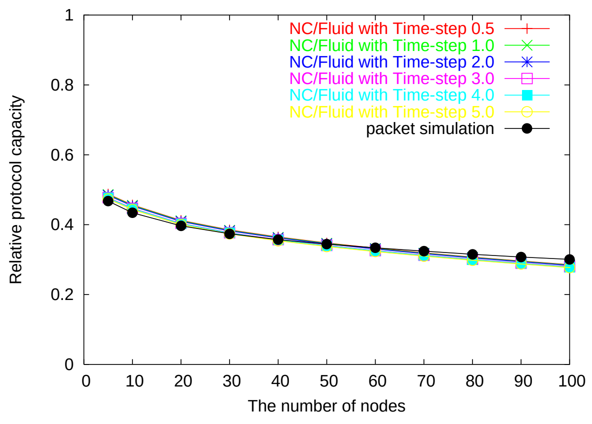
Figure 3.28: Relative protocol capacity as the number of nodes increases to 100 in the case that the RTS/CTS mechanism is used. The packet size is 250 bytes (100 slot times) in (a) and 500 bytes (200 slot times) in (b), respectively.

Performance in Terms of relative errors

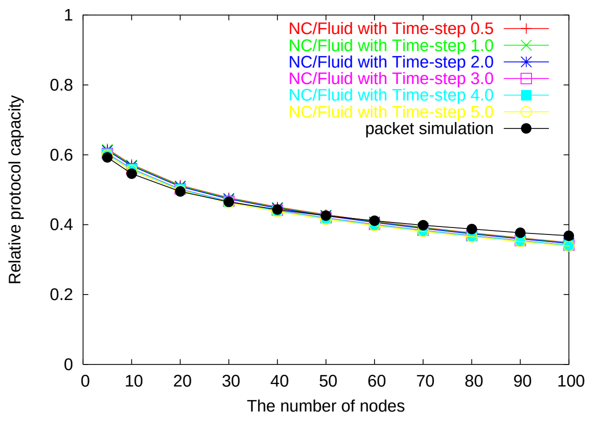
Figures 3.28–3.29 give the relative protocol capacity versus the number of nodes in the cases with and without the RTS/CTS mechanisms, respectively. The TCP default packet size is fixed at 250 (100 slot times) in (a) and 500 bytes (200 slot times) in (b) respectively. In all the cases, the discrepancy in the throughput capacity under the two simulation modes falls within about $2 \sim 4$ %. (Network calculus based simulation causes 2 % additional error due to its approximated round trip time computation.)

Performance in Terms of execution time

Figures 3.30–3.31 give the execution time versus the number of nodes, in both the cases with and without the RTS/CTS mechanism. The packet size is fixed with 250 bytes (100 slot times) in (a) and 500 bytes (200 slot times) in (b)), respectively. Approximately two orders of magnitude improvement can be observed, and the improvement is especially pronounced when the number of nodes increases or the packet size decreases. Note that the observed performance improvement is the combined effect of fluid model based simulation for data transmission activities in wireless LANs and network calculus based simulation for TCP traffic.

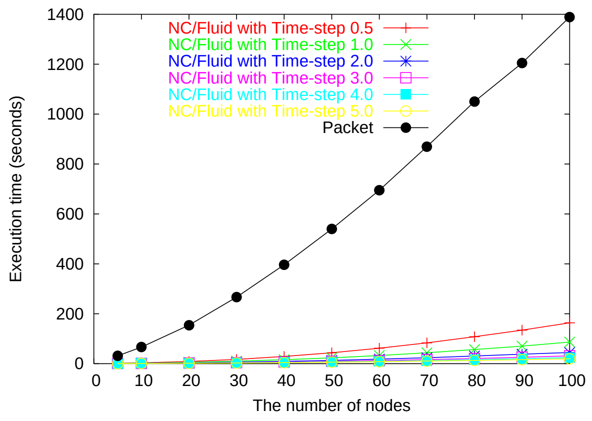


(a) TCP default packet size
= 250 bytes (100 slot times)

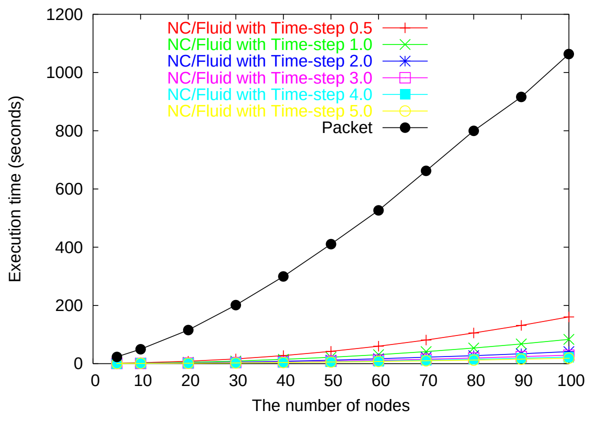


(b) TCP default packet size
= 500 bytes (200 slot times)

Figure 3.29: Relative protocol capacity as the number of nodes increases to 100 in the case that the RTS/CTS mechanism is not used. The packet size is 250 bytes (100 slot times) in (a) and 500 bytes (200 slot times) in (b), respectively.

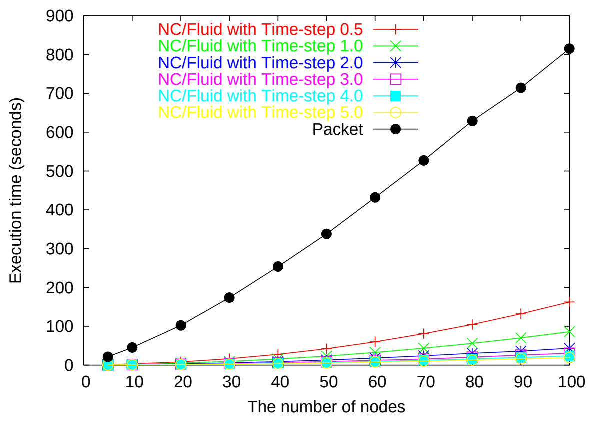


(a) TCP default packet size
= 250 bytes (100 slot times)

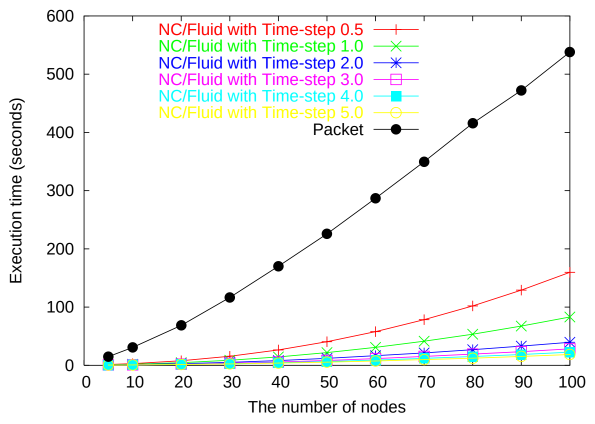


(b) TCP default packet size
= 500 bytes (200 slot times)

Figure 3.30: Execution time versus the number of nodes when the packet size is 250 (100) and 500 bytes (200 slot times) respectively, (the RTS/CTS mechanism is used).



(a) TCP default packet size = 250 bytes (100 slot times)



(b) TCP default packet size = 500 bytes (200 slot times)

Figure 3.31: Execution time versus the number of nodes when the packet size is 250 (100) and 500 bytes (200 slot times) respectively, (the RTS/CTS mechanism is not used).

3.5.3 Discussion

The simulation study indicates that the proposed simulation framework is quite effective in studying the transmission activities of IEEE 802.11-operated WLANs. Under several cases, it improves the performance by about two orders of magnitude as compared to packet level simulation, irrespective of TCP or UDP traffic. The performance improvement is even more pronounced as the number of nodes, the number of applications per node, and/or the packet generation rate per application increases. As a matter of fact, packet level simulation in *ns-2* throws an out-of-memory exception when the number of nodes in a WLAN increases to more than 1000. The relative errors incurred in the framework with the use of the time stepping technique, nevertheless, fall within 2 %, as long as time step value is appropriately determined. We also observe that the ripple effect usually found in fluid model based simulation is not significant in the proposed framework. This is because (i) all the simulation configurations have only one point of interaction — the fluid wireless channel — among wireless nodes; (ii) as the simulation is conducted in the time-stepped manner, the ripple effect is less pronounced than [38, 47].

We claim that the proposed simulation framework can be applied to other network protocols/entities. In our configuration, each protocol entity is considered as a network component, and the wireless channel is considered as a network component of interaction among different network flows. The latter component is, in general, applicable to network queues, multiplexers, switches, or routers. As a result, we believe that the advantages and drawbacks of fluid model based simulation that we observe in IEEE 802.11-operated WLANs can be generalized to other networks, as long as the the analytical model for the point of interaction in the network can be accurately derived.

Chapter 4

Network Calculus Based Simulation for TCP Networks

In this chapter, we examine the feasibility of incorporating network calculus based models in simulating TCP/IP networks. By exploiting network calculus properties, we characterize how TCP congestion control, additive increase and multiplicative decrease (AIMD) and *slow start*, together with the queue management strategy used in routers, regulates TCP flows. We first divide the time axis into intervals, each of which consists of multiple round-trip times, and derive a TCP throughput model which derives the attainable throughput of a TCP flow, given the number of losses in an interval. Then based on the derived throughput model, we define a set of network calculus based theorems that give upper and lower bounds on the attainable TCP throughput in each interval. Finally, we implement network calculus (NC) based simulation in *ns-2*, conduct simulation in both the packet mode and the network calculus based mode, and measure the performance gain in terms of the execution time thus reduced and the error discrepancy in terms of the discrepancy between the network calculus based simulation results and packet-level simulation results.

4.1 Network Calculus and Its Notations

Before delving into the derivation, we first introduce several important operations and notations that pertain to our work. In the network calculus theory a data flow is usually described by means of the cumulative function $f(t)$ or $g(t)$, defined as the number of units (bits or packets) seen on the flow in the time interval $[0, t]$. These functions belong to F, the set of wide-sense increasing sequences or functions f taking values in $R^+ \cup \infty$ [9, 13]. The following operations are defined as basic operations under min-plus algebra.

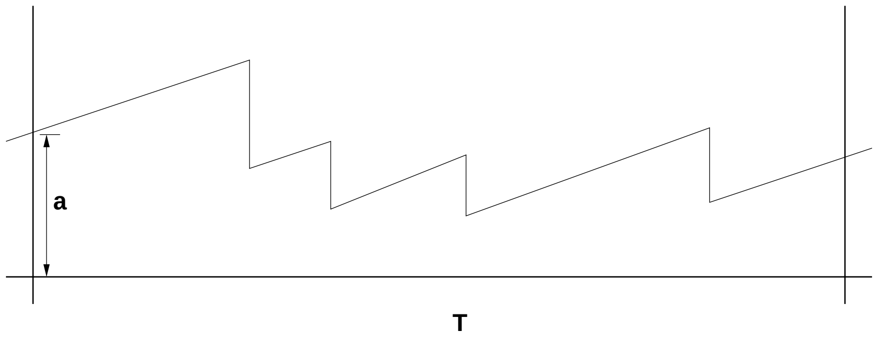
- $\wedge$ represents the minimum operation, $f \wedge g = \min[f, g]$;
- $\vee$ represents the maximum operation, $f \vee g = \max[f, g]$;
- $\otimes$ represents the min-plus convolution of two functions or sequences $f, g \in F$, $(f \otimes g)(t) = \inf_{0 \leq s \leq t}[f(t-s) + g(s)]$;
- $\overline{f}$, for $f \in F$ represents the sub-additive closure of f, which is $\overline{f} = \inf_{n \geq 0}\left[f^{(n)}\right]$.

To extend the min-plus algebra to the time varying case, the family of bivariate functions is introduced: $\tilde{F} = \{F(\cdot,\cdot) : F(s,t) \geq 0, F(s,t) \leq F(s,t+1), \text{ for all } 0 \leq s \leq t\}$, and if $F(\cdot,\cdot) \in \tilde{F}$, $F(t,t) = 0$ for all t. Also, the following functions are defined for $\tilde{F}$:

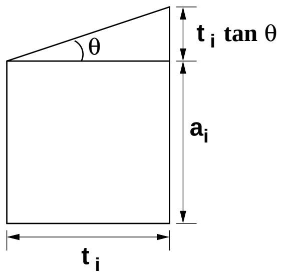
- $(F \wedge G)(s,t) = \min[F(s,t), G(s,t)]$;
- $(F \vee G)(s,t) = \max[F(s,t), G(s,t)]$;
- $(F \otimes G)(s,t) = \inf_{s \leq \tau \leq t}\{F(s,\tau) + G(\tau,t)\}$;
- $\overline{F}(s,t)$ represents the sub-additive closure of $F(s,t)$, $\overline{F}(s,t) = \inf_S \sum_{i=1}^{m}[F(t_{i-1}, t_i)]$, where $S = t_0, t_1, t_2, \ldots, t_m$ is any subset of {1,2,. . . ,t} with $t_0 = s < t_1 < t_2 < \ldots < t_m = t$.

4.2 Analytic Model

In this section, we derive two throughput models: the throughput model and the network calculus model. In the throughput model, we derive the throughput attained in a time interval, T, under the TCP AIMD *congestion avoidance* mechanism and the throughput attained until the time instant t under TCP *slow start* mechanism. The interval T will be used as the value of the time step in the simulation. Then, based on the derived throughput model, we develop the network calculus model that gives upper and lower bounds on the attainable TCP throughput in each interval. Finally, we derive the other functions, i.e., the queue, loss, and output functions needed to describe network behaviors corresponding to TCP interactions, and moreover to realize network calculus based simulation.



(a) TCP behavior during one time step



(b) One AI phase

Figure 4.1: A typical histogram of the attainable throughput under the TCP AIMD mechanism in an interval T, where a is the initial rate in the given period, and t_{R_i} is the round trip time.

4.2.1 Throughput Model

We denote, respectively, the additive increase parameter and the multiplicative decrease parameter under TCP AIMD as α and β. We make the following assumptions in the derivation: (i) all the routers in the network employ FIFO (drop tail) as their buffer occupancy discipline, and (ii) the round trip time, t_{R_i}, *within an interval* T is constant. (Note that we do not make the assumption that t_{R_i} is constant *throughout* the simulation.)

Throughput Model for TCP AIMD

We intend to capture the throughput dynamics under the TCP AIMD mechanism in an interval T. Let the number of packet losses in the interval T be $m-1$, and the initial window size be a. Figure 4.1 depicts a typical histogram of the attainable throughput under TCP AIMD in an interval T ((a)) and in an AI phase ((b)). Under the assumption that the round trip time is constant in each interval T, the slope in each AI phase in the interval T is equal to $\tan\theta_T = \alpha\frac{1}{t_{R_i}^2}$, by Kelly's deterministic differential equation, $\dot{r} = \alpha\frac{1}{t_{R_i}^2} - \beta \cdot r(t - t_{R_i}) \cdot p(r(t - t_{R_i}))$, that characterizes the TCP throughput dynamics (where $p(t)$ is the packet loss rate function [27]).

Given $m-1$ packet losses in an interval T, we have m AI phases within that interval, and the total attainable throughput can be expressed as

$$
\begin{aligned}
S_{T,m-1} = \sum_{i=1}^{m} s_i &= a \cdot \left(t_1 + \beta t_2 + \beta^2 t_3 + \ldots + \beta^{m-1} t_m\right) \\
&+ \tan\theta_T \cdot t_1 \cdot \left(\frac{1}{2} t_1 + \beta t_2 + \ldots + \beta^{m-1} t_m\right) \\
&+ \cdots \\
&+ \tan\theta_T \cdot t_m \cdot \left(\frac{1}{2} t_m\right),
\end{aligned}
\tag{4.1}
$$

where $t_i = c_i \cdot t_{R_i}$, $c_i \geq 1$ and $c_i \in Z$, for $1 \leq i \leq m$, and s_i stands for the throughput of each AI phase within T.

Eq. (4.1) can be recast in the vector and matrix notation as

$$
S_{T,m-1} = \tan\theta_T \cdot t^T \cdot Q \cdot t + a \cdot P^T \cdot t, \tag{4.2}
$$
[1]

where

$$
\begin{aligned}
t &= \begin{pmatrix} t_1 & t_2 & t_3 & \ldots & t_{m-1} & t_m \end{pmatrix}^T, \\
Q &= \begin{pmatrix}
\frac{1}{2} & \beta & \beta^2 & \ldots & \beta^{m-2} & \beta^{m-1} \\
0 & \frac{1}{2} & \beta & \beta^2 & \ldots & \beta^{m-2} \\
\ldots & & \ldots & & & \ldots \\
0 & 0 & 0 & \ldots & \frac{1}{2} & \beta \\
0 & 0 & 0 & \ldots & 0 & \frac{1}{2}
\end{pmatrix}, \\
P &= \begin{pmatrix} 1 & \beta & \beta^2 & \ldots & \beta^{m-2} & \beta^{m-1} \end{pmatrix}^T.
\end{aligned}
$$

Minimum throughput: We derive the minimum and maximum attainable throughput for a tagged TCP flow when the interval, T, and, the number of losses, $m-1$, are given. We formulate

[1] The superscript symbol T represents transposition.

the problem of deriving the minimum attainable throughput as follows:

$$\begin{aligned} \min \qquad & S_{T,m} = \tan\theta_T \cdot t^T \cdot Q \cdot t + a \cdot P^T \cdot t \\ \text{subject to} & \sum_{i=1}^{m} t_i = T. \end{aligned} \tag{4.3}$$

In order to solve the problem, we first investigate (with the use of the Lagrange multiplier theory [5]) whether or not a minimum solution exists, and if so, we can use one of the Lagrange multiplier algorithms [5] to obtain the solution. We formulate the Lagrangian function L as follows:

$$L(t, \lambda) = S_{T,m}(t) + \lambda h(t), \tag{4.4}$$

where $h(t) = \sum_{i=1}^{m} t_i - T = 0$.

The first and second derivatives of Eq. (4.4) are, respectively,

$$\begin{aligned} \bigtriangledown_t L(t,\lambda) &= \tan\theta_T \cdot Q' \cdot t + a \cdot P + \lambda, && (4.5) \\ \bigtriangledown_\lambda L(t,\lambda) &= \sum_{i=1}^{m} t_i - T, && (4.6) \\ \bigtriangledown_{tt}^2 L(t,\lambda) &= \tan\theta_T \cdot Q', && (4.7) \end{aligned}$$

where

$$Q' = \begin{pmatrix} 1 & \beta & \beta^2 & \dots & \beta^{m-2} & \beta^{m-1} \\ \beta & 1 & \beta & \beta^2 & \dots & \beta^{m-2} \\ \beta^2 & \beta & 1 & \beta & \dots & \beta^{m-3} \\ \dots & \beta^2 & \beta & \dots & \beta & \dots \\ \beta^{m-2} & \beta^{m-3} & \dots & \beta & 1 & \beta \\ \beta^{m-1} & \beta^{m-2} & \beta^{m-3} & \dots & \beta & 1 \end{pmatrix}. \tag{4.8}$$

As the matrix Q' is positive definite since $\det(Q') > 0$ and $\tan\theta_T > 0$ since $0 < \theta_T < 90^o$, there exists a minimum solution of S_T, subject to $\sum_{i=1} t_i = T$.

Maximum throughput: Similarly we formulate the problem of deriving the maximum attainable throughput as

$$\begin{aligned} \min \qquad & S^{'}_{T,m-1} = -S_{T,m-1} = -(\tan\theta_T \cdot t^T \cdot Q \cdot t + a \cdot P^T \cdot t) \\ \text{subject to} & \sum_{i=1}^{m} t_i = T. \end{aligned} \tag{4.9}$$

Following the same line of derivation as in the minimization problem, we define the Lagrangian function L' for this maximization problem as follows:

$$L'(t,\lambda) = -(\tan\theta_T \cdot t^T \cdot Q \cdot t + a \cdot P^T \cdot t) + \lambda h(t), \tag{4.10}$$

where $h(t) = \sum_{i=1}^{m} t_i - T = 0$.

The first and second derivatives of Eq. (4.10) are, respectively,

$$\begin{aligned} \bigtriangledown_t L'(t,\lambda) &= -\big(\tan\theta_T \cdot Q' \cdot t + a \cdot P\big) + \lambda, & (4.11)\\ \bigtriangledown_\lambda L'(t,\lambda) &= T - \sum_{i=1}^{m} t_i, & (4.12)\\ \bigtriangledown^2_{tt} L'(t,\lambda) &= -\tan\theta_T \cdot Q', & (4.13) \end{aligned}$$

where Q' is given in Eq. (4.8).

Since the second derivative, $-\tan\theta_T \cdot Q$, of the Lagrangian function in Eq. (4.10) does not satisfy the second order condition for optimality in Lagrange multiplier theory, we cannot draw the conclusion of whether or not there exists a solution for this problem. However, by Weistrass' Theorem [5], we know that the solution exists, and can be obtained when all the losses occur at the beginning or ending point of the interval. Figure 4.2 depicts such a scenario. That is, we can calculate the maximum attainable throughput in a given interval T as

$$S_T^{max} = \begin{cases} \big(a + \frac{1}{2}\tan\theta_T \cdot T\big) \cdot T, & \text{if } T > 0; \\ 0, & \text{if } T \leq 0. \end{cases} \tag{4.14}$$

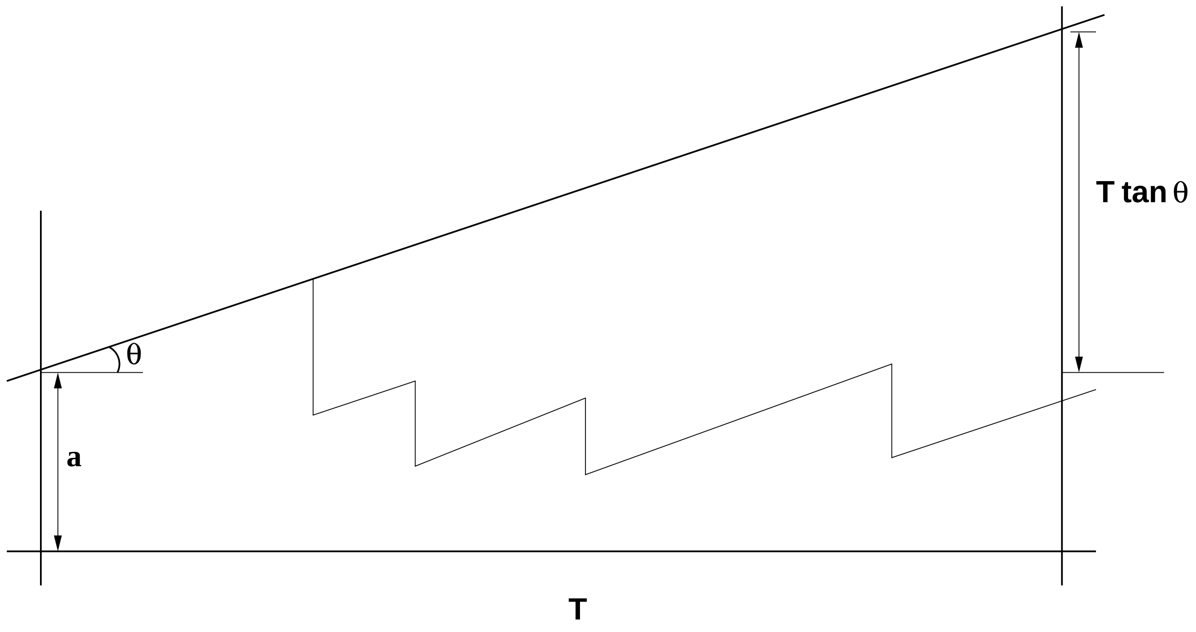


Figure 4.2: The maximum attainable throughput in an interval of T.

Throughput Model for TCP Slow Start Phase

In the TCP *slow start* phase, a TCP flow seeks a maximally attainable throughput by exponentially increasing the TCP congestion window (and hence the sending rate) until it encounters packet loss(es). The instantaneous rate at time t in the *slow start* phase, $r_{ss}(t)$, can be expressed as:

$$r_{ss}(t) = 2^{(t-1)} \tag{4.15}$$

and the cumulative throughout till the time instant t is

$$S_{ss} = 1 + 2 + 2^2 + \cdots + 2^{t-1} = \sum_{i=1}^{t} 2^{(t-1)} = 2^t - 1. \tag{4.16}$$

Note that we divide the time domain with each unit equal to one round trip time in any interval T.

4.2.2 Network Calculus Model

Based on the above throughput models, we now derive the network calculus models that will be used to determine the TCP throughput in network simulation. Let R, X, Y, and L denote, respectively,

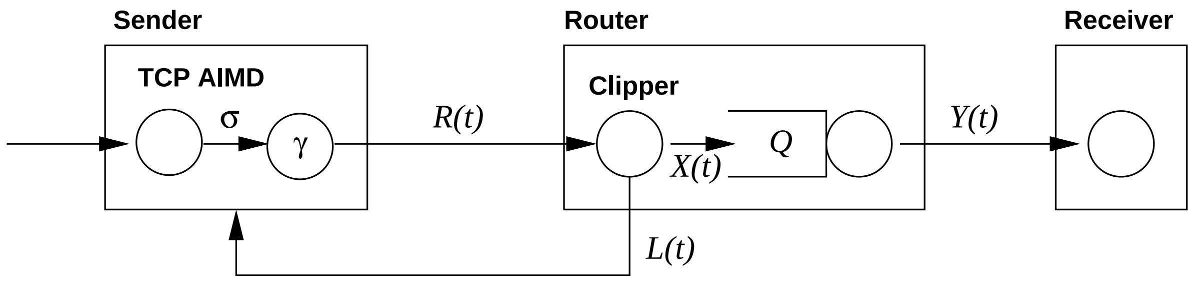


Figure 4.3: The network configuration in the network calculus domain.

the cumulative amount of the various types of traffic as labeled in Figure 4.3. For example, $R(t)$ denotes the amount of traffic sent by the TCP connection in $[0, t]$. These functions belong to the set, F, of wide-sense increasing sequences or functions f taking values in $R^+ \cup \infty$ (Section 4.1). Note that we take t in the discrete time domain in this study ($t \in Z$), and consider that the time axis is divided into intervals in the discrete time domain, $T_i = [t_i, t_{i+1}]$, $i \geq 0$, each of which in turn consists of multiple round-trip times of equal length. We also define

$$R_i(t) = R(t_i + t) - R(t_i), \text{ when } t \in T_i = [t_i, t_{i+1}]\,. \tag{4.17}$$

Network Calculus Model for TCP AIMD

We first prove that TCP AIMD is a piecewise linear time varying shaper in the following theorems and corollary. The piecewise linear property in the shaper function means that the function is defined by concatenating several linear functions without overlapping, and the time varying property means that given a function of two time variables $H(s, t)$, the shaper forces the output $R(t)$ to satisfy the condition

$$R(t) \leq H(s, t) + R(s)$$

for all $s \leq t$.

Theorem 2: TCP AIMD is a piecewise linear time varying shaper with the time varying shaping curve σ_{T_i} in the interval $T_i = [t_i, t_{i+1}]$, where

$$\begin{aligned}
\sigma_{T_i}(s,t) &= \sigma_{T_i}(t) - \sigma_{T_i}(s), \\
\sigma_{T_i}(t) &= \left(a_i + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t\right) t, \\
\text{and } a_i &= r(t_i) = R(t_i) - R(t_i - 1).
\end{aligned} \tag{4.18}$$

Proof: First we know that the function $\sigma_{T_i}(t)$ is convex and piecewise linear since t takes discrete time. Hence we can connect different successive points $\left(t, (a_i + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}t)t\right)$ for all $t = 0, 1, 2, 3, \ldots$, and the resulting curve will be a concatenation of segments, whose slopes are equal to $\frac{1}{2}\frac{\alpha}{t_{R_i}^2}(2t+1) + a_i$ for each interval $[t, t+1]$ when $t \geq 1$ but the initial slope in the interval $[0,1]$ is $a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}$.

Let $R_i(t) = \sum_{i=1}^{t} r_i(t)$ be the cumulative throughput in the interval $[1,t]$, and $\hat{R}_i(s,t) = R_i(t) - R_i(s)$ be the cumulative throughput in the interval $[s+1, t]$, and $R_i(t) = 0$, for $t \leq 0$. Then we have the following relation:

$$\hat{R}_i(s,t) = R_i(t) - R_i(s) = \sum_{k=s+1}^{t} r_k(t),$$
$$\text{for all } 0 \leq s \leq t \leq (t_{i+1} - t_i).$$

From the previous analysis for maximum attainable throughput, we know that the throughput is upper-constrained by σ_{T_i} in the interval $T_i = [t_i, t_{i+1}]$ (Figure 4.2 and Eq. (4.14)). Therefore, we have

$$\begin{aligned}
R_i(t) - R_i(s) &= \sum_{k=s+1}^{t} r_k(t) \\
&\leq \sigma_{T_i}(s,t) = \left(a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t\right) t - \left(a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} s\right) s.
\end{aligned}$$

Finally we have

$$\begin{aligned} R_i(t) &\leq R_i(s) + \sigma_{T_i}(s,t), \\ R_i(t) &\leq \min_{0 \leq s \leq t} \left[R_i(s) + \sigma_{T_i}(s,t) \right]. \end{aligned} \tag{4.19}$$

By Eq. (4.19), we conclude that TCP AIMD is a piecewise linear time varying shaper with the time varying shaping curve σ_{T_i} in the interval T_i. ■

Theorem 3: TCP AIMD is upper constrained by a linear function $C(t)$ in any interval $T_i = [t_i, t_{i+1}]$, where C represents a given constant output capacity:

$$\begin{aligned} R_i(t) &\leq R_i(s) + C(s,t), \\ R_i(t) &\leq \min_{0 \leq s \leq t} \left[R_i(s) + C \cdot (t-s) \right]. \end{aligned} \tag{4.20}$$

Proof: The proof is similar to that for Theorem 2, and is then omitted. ■

Theorem 4: TCP AIMD is upper constrained by a linear function $W(t)$ in any interval $T_i = [t_i, t_{i+1}]$, where W represents a given constant maximum advertised window [2]:

$$\begin{aligned} R_i(t) &\leq R_i(s) + W(s,t), \\ R_i(t) &\leq \min_{0 \leq s \leq t} \left[R_i(s) + W \cdot (t-s) \right]. \end{aligned} \tag{4.21}$$

Proof: The proof is similar to that for Theorem 2, and is then omitted. ■

Corollary 1: The maximum solution of Eqs. (4.19), (4.20), and (4.21), $R_i^{max}(t)$, is

$$\begin{aligned} R_i^{max}(t) &= (\overline{\sigma}_{T_i} \otimes (C \wedge W)(t), \qquad (4.22) \\ \text{where } \overline{\sigma}_{T_i}(t) &= a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t. \end{aligned}$$

[2]This constraint is only for the comparison with packet level simulation (*ns-2*), in which the amount that each TCP can send is upper-bounded by W even if it can send the more number of packets.

Proof: We note that the initial condition $R_i(t) = 0$, for $t \leq 0$. Also, following [9] we define the $\delta(t)$ function as

$$\delta_\tau = \begin{cases} \infty, & \text{if } t > \tau; \\ 0, & \text{if } t \leq \tau. \end{cases}$$

Therefore, $R_i(t)$ is constrained by

$$R_i(t) \leq (\sigma_{T_i} \otimes R_i)(t) \wedge (C \otimes R_i)(t) \wedge (W \otimes R_i)(t) \wedge \delta_0(t).$$

By Theorem 4.3.1 in [9] or Theorem 4.1.5 in [13],[3], we know that the maximal solution of the above equation is

$$R_i^{max}(t) = (\overline{\sigma}_{T_i} \otimes (C \wedge W))(t), \text{ where } \overline{\sigma}_{T_i}(t) = \min_{n \geq 0} \left[\sigma_{T_i}^{(n)} \right].$$

Note that the min-plus convolution, $\otimes$, of two convex functions is also convex. Furthermore, by the property stated/proved in the section entitled *properties of* $\otimes$ *for convex function* in [9], if the two functions are convex and piecewise linear, the min-plus convolution can be obtained by sorting the different linear pieces of the two functions in the order of increasing slopes. Now σ_{T_i} is convex and piecewise linear since it has a linear slope, $\frac{1}{2}\frac{\alpha}{t_{R_i}^2}(2t+1) + a$, in each interval $[t, t+1]$ for $t \geq 1$, except the initial slope in the interval $[0, 1]$, which is $a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}$. By using the aforementioned property, $\sigma_{T_i} \otimes \sigma_{T_i}$ can be obtained by doubling the length of different linear segments of σ_{T_i}, and sorting them end-to-end in the increasing order of their slopes. Note that the first linear segment whose slope $a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}$ has a double length. If we repeat this operation for convolution n times, we obtain a convex piecewise linear sequence $\sigma_{T_i}^{(n)}(t)$:

$$\sigma_{T_i}^{(n)}(t) = a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t \qquad \text{if} \quad 0 \leq t \leq n.$$

Conclusively, the sub-additive closure of σ_{T_i} is determined when $n \to \infty$, and therefore,

$$\overline{\sigma}_{T_i}(t) = a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t.$$

■

[3]The proof is given in [1]

In what follows we prove in the following Lemma and Theorem that TCP AIMD is a system that offers a time varying service curve γ_{T_i} in interval T_i. The piecewise linear property in the service curve means that the curve is defined by concatenating several linear functions without overlapping, and the time varying property means that given a function of two time variables $H(s,t)$, the curve serves the output $R(t)$ to satisfy the condition

$$R(t) \geq H(s,t) + R(s)$$

for all $s \leq t$.

Lemma 1: The function $\gamma_T(t_i, t_j)$ is non-negative and wide-sense increasing in $t \in T_i = [t_i, t_{i+1}]$, where

$$\begin{aligned} \gamma_{T_i}(t_i, t_j) &= \gamma_{T_i}(t_j) - \gamma_{T_i}(t_i), & (4.23) \\ \text{where } \gamma_{T_i}(t) &= \min S_{t,m}, \ \text{when } t \in T_i. & (4.24) \end{aligned}$$

Proof: According to [9] and [13], if $\gamma_{T_i}(t_i, t_j) \geq 0$ and $\gamma_{T_i}(t_i, t_j) \leq \gamma_{T_i}(t_i, t_j + 1)$ for all $0 \leq t_i \leq t_j$, $\gamma_{T_i}(t_i, t_j)$ is a nonnegative and increasing in t. We know that $\gamma_{T_i}(t_i, t_j) \geq 0$ since $\min S_{(t_j - t_i),m}$ is always greater than or equal to 0 when $m \geq 0$, and $(t_j - t_i) > 0$. In addition, since $\gamma_{T_i}(t_i, t_j + 1) = \gamma_{T_i}(t_i, t_j) + \gamma_{T_i}(t_j, t_j + 1)$ and $\gamma_{T_i}(t_j, t_j + 1) = \min S_{1,\tilde{m}}$, for $\tilde{m} \geq 0$, is always non-negative, we know that $\gamma(t_i, t_j + 1) \geq \gamma(t_i, t_j)$. Hence $\gamma(t_i, t_j)$ is non-negative and increasing in t. ■

Theorem 5: When the number of packet losses, $m - 1$, in an interval $T_i = [t_i, t_{i+1}]$ is known, TCP AIMD is a system that offers a time varying service curve γ_{T_i} which is piecewise linear in the interval T_i, where $\gamma_{T_i}(t) = \min S_{t,m}$, when $t \in T_i$.

Proof: First let t^* be the solution of minarg $S_{T_i,m}$. Given the number of packet losses, we can separate T_i into m AI phases $T_{i,j} = [t_{i,j-1}, t_{i,j}]$ for $1 \leq j \leq m$, each of which has the throughput function

$$\begin{aligned} \gamma_{T_{i,j}}(t) &= \left(a_{i,j} + \frac{1}{2} \frac{\alpha}{t_{R_i}^2} t \right) t, \\ \text{where } a_{i,j} &= r(t_{i,j-1}) \text{ and } t_{i,j-1} \leq t \leq t_{i,j}. \end{aligned}$$

We know that each function $\gamma_{T_{i,j}}$ is convex and piecewise linear, and since the sum of convex functions is convex, $\gamma_{T_i}(t)$ is convex and piecewise linear. The remaining derivation is similar to that in Theorems 2, 3, and 4, and finally we have:

$$\begin{aligned} R_i(t) &\geq R_i(s) + \gamma_{T_i}(s,t), \\ R_i(t) &\geq \min_{0 \leq s \leq t} \left[R_i(s) + \gamma_{T_i}(s,t) \right]. \end{aligned} \tag{4.25}$$

■

Corollary 2: The minimum solution of Eq. (4.25), $R_i^{min}(t)$, is

$$R_i^{min}(t) = \overline{\gamma}_{T_i}(t) = \sum_{j=1}^{m} \left\{ a_{j-1} + \frac{1}{2} \frac{\alpha}{t_{R_i}^2} (t_{i,j} - t_{i,j-1}) \right\}, \tag{4.26}$$

where all the terms are defined in Theorem 5.

Proof: By Eq. (4.25), $R_i(t)$ is constrained by

$$R_i^{min}(t) = \left(\gamma_{T_i} \otimes R_i \right)(t).$$

From Theorem 4.3.1 in [9] or Theorem 4.1.5 in [13], the solution of the above equation is

$$\overline{\gamma}_{T_i}(t) = \min_{n \geq 0} \left[\gamma_{T_i}^{(n)} \right].$$

Since $T_i = [t_i, t_{i+1}]$ consists of non-overlapped m sub-intervals, where $T_{i,j} = [t_{i,j-1}, t_{i,j}]$ for $1 \leq j \leq m$, has the throughput function

$$\gamma_{T_{i,j}}(t) = \left(a_{i,j-1} + \frac{1}{2} \frac{1}{t_{R_i}^2} t \right) t, \tag{4.27}$$

we know that $R_i(t)$ is served with the time varying service curve given by (4.27) in the sub-interval $T_{i,j}$. Let $x(\tau) \overset{\triangle}{=} R_i(t_{i,j-1} + \tau) - R_i(t_{i,j-1})$. Then,

$$x(\tau) \geq (\gamma_{T_{i,j}} \otimes x)(\tau) = \min_{0 \leq \omega \leq \tau} \left[\gamma_{T_{i,j}}(\omega, \tau) + x(\omega) \right].$$

By employing the same reasoning as in the proof of Corollary 1, the solution of the above equation is

$$x^{min}(\tau) = \overline{\gamma}_{T_{i,j}}(\tau), \text{ where } \overline{\gamma}_{T_{i,j}}(t) = a_{i,j-1} + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}t.$$

By recasting the above two equations with the original notation, we obtain respectively

$$\begin{aligned}
&R_i(t_{i,j-1}+\tau) - R_i(t_{i,j-1}) \geq \min_{0\leq\omega\leq\tau}\left[\gamma_{T_{i,j}}(\omega,\tau) + R_i(t_{i,j-1}+\omega) - R_i(t_{i,j-1})\right], \\
&R_i^{min}(t_{i,j-1}+\tau) - R_i^{min}(t_{i,j-1}) = \overline{\gamma}_{T_{i,j}}(\tau).
\end{aligned}$$

Then, we have

$$\begin{aligned}
R_i(t) &\geq \min_{t_{i,j-1}\leq s\leq t}\left[\gamma_{T_{i,j}}(s,t) + R_i(s)\right], \\
R_i^{min}(t) &= \overline{\gamma}_{T_{i,j}}(t) + R_i^{min}(t_{i,j-1}), \text{ when } t_{i,j-1}\leq t\leq t_{i,j}.
\end{aligned}$$

By repeating this procedure for each sub-interval, we obtain the minimum solution, $R_i^{min}(t)$:

$$R_i^{min}(t) = \overline{\gamma}_{T_i}(t) = \sum_{j=1}^{m}\overline{\gamma}_{T_{i,j}}(t) = \sum_{j=1}^{m}\left\{a_{j-1} + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}(t_{i,j} - t_{i,j-1})\right\}. \tag{4.28}$$

■

Based on Theorems 2-5 and their subsequent Corollaries or Lemmas, we recast the TCP/IP operated network in the domain of network calculus (Figure 4.3). Then, we can use the cumulative input function $R(t)$ to emulate the TCP AIMD throughput behavior with the time-stepping technique.

The following theorems extend the throughput derivation from within an interval to the entire simulation period.

Theorem 6: When t is in the interval $T_i = [t_i, t_{i+1}]$, $R(t)$ is constrained by the time varying shaping curve σ_{T_i}, and hence $R(t)$ is upper constrained as follows:

$$R(t) \leq \min_{t_i\leq s\leq t}\left[\sigma_{T_i}(s,t) + R(s)\right], \text{ when } t\in T_i. \tag{4.29}$$

Proof: By Eq. (4.19), $R_i(t)$ is upper constrained by a time varying shaping curve $\sigma_{T_i}(t_i, t_{i+1})$ in the interval $T_i = [t_i, t_{i+1}]$. Recall (from Eq. (4.17)) that

$$R_i(\tau) = R(t_i + \tau) - R(t_i). \tag{4.30}$$

Since $R_i(\tau)$ is upper constrained by σ_{T_i},

$$R_i(\tau) \leq \left(\sigma_{T_i} \otimes R_i\right)(\tau) = \min_{0 \leq \omega \leq \tau} \left[\sigma_{T_i}(\omega, \tau) + R_i(\omega)\right].$$

Therefore, by Corollary 1,

$$R_i^{max}(\tau) = (\overline{\sigma}_{T_i} \otimes (C \wedge W))(\tau), \text{ where } \overline{\sigma}_{T_i}(t) = a_i + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t.$$

Recasting the original notation into the above equation, we obtain

$$\begin{aligned} &R(t_i + \tau) - R(t_i) \leq \min_{0 \leq \omega \leq \tau} \left[\sigma_{T_i}(\omega, \tau) + R(t_i + \omega) - R(t_i)\right], \\ &R^{max}(t_i + \tau) - R^{max}(t_i) = (\overline{\sigma}_{T_i} \otimes (C \wedge W))(\tau). \end{aligned}$$

Rearranging the terms, we have

$$\begin{aligned} R(t) &\leq \min_{t_i \leq s \leq t} \left[\sigma_{T_i}(s, t) + R(s)\right], \\ R^{max}(t) &= (\overline{\sigma}_{T_i} \otimes (C \wedge W))(t) + R^{max}(t_i), \text{ when } t \in T_i = [t_i, t_{i+1}]. \end{aligned} \tag{4.31}$$

■

Theorem 7: When the cumulative input function $R(t)$ is in the interval $T_i = [t_i, t_{i+1}]$, it is served by a time varying service curve γ_{T_i}, and hence $R(t)$ is minimally guaranteed as follows:

$$R(t) \geq \min_{t_i \leq s \leq t} \left[\gamma_{T_i}(s, t) + R_i(s)\right], \text{ where } t \in T_i. \tag{4.32}$$

Proof: By Eq. (4.25), $R_i(t)$ is lower bounded by a time varying service curve $\gamma_{T_i}(t_i, t_{i+1})$ in the interval $T_i = [t_i, t_{i+1}]$. Using the expression of $R_i(\tau)$ given in Eq. (4.30), we have

$$R_i(\tau) \geq (\gamma_{T_i} \otimes R_i)(\tau) = \min_{0 \leq \omega \leq \tau} \left[\gamma_{T_i}(\omega, \tau) + R_i(\omega)\right].$$

Following the same line of reasoning as in Theorem 6, we obtain

$$\begin{aligned} R(t) &\geq \min_{t_i \leq s \leq t} \left[\gamma_{T_i}(s, t) + R(s)\right], \\ R^{min}(t) &= \overline{\gamma}_{T_i}(t) + R^{min}(t_i), \text{ when } t \in T_i = [t_i, t_{i+1}]. \end{aligned} \tag{4.33}$$

■

With Theorems 6–7, we determine the maximum value of $R(t)$ using Eq. (4.31) and the minimum value of $R(t)$ using Eq. (4.33) at each time step T. Note that any non-decreasing function $R(t)$ such that $R^{min}(t) \leq R(t) \leq R^{max}(t)$ can be a solution to emulate TCP AIMD throughput in the context of *network calculus*.

Algorithm for determining the amount of TCP traffic sent by a TCP flow: In the simplest form, the amount of TCP traffic sent by a TCP flow in the current time slot $T_i = [t_i, t_{i+1}]$ is $R_j^{max}(t_{i+1}) - R_j^{max}(t_i)$ if no packet loss was incurred in the previous slot; and $R_j^{min}(t_{i+1}) - R_j^{min}(t_i)$ otherwise. To more accurately determine the amount of TCP traffic sent by a TCP flow, we present a refined algorithm that takes into account of activities in the past slots.

Let t_i denote the current time, L_{prev} the number of packet losses in $T_{i-1} = [t_{i-1}, t_i]$, U_{prev} the amount of traffic sent in $T_{i-1} = [t_{i-1}, t_i]$, and U_{succ} the amount of traffic sent in the most recent interval that does not incur packet loss. Then, the amount of traffic, U_{next}, to be sent in $[t_i, t_{i+1}]$ is determined as follows:

```
1. if $L_{prev} = 0$, $U_{next} \leftarrow R_j^{max}(t_{i+1}) - R_j^{max}(t_i)$;
2. else {
3.     if $\left(R_j^{min}(t_{i+1}) - R_j^{min}(t_i) < U_{prev}\ \&\&\ R_j^{min}(t_{i+1}) - R_j^{min}(t_i) < U_{succ} < R_j^{max}(t_{i+1}) - R_j^{max}(t_i)\right)$
4.         $U_{next} \leftarrow U_{succ}$;
5.     else if $\left(R_j^{min}(t_{i+1}) - R_j^{min}(t_i) < U_{prev}\ \&\&\ R_j^{min}(t_{i+1}) - R_j^{min}(t_i) > U_{succ}\right)$
6.             $U_{next} \leftarrow \left(R_j^{min}(t_{i+1}) - R_j^{min}(t_i)\right)$;
7.     else if $\left(R_j^{min}(t_{i+1}) - R_j^{min}(t_i) > U_{succ} > U_{prev}\right)$
8.             $U_{next} \leftarrow U_{prev} - L_{prev}$;
9.     else if $\left(R_j^{min}(t_{i+1}) - R_j^{min}(t_i) > U_{prev} > U_{succ}\right)$
10.            $U_{next} \leftarrow U_{succ}$;
11. }
```

The refined algorithm focuses on the scenarios with incurred losses since losses are feedback from the network to notify the change of network state; it just sends $R_j^{max}(t_{i+1}) - R_j^{max}(t_i)$ if no packet loss was incurred (line 1), but in the other cases it takes the elaborate steps as follows (line 2-11). The refined algorithm basically sends $R_j^{min}(t_{i+1}) - R_j^{min}(t_i)$ if packet loss occurs in the previous time slot (line 5-6). However, it uses U_{succ} (line 3-4) if U_{succ} is in between $R_j^{min}(t_{i+1}) - R_j^{min}(t_i)$ and $R_j^{max}(t_{i+1}) - R_j^{max}(t_i)$ since U_{succ} is one of the solutions that satisfy $R^{min}(t) \leq R(t) \leq R^{max}(t)$. The remaining portion (line 7-10) allows the algorithm to quickly switch from an inappropriate state to a desirable state (which produces the appropriate amount of traffic); if $R_j^{min}(t_{i+1}) - R_j^{min}(t_i)$ is larger than U_{prev} (which caused losses), $R_j^{min}(t_{i+1}) - R_j^{min}(t_i)$ might cause losses again, which means that the current AIMD phase at which the algorithm targets does not agree well with the new network state which results from the instantaneous traffic change. In order to rapidly switch the current AIMD phase to the appropriate AIMD phase, we simply exploit U_{succ}, U_{prev}, and L_{prev} as shown in the algorithm. Note that we can also use the algorithm for TCP slow start phase (which will be explained in the following section) without any modification for this transition and the only intention of the algorithm is to speed it up by discovering the next allowable traffic amount as soon as possible.

The refined algorithm and the rationale behind the algorithm are best illustrated with the example given in Figure 4.4 (which depicts the decision flow which the refined algorithm will take). Note that the y-axis in Figure 4.4 represents the amount of traffic sent in each time slot, but not the cumulative amount of traffic till t. For example, in Figure 4.4, the amount of traffic sent in $[t_1, t_2]$ is determined at time t_1, and is represented by the y-axis value at t_2.

Figure 4.4 (1) shows the basic cases in which the algorithm uses $R_j^{max}(t)$ or $R_j^{min}(t)$ to determine the amount of TCP traffic to be sent. At time t_1, the algorithm determines the amount of traffic according to $R_j^{max}(t_2) - R_j^{max}(t_1)$ since no packet loss is incurred in the previous slot T_0; in time slot T_1, the TCP flow incurs some packet loss and hence at time t_2 the algorithm determines the amount of traffic according to $R_j^{min}(t_3) - R_j^{min}(t_2)$. In Figure 4.4 (2), the TCP flow incurs packet loss in T_3, and hence at time t_4 the algorithm computes the amount of traffic based on $R_j^{min}(t)$. Fortunately, U_{succ}, the amount of traffic successfully sent up to now (the amount sent at in T_2), is greater than $R_j^{min}(t_5) - R_j^{min}(t_4)$ and less than $R_j^{max}(t_5) - R_j^{max}(t_4)$, so that U_{succ} is another solution of satisfying $R^{min}(t) \leq R(t) \leq R^{max}(t)$. As a result, the algorithm uses the amount of traffic successfully sent in T_2 (i.e., the y-axis value at time t_3) as the amount of traffic to be sent

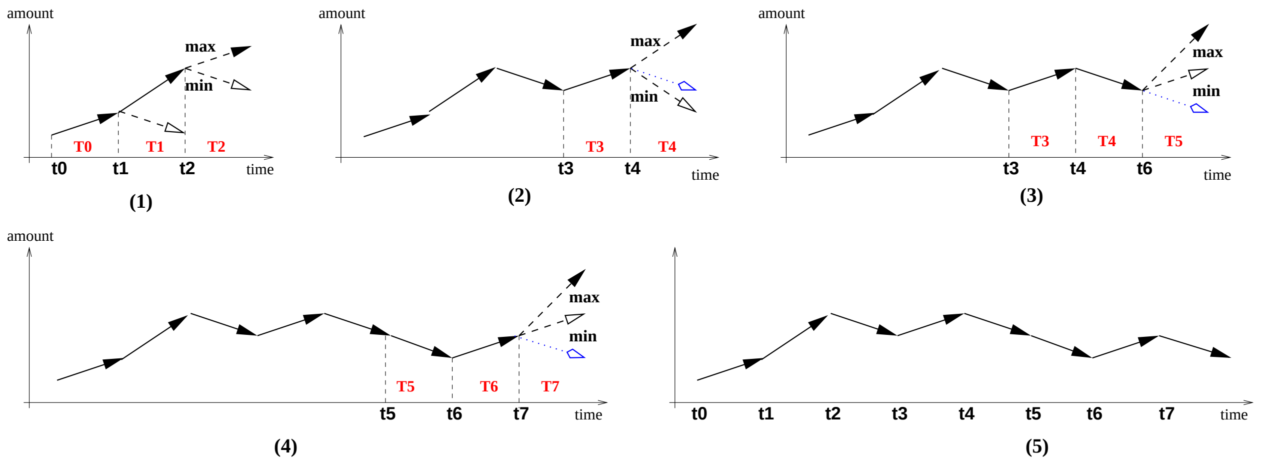


Figure 4.4: The decision flow that the elaborated algorithm takes to determine the amount of TCP traffic sent by a TCP flow in the current slot. The x axes is the time, and the y axes denotes the amount of traffic.

in $[t_4, t_5]$, i.e., the dotted blue line in Figure 4.4 (2). Compared with these previous scenarios, the following scenarios are only for inappropriate cases. In Figure 4.4 (3), the TCP flow also incurs packet loss in T_4, and hence the algorithm determines the amount of traffic to be sent in $T_5 = [t_5, t_6]$ based on $R_j^{min}(t)$. However, the computed result, $R_j^{min}(t_6) - R_j^{min}(t_5)$, is larger than both (i) the amount of traffic sent in $T_4 = [t_4, t_5]$ (U_{prev}), i.e., the y-axis value at time t_5 (that leads to packet loss), and (ii) the amount of traffic successfully sent in $T_3 = [t_3, t_4]$ (U_{succ}), i.e., the y-axis value at time t_4. In this case, the algorithm computes the amount of traffic to be sent in T_5 as the amount of traffic sent in T_4 minus the amount of packet losses in T_4 (line 7-8). The dotted blue line in Figure 4.4 (3) depicts this decision. In Figure 4.4 (4), the TCP flow encounters the same situation as in Figure 4.4 (3), except that the amount of traffic sent in T_6 (U_{prev}) is larger than that successfully sent in T_5 (U_{succ}), and therefore, the algorithm simply uses the amount of traffic successfully transmitted in T_5 as the amount of traffic to be sent in T_7. The blue dotted in Figure 4.4 (4) line depicts this decision. Figure 4.4 (5) shows the complete set of decisions that the algorithm makes for the amount of TCP traffic to be sent.

Note that the above algorithm only uses $R_j^{max}(t)$, $R_j^{min}(t)$, the amount of traffic sent in the previous slot, the amount of packet losses in the previous slot, and the amount successfully sent in the most recent slot to determine the amount of traffic to be sent in the next slot, so that it can rapidly discover the appropriate amount of traffic.

Network Calculus Model for TCP Slow Start Phase

Following the same line of derivation for the network calculus model for TCP AIMD algorithm, we first show that TCP *slow start* is piecewise linear in the following theorems and corollaries. In contrast to the derivation for TCP AIMD, we do not use the time varying property, as the same rate function (Eq. (4.15)) and the same cumulative throughput function (Eq. (4.16)) are used till the end of the *slow start* phase.

Theorem 8: TCP *slow start* is upper constrained by a linear function $C(t)$ in any interval where C represents a given constant output link capacity:

$$\begin{aligned} R_{ss}(t) - R_{ss}(s) &\leq C \cdot (t-s), \\ R_{ss}(t) &\leq \min_{0 \leq s \leq t} \left[R_{ss}(s) + C \cdot (t-s) \right]. \end{aligned} \tag{4.34}$$

Proof: The proof is similar to that for Theorem 2, and is then omitted. ■

Theorem 9: The amount of traffic, $R_{ss}(t)$, sent by a TCP flow in the slow start phase is upper constrained by a linear function $W(t)$ in any interval where W represents a given constant maximum advertised window:

$$\begin{aligned} R_{ss}(t) - R_{ss}(s) &\leq W \cdot (t-s), \\ R_{ss}(t) &\leq \min_{0 \leq s \leq t} \left[R_{ss}(s) + W \cdot (t-s) \right]. \end{aligned} \tag{4.35}$$

Proof: The proof is similar to that for Theorem 2, and is then omitted. ■

Corollary 3: The maximum solution of Eqs. (4.34) and (4.35), $R_{ss}^{max}(t)$, is

$$R_{ss}^{max}(t) = (C \wedge W)(t). \tag{4.36}$$

Proof: $R_s s(t)$ is constrained by

$$R_{ss}(t) \leq (C \otimes R_i)(t) \wedge (W \otimes R_i)(t) \wedge \delta_0(t).$$

By Theorem 4.3.1 in [9] or Theorem 4.1.5 in [13], we know that the maximal solution of the above equation is

$$R_{ss}^{max}(t) = \overline{(C \wedge W)}(t) = (\overline{C} \otimes \overline{W}))(t) = (C \wedge W)(t). \tag{4.37}$$

Note that we do not derive a maximum throughput function, as in Theorem 2. This is because (i) in the *slow start* phase a TCP flow intends to reach an maximally allowable throughput as fast as possible, and hence maximal regulation is not necessary; (ii) since the exponential function (Eqs. (4.15) and (4.16)) that a TCP flow follows to send packets in the *slow start* phase is non-polynomial, for which a corresponding linear function can hardly be defined.

■

In what follows, we show that TCP *slow start* is a system that offers a piecewise linear service curve in any interval.

Theorem 10: TCP *slow start* shapes $R_{ss}(t)$ with a piecewise shaper with the shaping curve $\xi(t)$ in any interval, where

$$\xi(t) = \begin{cases} 2^t - 1, & \text{if } t \geq 1; \\ 0 & t \leq 0. \end{cases} \tag{4.38}$$

Proof:

First we know that the function $\xi(t)$ is convex and piecewise linear as it takes discrete time, and thus we can connect different successive points $(t, 2^t)$ when $t \geq 1$ and $(0,0)$ when $t = 0$, and the resulting curve will be a concatenation of segments, whose slopes are equal to 2^t for each interval $[t, t+1]$ when $t \geq 0$.

Since $R_{ss}(t) = \sum_{i=1}^{t} r_i(t)$ in the interval $[1, t]$, and $R_{ss}(t) = 0$, for $t \leq 0$, we have the following relation:

$$R_{ss}(t) - R_{ss}(s) = \sum_{k=s+1}^{t} r_k(t), \tag{4.39}$$

for all $0 \leq s \leq t$. By integrating $\xi(t)$ into the above equation, we have

$$\begin{aligned} R_{ss}(t) - R_{ss}(s) &= \xi(t) - \xi(s) = (2^t - 1) - (2^s - 1) = 2^s \cdot (2^{(t-s)} - 1) \\ &= 2^s \cdot R_{ss}(t-s) \;\; \geq \;\; R_{ss}(t-s). \end{aligned} \tag{4.40}$$

Finally we have

$$\begin{aligned} R_{ss}(t) &\geq R_{ss}(s) + \xi(t-s) \\ R_{ss}(t) &\geq \min_{0 \leq s \leq t} \left[R_{ss}(s) + \xi(t-s) \right]. \end{aligned} \tag{4.41}$$

By Eq. (4.41), we conclude that a TCP flow in the *slow start* phase is served by a piecewise linear curve ξ. ■

Corollary 4: The minimal solution of Eq. (4.41) is

$$R_{ss}^{min}(t) = \overline{\xi}(t), \tag{4.42}$$

where $\overline{\xi}(t) = t$.

Proof: $R_{ss}(t)$ is served by

$$R_{ss}(t) \geq (\xi \otimes R_{ss})(t). \tag{4.43}$$

Again by Theorem 4.3.1 in [9] or Theorem 4.1.5 in [13], we know that the minimal solution of the above equation is

$$R_{ss}^{min}(t) = \overline{\xi}(t), \text{ where } \overline{\xi}(t) = \min_{n \geq 0} \left[\xi^{(n)} \right]. \tag{4.44}$$

Following a similar line of reasoning as in the proof of Corollary 1, the solution to Eq.(4.44) is

$$\overline{\xi}(t) = t.$$

■

In the *slow start* phase, the algorithm to determining the amount of traffic sent by a TCP flow in the current time slot $T_i = [t_i, t_{i+1}]$ is as follows: the amount of traffic is calculated as $R_{ss}^{min}(t_{i+1}) - R_{ss}^{min}(t_i)$ according to Eq. (4.42), and at the same time whether or not the computed amount exceeds the maximally allowed amount of traffic in Eq. (4.36) is checked.

4.2.3 Queue, Loss, and Output Function

We also express the output function, the queue length function, and the loss function at each router during $T_i = [t_i, t_{i+1}]$. Let N be the total number of flows routed to the same output link, l, at each router. Then, the cumulative output function, the queue length function, and the cumulative amount of packet loss at the output link, l, are respectively determined as follows:

$$\begin{aligned} Y(t_{i+1}) &= Y(t_i) + \min(\sum_{j=1}^{N} R^j(t_i, t_{i+1}), C_l(t_{i+1} - t_i)), & (4.45) \\ q(t_{i+1}) &= \min(Q, [q(t_i) + R(t_i, t_{i+1}) - C_l(t_i, t_{i+1})]^+), & (4.46) \\ L(t_{i+1}) &= L(t_i) + [q(t_i) + R(t_i, t_{i+1}) - C_l(t_i, t_{i+1}) - Q]^+ \,, & (4.47) \end{aligned}$$

where in Eq. (4.45) we use superscript to denote a flow among multiple flows, and C_l represents the capacity of the output link.

Each TCP flow "contributes to" the cumulative amount of output, the packet loss and the queue length according to the proportion of its flow amount to the total amount of flows. Additionally, we have to consider the round trip time per flow since flows with smaller round trip times contribute to a larger portion of the output, loss, and queue functions, as compared with flows with larger round trip times. Based on this observation, a router divides flows into several classes according to their round trip time, and then distribute the amount of output, packet loss, and queue length to each class in the ascending order of round trip times until the amount of $(Y(t_{i+1}) - Y(t_i))$ is consumed.

Let $N'(\leq N)$ be the number of classes, and $Class(k), 1 \leq k \leq N'$, the kth class. The amount of traffic that the kth class can send during $T_i = [t_i, t_{i+1}]$ is:

$$Y^{Class(k)}(t_{i+1}) = Y^{Class(k)}(t_i) + (Y(t_{i+1}) - Y(t_i)) \cdot \frac{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}{\sum_{j=1}^{N} R^j(t_i, t_{i+1})}. \quad (4.48)$$

The queue length that the kth class contributes is also determined as:

$$q^{Class(k)}(t_{i+1}) = q(t_{i+1}) \cdot \frac{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}{\sum_{j=1}^{N} R^j(t_i, t_{i+1})}. \quad (4.49)$$

Within each class, we determine the output function, the packet loss function, and the queue length function for each flow j as follows. First, the output function for flow j is

$$Y^j(t_{i+1}) = Y^j(t_i) + \left(Y^{Class(k)}(t_{i+1}) - Y^{Class(k)}(t_i)\right) \cdot \frac{R^j(t_i, t_{i+1})}{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}; \quad (4.50)$$

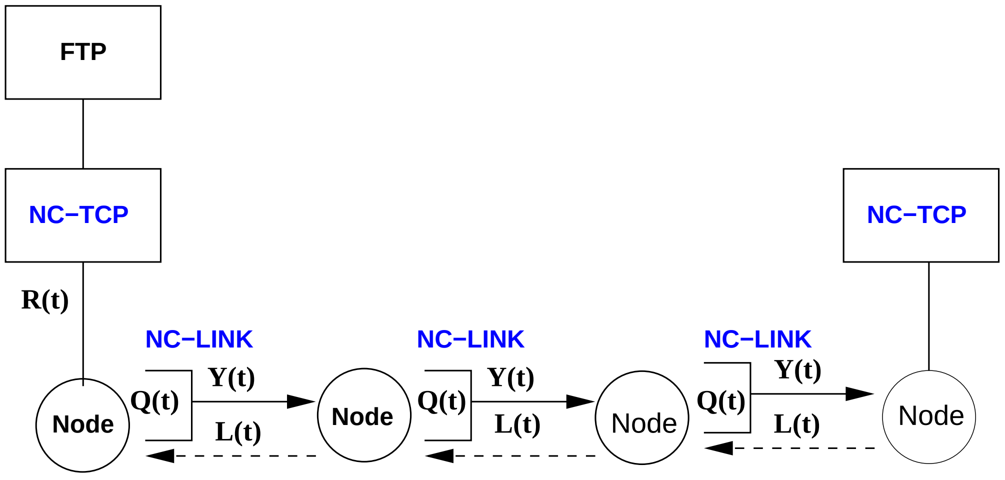


Figure 4.5: Network components implemented for the network calculus based simulation.

Second, the queue length function for flow j is

$$q^j(t_{i+1}) = q^j(t_i) + q^{Class(k)}(t_{i+1}) \cdot \frac{R^j(t_i, t_{i+1})}{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}; \tag{4.51}$$

and finally the loss function for flow j is

$$L^j(t_{i+1}) = L^j(t_i) + \left[R^j(t_i, t_{i+1}) - \left(Y^j(t_{i+1}) - Y^j(t_i)\right) - q^j(t_{i+1})\right]^+ . \tag{4.52}$$

4.3 Simulation Study

In this section, we discuss how we incorporate network calculus models into *ns-2*. Based on the implementation, we also evaluate network calculus based (the network calculus based) simulation for TCP/IP-operated networks by conducting an extensive set of simulations.

4.3.1 Implementation

Based on all the equations derived in Section 4.2, we implement *network calculus* based simulation in *ns-2*. In particular, we create two modules, *NC-TCP* and *NC-LINK*, and a new packet *nc-tcp*. Figure 4.5 gives the software architecture that consists of *NC-TCP*, *NC-LINK*, and existing modules such as FTP module and nodes.

A *NC-TCP* module at the sending side performs the following tasks: it (i) directly supports Theorems 6 7, 8, 9, and 10, (ii) computes a round trip time used for the next time step on the basis of the loss and delay information fed back from the corresponding receiver *NC-TCP* module, (iii) determines the maximum throughput and minimum throughput by Eqs. (4.31) and (4.33) in the *congestion avoidance* phase or by Eqs. (4.36) and (4.42) in the *slow start* phase, and finally (iv) decides the amount of traffic according to the algorithm introduced in Section 4.2.2, and sends the computed amount of traffic via a *nc-tcp* packet at each time step. A *NC-TCP* module at the receiving side simply computes the effective throughput when it receives *nc-tcp* packets, and responds with another *nc-tcp* packet with the loss and delay information collected over the forward path. This acknowledgment is delivered to the sending *NC-TCP* without intervention with *NC-LINK* since the backward path consists of existing *SimpleLink module* in *ns-2*.

NC-LINK modules are used to connect network nodes. They are equipped with drop-tail queue, can process *nc-tcp* packets and determine the output, loss, and queue function using Eqs. (4.50), (4.52), and (4.51), respectively. *NC-TCP* operates in the time-stepped fashion [22] while *NC-LINK* is activated when all the flow information it handles arrives.

4.3.2 Simulation

We have conducted simulation to evaluate the network calculus based simulation and compared its performance (in terms of reduction in execution time and discrepancy between network calculus model based simulation and packet level simulation) against fluid model based simulation as well as packet level simulation, in a wide variety of network topologies. The packet size is set to 1000 bytes, unless otherwise stated. Two sets of experiments were carried out: one employs *TCP Tahoe* and the other uses *TCP Reno* for both packet level simulation and fluid model based simulations. In what follows, we only present results with *TCP Reno*.

All the experiments are conducted in Linux 2.4.18 on a Pentium 4/1.9 *Ghz* PC with 1 *GBytes* memory and with 2 *GBytes* swap memory, and *ns-2.1b9a* is chosen as the underlying simulation environment.

Performance of the network calculus based simulation w.r.t. error discrepancy: First we examine how close the TCP throughput obtained under the network calculus based simulation is to that under packet level simulation. The simulation is carried out in a simple network (Figure 4.6) with the number of FTP nodes varying from 10 to 100. The link bandwidth and delay are labeled in the figure, and the buffer size at each router is 50 packets. Figure 4.7 gives the number of packets delivered every 100 seconds in the system in an ℓ-second simulation under both simulation modes

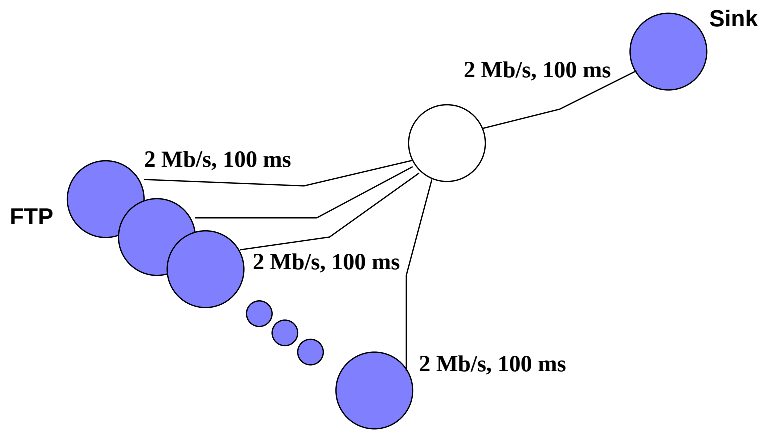
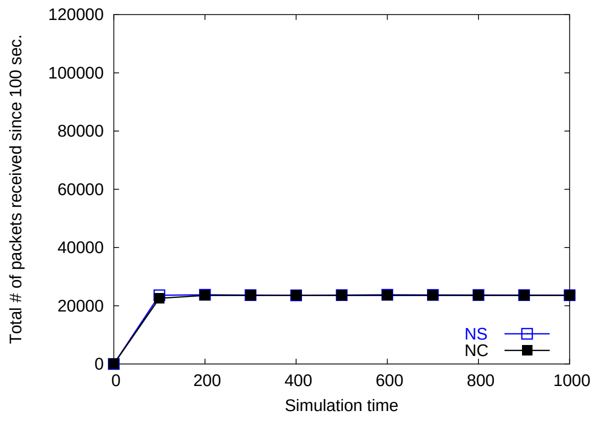


Figure 4.6: The network configuration used in the first set of experiments.

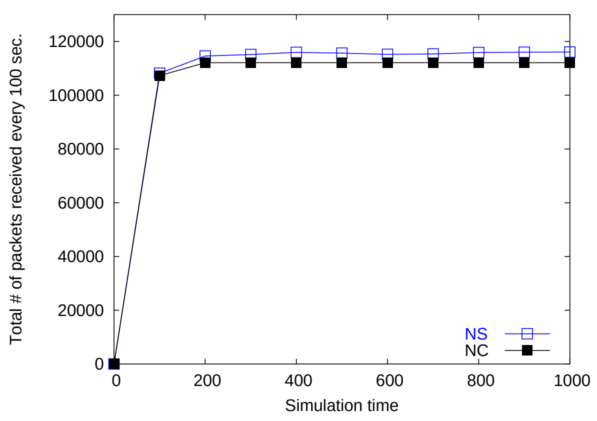
(ℓ varies from 100 to 1000), when the number of nodes is 40 and the link bandwidth of each link is 2 Mb/s ((a)) or 10 Mb/s ((b)). As shown in Figure 4.7, the results generated under the network calculus based simulation are very close to those under packet level simulation. The discrepancy in the results between the two simulation modes is maximally $\sim$10 %.

Even though the error discrepancy in the aggregate throughput maximally reaches to 10 %, the error discrepancy on a per flow basis is more acceptable. Figure 4.8 shows the comparison on a per flow basis between two sets of throughputs: (i) the maximum, the minimum, and the average throughput in the packet level simulation, and (ii) the maximum, the minimum, and the average throughput in the network calculus based simulation. As shown in Figure 4.8, All three throughputs in the network calculus based simulation exist within the throughput range in the packet level simulation, and they are all close to the average value of the packet level simulation. Therefore we can claim that the flows generated in the network calculus based simulation are TCP-friendly.

To verify whether or not the network calculus based simulation still renders high-fidelity simulation results in more complicated network configurations, we repeat the same experiments but in the configurations given in Figures 4.9, 4.12, and 4.15. In Figure 4.9, the number of nodes in each class varies from 10 to 100, and the buffer size at each router is set to 50 packets. Figure 4.10 gives the number of packets delivered every 100 seconds in the system in an ℓ-second simulation under both simulation modes (again ℓ varies from 100 to 1000), when the number of nodes is 40 and the link capacity of bottleneck link is 10 Mb/s ((a)) or 20 Mb/s ((b)). As shown in Figure 4.10, the discrepancy in the results between the two simulation modes is maximally $\sim$10 %.

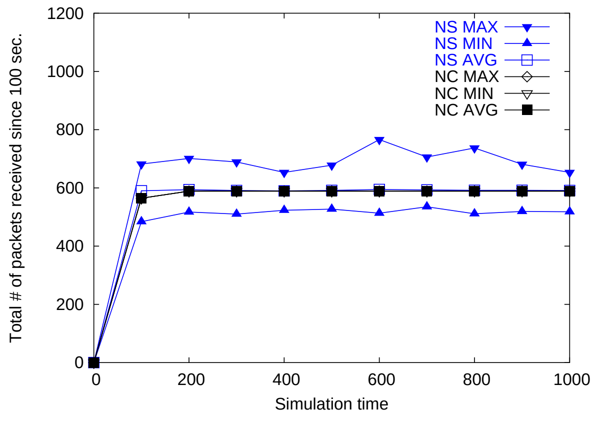


(a) Link bandwidth = 2 Mb/s in Figure 4.6

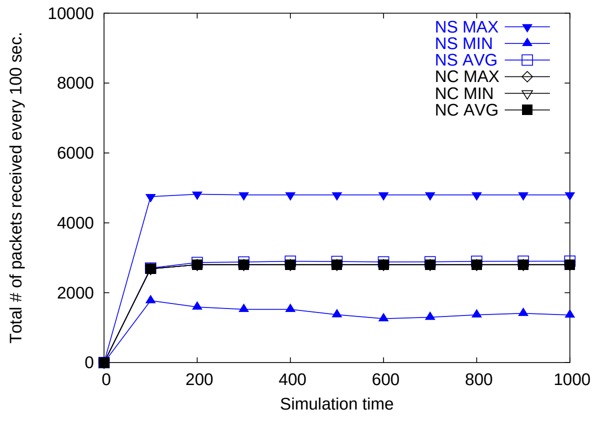


(b) Link bandwidth = 10 Mb/s in Figure 4.6

Figure 4.7: The total number of packets received every 100 seconds in the network calculus based and and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.6, where the number of nodes is 40.



(a) Link bandwidth = 2 Mb/s in Figure 4.6



(b) Link bandwidth = 10 Mb/s in Figure 4.6

Figure 4.8: The maximum, minimum, and the average number of packets received every 100 seconds at each node in the network calculus based and the packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.6, where the number of nodes is 40.

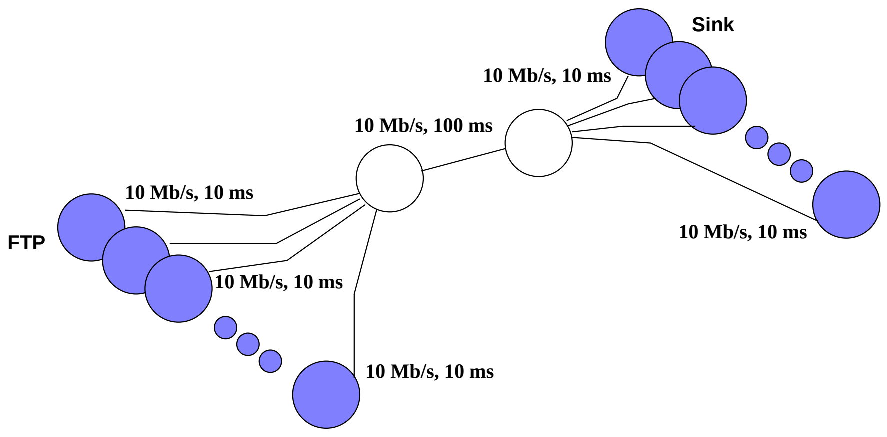
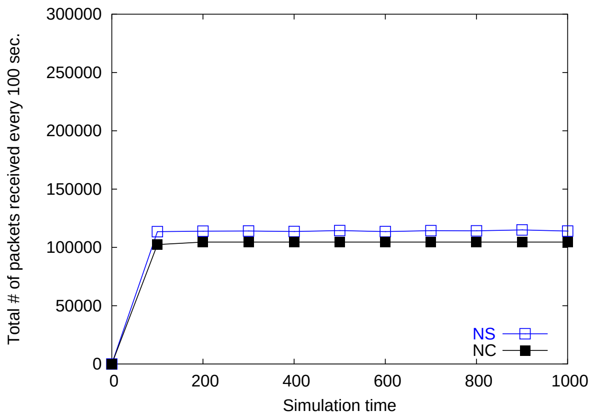
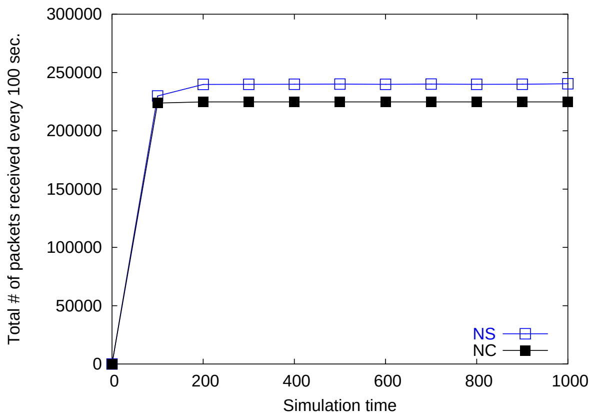


Figure 4.9: The network configuration used in the second set of experiments.

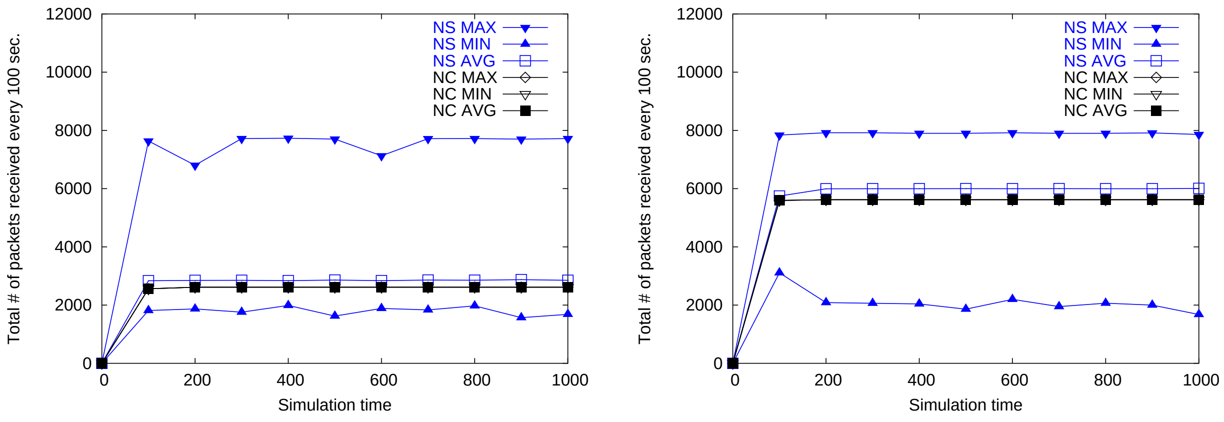


(a) Link bandwidth = 10 Mb/s in Figure 4.9



(b) Link bandwidth = 20 Mb/s in Figure 4.9

Figure 4.10: The total number of packets received every 100 seconds in the network calculus based and and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.9, where the number of nodes is 40.



(a) Link bandwidth = 10 Mb/s in Figure 4.9 (b) Link bandwidth = 20 Mb/s in Figure 4.9

Figure 4.11: The maximum, minimum, and the average number of packets received every 100 seconds at each node in the network calculus based and the packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.9, where the number of nodes is 40.

Figure 4.11 compares the throughputs on a per flow basis between the packet level and the network calculus based simulations. The same observation as done in Figure 4.8 is made: The flows generated in the network calculus based simulation mimic the throughput behaviors of TCP flows.

In Figure 4.12, the number of nodes in each class varies from 10 to 100, and the buffer size at each router is set to 50 packets. Figure 4.13 depicts the number of packets delivered every 100 seconds in each class in an ℓ-second simulation under both simulation modes, when each of the two TCP classes consists of 20 nodes ((a)) and 40 nodes ((b)), respectively. Again the network calculus based simulation renders very close results to packet level simulation. As the number of flows increases, the discrepancy in the results between the two simulation modes also increases and is maximally $\sim$10 %. Figure 4.14 compares the throughputs on a per flow basis between the packet level and the network calculus based simulations. The same observation as done in Figures 4.8 and 4.11 is made.

Figure 4.15 gives another complicated network configuration where the buffer size at each router is 50 packets and the number of nodes in each class varies from 10 to 100. Figure 4.16 gives the corresponding results under both simulation nodes, when each of the three TCP classes consists of 40 nodes ((a)) and 60 nodes ((b)), respectively. As shown in Figure 4.16, a similar trend can be observed, except that the error discrepancy is now maximally $\sim$12 %. Figure 4.17 compares

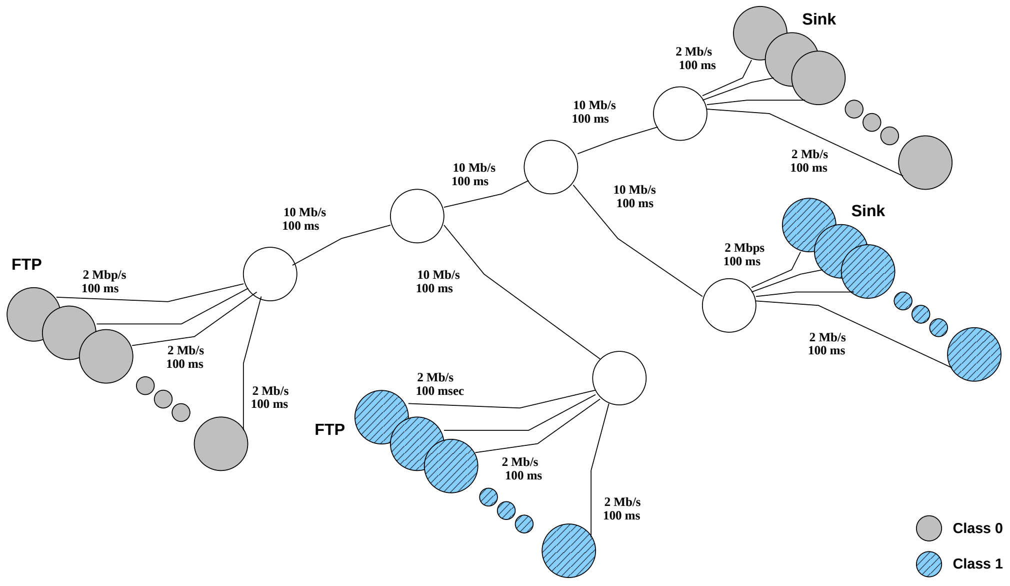
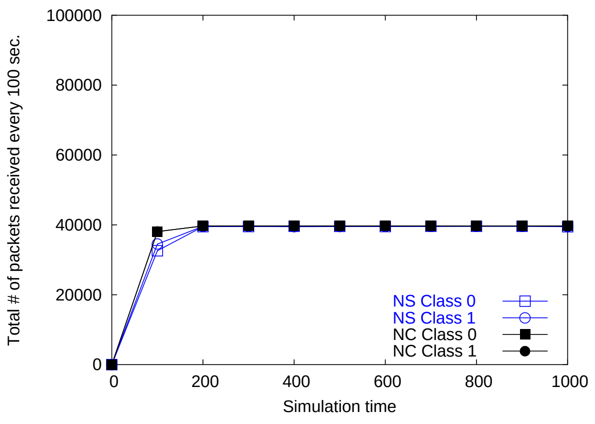
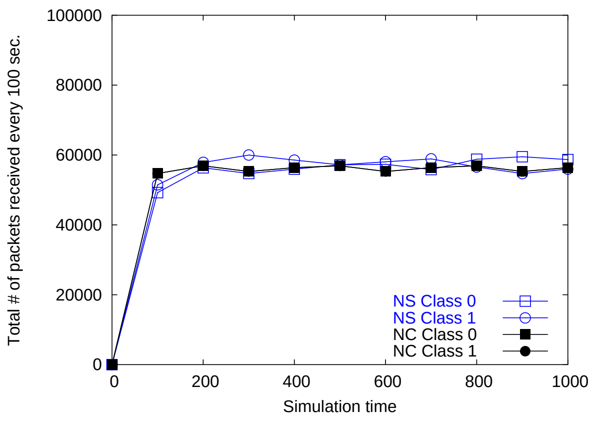


Figure 4.12: The network configuration used in the third set of experiments.

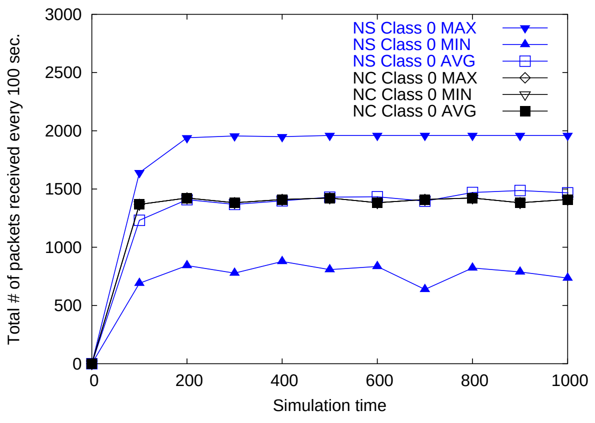


(a) # of nodes in each class = 20 nodes in Figure 4.12

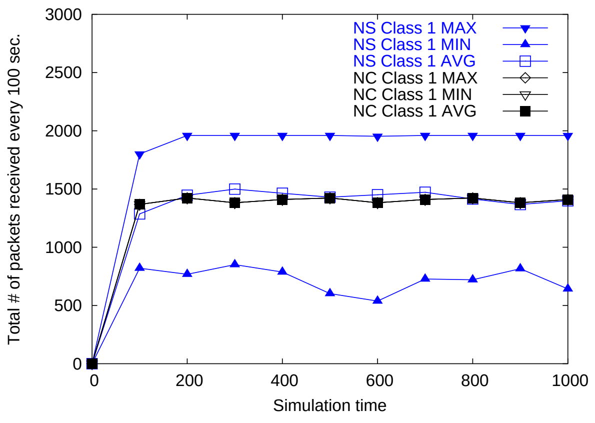


(b) # of nodes in each class = 40 nodes in Figure 4.12

Figure 4.13: The total number of packets received every 100 seconds in the network calculus based and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.12.



(a) # of nodes in class 0 = 40 nodes in Figure 4.12



(b) # of nodes in class 1 = 40 nodes in Figure 4.12

Figure 4.14: The maximum, minimum, and the average number of packets received every 100 seconds at one node in class 0 (a) and one node in class 1 (b) in the network calculus based and the packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.12.

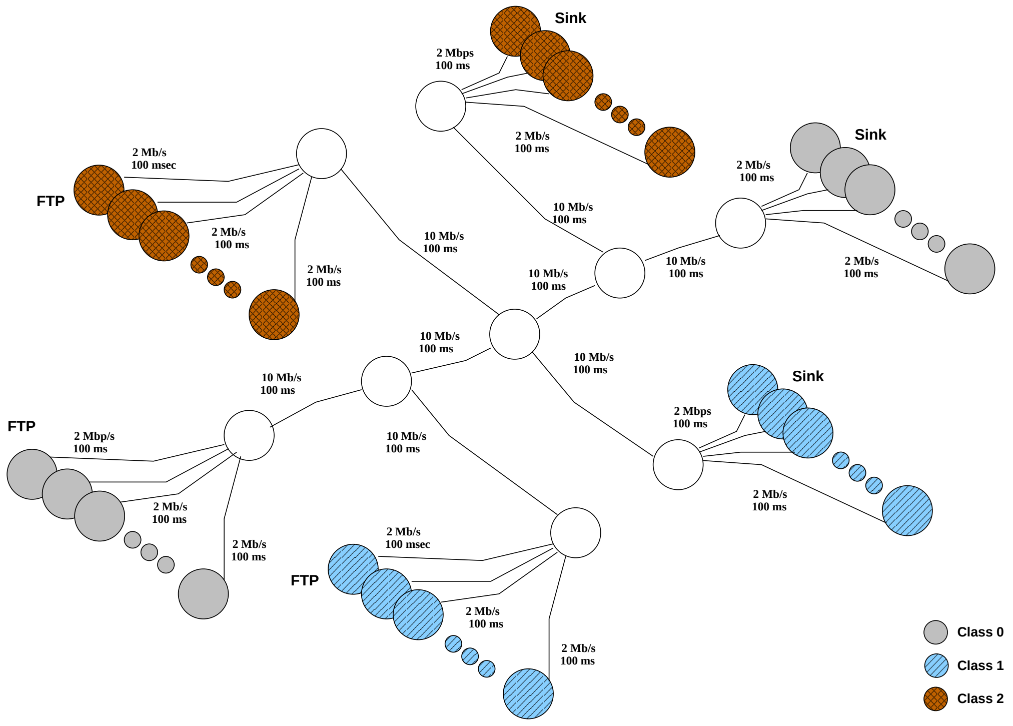
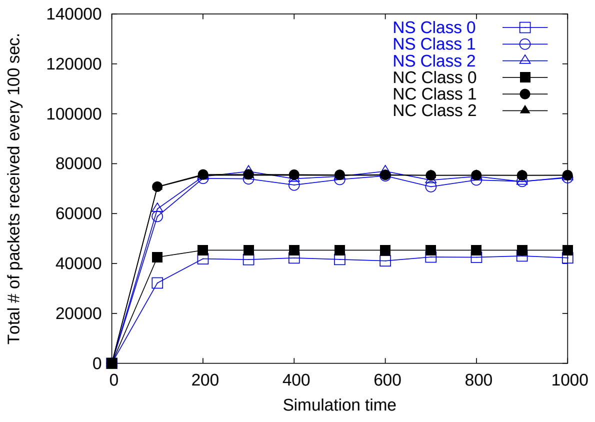
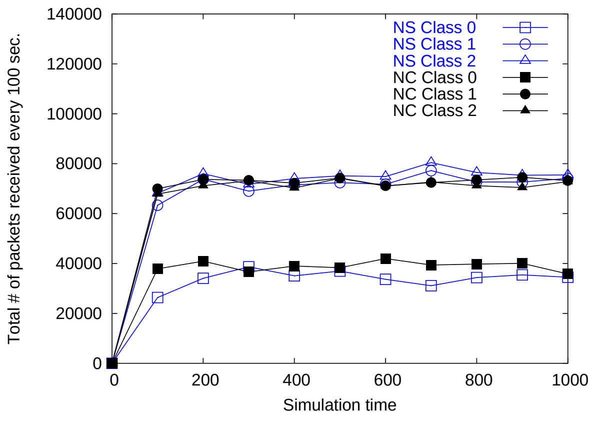


Figure 4.15: The network configuration used in the forth set of experiments.



(a) # of nodes in each class = 40 nodes in Figure 4.12



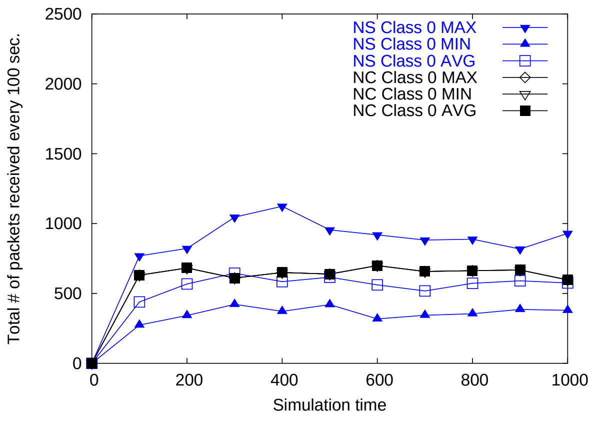
(b) # of nodes in each class = 60 nodes in Figure 4.12

Figure 4.16: The total number of packets received every 100 seconds in the network calculus based and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is in Figure 4.15.

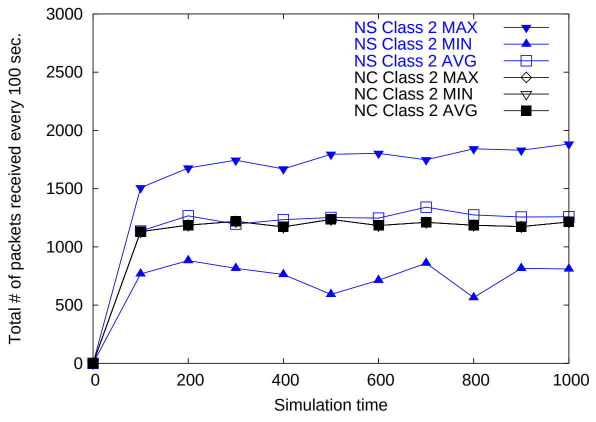
the throughputs on a per flow basis between the packet level and the network calculus based simulations. The same observation as done in the previous cases is made.

We believe the observed error discrepancy in the aggregate throughput across all the nodes results from the facts that (i) the algorithm used to determine the amount of TCP traffic to be sent is not exact and (ii) *network calculus* cannot fully describe the non-linear TCP dynamics with linearity. Note, however, that the observed error discrepancy is more acceptable on a per flow basis.

Performance of the network calculus based simulation w.r.t. execution time: Figure 4.18 (Figure 4.19) gives the execution time it takes to carry out 1000-second simulation versus the number of nodes under packet level and the network calculus based simulation in the network configuration given in Figure 4.6 (Figure 4.9), while Figure 4.20 (Figure 4.21) gives the execution time taken to carry out 1000-second simulation versus the number of nodes per class in the network configuration given in Figure 4.12 (Figure 4.15). As shown in Figures 4.18–4.21, the execution time incurred in the network calculus based simulation is maximally 5-10 times less than that in packet level simulation. (The speed-up can be further improved with a larger time step value.) One interesting result is that the execution time incurred in the network calculus based simulation slightly increases with the number of nodes per class. This results from the time step overhead: even if the number of packets to be sent is small, the number of timing events in the network calculus based simulation increases with the number of nodes in the network configuration, and moreover, such the



(a) # of nodes in class 0 = 60 nodes in Figure 4.12



(b) # of nodes in class 2 = 60 nodes in Figure 4.12

Figure 4.17: The maximum, minimum, and the average number of packets received every 100 seconds at one node in class 0 (a) and at one node in class 1 (b) in the network calculus based and the packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Figure 4.15.

overhead increases as congestion at the bottleneck link increases. We are currently investigating how to alleviate such the timing overhead based on the method of simulating large IP network at the flow level in [3].

Effects of the time step value: To investigate how the time step value T (Figure 4.1), affects the performance in the network calculus based simulation, we carry out simulation that varies the time step value T. Tables 4.1–4.4 give the results with varying time step values for the network configurations given in Figures 4.6, 4.9, 4.12, and 4.15, respectively. In all the configurations, the buffer size at each router is set to 50 packets.

As shown in the tables, the performance of the network calculus based simulation can be improved (up to about 20 times better than that of packet level simulation) as the time step value increases. Note that the error discrepancy does not always increase with the time step value, and hence the execution time cannot be unlimitedly reduced by increasing the time step value. This is due to the facts that in the current the network calculus based simulation implementation: (i) sometimes the error that results from the computation optimization in the matrix computation (Section 4.2) is larger than that incurred in the choice of larger time step values, and dominates the total error; as a result the error does not fully depends on time step values, and (ii) as the number of losses (usually) linearly increases with the increase in the time step value and the performance of

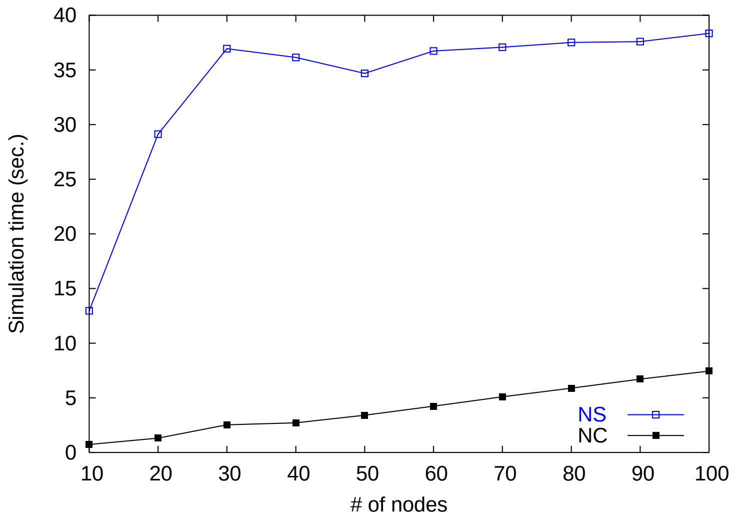


Figure 4.18: The execution time taken to carry out 1000-second simulation versus the number of nodes, in both packet level and the network calculus based simulation in the network configuration in Figure 4.6, when the link bandwidth is *10 Mb/s.*

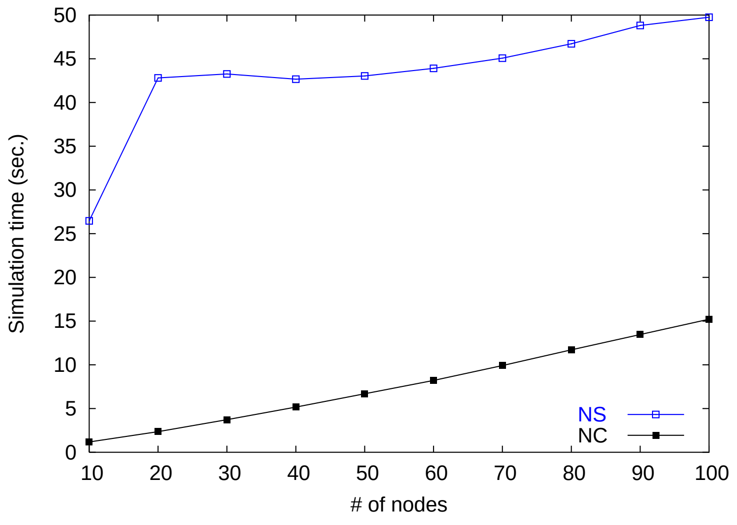


Figure 4.19: The execution time taken to carry out 1000-second simulation versus the number of nodes, in both packet level and the network calculus based simulation in the network configuration in Figure 4.9, when the bottleneck link bandwidth is *10 Mb/s.* and bottleneck link delay is *100 ms*

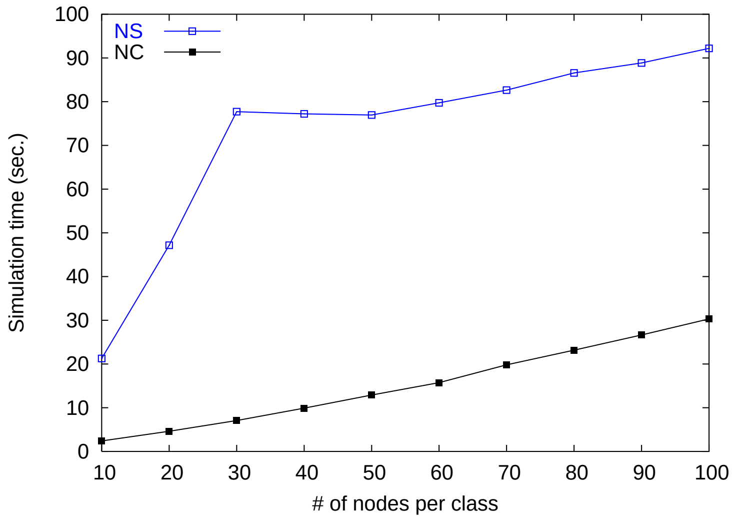


Figure 4.20: The execution time taken to carry out 1000-second simulation versus the number of nodes per class, in both packet level and the network calculus based simulation in the network configuration in Figure 4.12.

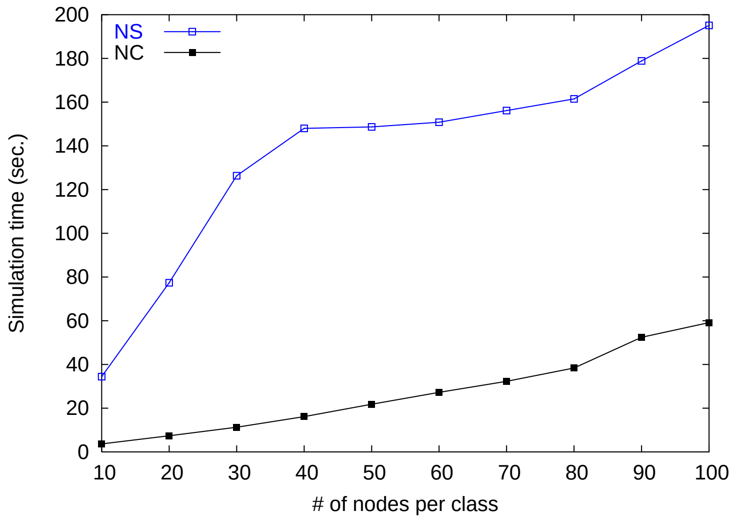


Figure 4.21: The execution time taken to carry out 1000-second simulation versus the number of nodes per class, in both packet level and the network calculus based simulation in the network configuration in Figure 4.15.

Table 4.1: The number of packets received in a 1000-second simulation run and execution time (s) in both packet level and the network calculus based simulation in the network configuration in Figure 4.6, where the link bandwidth, delay, and the number of nodes are *10 Mb/s*, *100 ms.*, and 40, respectively.

Time step	packet level simulation		the network calculus based simulation	
value (s)	total # of packets	time	total # of packets	time
	1148061	37.69		
0.5			1127072	8.93
1.0			1028183	6.15
2.0			1116222	2.63
4.0			1175617	1.80

Table 4.2: The number of packets received in a 1000-second simulation run and execution time (s) in both packet level and the network calculus based simulation in the network configuration in Figure 4.9, where the bottleneck link bandwidth, delay, and the number of nodes are *10 Mb/s*, *100 ms.*, and 40, respectively.

Time step	packet level simulation		the network calculus based simulation	
value (s)	total # of packets	time	total # of packets	time
	1140723	43.07		
0.5			1187571	13.58
1.0			1141306	7.73
2.0			1043305	4.92
4.0			1128313	3.52

computation optimization depends on the number of losses, the optimization algorithm consumes more computation time when the number of losses is large.

Comparison with fluid model based simulation: We also compare the network calculus based simulation against fluid model based simulation. We use the time-stepped hybrid simulation (TSHS) technique [22] to realize fluid model based simulation, since instead of solving a set of differential equations for TCP dynamics, this technique actually sends, receives, drops, and acknowledges packets as other network simulators usually do. In TSHS, packets that are sent by the same session in each time step are grouped into a *chunk*, and all the packets in a chuck are assumed to be evenly spaced within the time step (the process of which is termed as *packet smoothing*). We use Eqs. (4.45), (4.47) and (4.46), to compute the amount of output, the packet loss, and the queue length at each link module (output of router), and when packet losses occur, we distribute them evenly over all flows. We believe this technique has three major advantages in our implementation: (i) we do not randomly drop packets in a sequence of packets delivered in a *chunk* packet, but instead drop packets from the end of the packet sequence at each routing node. This improves TCP

Table 4.3: The number of packets received in a 1000-second simulation run and execution time (s) in both packet level and the network calculus based simulation in the network configuration in Figure 4.12, where the number of nodes is 40.

Time step value (s)	packet level simulation			the network calculus based simulation		
	class 1	class 2	time	class 1	class 2	time
	563605	569206	78.43			
0.5				623860	623860	29.68
1.0				608808	608808	16.43
2.0				524901	524901	9.95
4.0				531605	531605	6.73

Table 4.4: The number of packets received in a 1000-second simulation run and execution time (s) in both packet level and the network calculus based simulation in the network configuration in Figure 4.15, where the number of nodes is 40.

Time step value (s)	packet level simulation				the network calculus based simulation			
	class 1	class 2	class 3	time	class 1	class 2	class 3	time
	410994	718703	735189	148.09				
0.5					391877	787703	792450	43.66
1.0					400430	776574	787850	24.73
2.0					318217	781412	782459	15.34
4.0					447279	712530	733790	11.00

throughput, as it prevents TCP from retransmitting successfully received packets due to packets dropped with smaller sequence numbers; (ii) we apply the time stepping technique only to TCP modules, so that no waiting time is incurred at nodes on the path. This prevents the round trip time from being prolonged as a result of using the time stepping technique at intermediate nodes; and (iii) we do not apply transmission, queuing, and propagation delay to data packets on the forward path, but to ACK packets. This improves accuracy in terms of throughput for fluid model based simulation.

Tables 4.5–4.6 give the results in packet level, the network calculus based, and fluid model based simulation in the network configuration given in Figure 4.6 where the buffer size is 50 packets. The link bandwidth and delay used are, respectively, {*2 Mb/s*, *100 ms.*}, and {*10 Mb/s, 100 ms.*}. Note that we have attempted to tune the time step value to obtain the best results (in terms of error discrepancy and execution time) in both the network calculus based and fluid model based simulation. As compared with fluid model based simulation, the network calculus based simulation not only incurs (slightly) smaller execution time, but also more closely follows the trajectory in packet level simulation. Fluid model based simulation produces good results only when there exist

Table 4.5: The number of packets received in a 1000-second simulation run and execution time (s) in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.6, where the link bandwidth is *2 Mb/s* and the delay is *100 ms.* (Time step value in the network calculus based simulation is *1.0 (s)* while that in fluid model based simulation is *100 (ms).*

# of nodes	packet level simulation		the network calculus based simulation		fluid simulation	
	total # of packets	time	total # of packets	time	total # of packets	time
10	239704	6.08	223046	1.50	138907	3.00
20	238499	6.78	239558	2.71	190110	4.96
40	236746	7.66	224672	4.99	241965	7.96

Table 4.6: The number of packets received in a 1000-second simulation run and execution time (s) in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.6, where the link bandwidth is *10 Mb/s* and the delay is *100 ms.* (Time step in the network calculus based simulation is *3.0 (s)* while that in fluid model based simulation is *100 (ms).*)

# of nodes	packet level simulation		the network calculus based simulation		fluid simulation	
	total # of packets	time	total # of packets	time	total # of packets	time
10	495286	13.13	493356	0.71	154316	3.47
20	988174	30.61	986712	1.13	190811	4.96
40	1148061	37.69	1116758	2.12	408948	12.05

a sufficiently large number of flows to saturate bottle-neck links.

Tables 4.7 gives the results in packet level, the network calculus based, and fluid model based simulation in the network configuration given in Figure 4.9. The same observation can be made; the network calculus based simulation produces similar results to those in packet level simulation, while incurring much less execution time. Table 4.8–4.9 gives the results in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.12 where the buffer size is 50 packets and the link bandwidth is either as specified in the figure (Table 4.8) or set to 10 Mb/s (Table 4.9). Again the network calculus based simulation follows more closely the trajectory in packet level simulation, while incurring much less execution time. Finally tables 4.10 gives the results in packet level, the network calculus based, and fluid model based simulation in the network configuration given in Figure 4.15, from which the same observation can be made.

In all the above experiments, we observe that the performance of fluid model based simulation is sensitive to the time step values. If the time step value is small, fluid model based simulation suffers from the excessive timeout events and cannot reduce the execution time as desired. On the other hand, if the time step value is large (especially when $n\times$ time step $\sim$ RTT), the error discrepancy

Table 4.7: The number of packets received per class in a 1000-second simulation run and execution time (s) in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.9. (Time step in the network calculus based simulation is *0.5 (s)* while that in fluid model based simulation is *100 (ms).*)

# of nodes	packet level simulation		the network calculus based simulation		fluid simulation	
	total # of packets	time	total # of packets	time	total # of packets	time
10	821379	26.78	816172	3.40	241922	5.25
20	1192427	42.91	1032863	8.01	1074644	18.54
40	1140723	43.37	1187571	13.58	1013768	21.60

Table 4.8: The number of packets received per class in a 1000-second simulation run and execution time (s) in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.12. (Time step in the network calculus based simulation is *1.5 (s)* while that in fluid model based simulation is *100 (ms).*)

# of nodes	packet level simulation			network calculus simulation			fluid simulation		
per class	class 0	class 1	time	class 0	class 1	time	class 0	class 1	time
10	194336	196889	21.43	197859	197859	3.24	137528	114057	10.14
20	388598	390500	47.55	395719	395719	6.47	215649	309420	22.93
40	563605	569206	77.34	559290	559290	12.01	425703	537156	48.32

becomes prohibitively larger. When TCP connections of different round trip times co-exist, it becomes a non-trivial task to tune the time-step value. Furthermore, the network calculus based simulation reduce the execution time more significantly (30 times less than packet level simulation) when we choose larger time step values in these experiments.

Current limitation and improvement: As the number of flows (or the number of nodes) in all network configurations increases, the error discrepancy in the network calculus based simulation also increases. The reason is as follows: as the number of flows increases, some flows, especially flows

Table 4.9: The number of packets received per class in a 1000-second simulation run and execution time (s) in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.12 except that all the link bandwidth are set to *10 Mb/s.* (Time step in the network calculus based simulation is *2.0 (s)* while that in fluid model based simulation is *100 (ms).*)

# of nodes	packet level simulation			network calculus simulation			fluid simulation		
per class	class 0	class 1	time	class 0	class 1	time	class 0	class 1	time
10	196081	198492	21.53	198871	198871	2.54	128712	156772	11.44
20	391540	393428	47.71	397742	397742	4.67	215649	309420	23.71
40	534113	596307	75.61	572605	572605	9.96	425703	537156	47.74

Table 4.10: The number of packets received per class in a 1000-second simulation run and execution time (s) in packet level, the network calculus based, and fluid model based simulation in the network configuration in Figure 4.15. (Time step in the network calculus based simulation is *1.0 (s)* while that in fluid model based simulation is *500 (ms).*)

# of nodes	packet level simulation				network calculus simulation				fluid simulation			
per class	class 0	class 1	class 2	time	class 0	class 1	class 2	time	class 0	class 1	class 2	time
10	161766	195872	195872	34.60	165125	197915	197915	6.90	132710	132710	132710	16.56
20	321719	386237	387977	77.52	330250	395830	395830	12.30	233315	242935	262132	33.08
40	410994	718703	735189	148.09	400430	776574	787850	24.73	345097	396085	410049	58.89

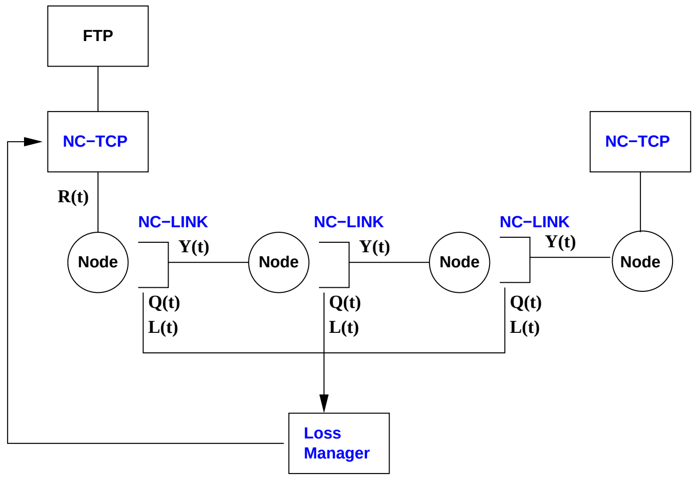


Figure 4.22: A modified architecture for implementing the network calculus based simulation.

with longer round trip time, often cannot enqueue their packets in the buffer in the network and hence cannot send any packet in a time step, T. However, the current buffer model (Section 4.2.3) can serve all flows even in the extremely congested cases. We are currently investigating how to resolve this problem.

Additionally, by modifying the software architecture for implementing the network calculus based simulation from Figure 4.5 to Figure 4.22, we can further reduce the execution time by 10-20 %. This is because instead of feeding back *nc-tcp* packet (acknowledgment) at the receiver side, the *Loss Manager* module collects the loss information from routers and provides the information to the sender. As a result, the execution time is reduced as a result of the reduction in the number of packet events to be processed.

Chapter 5

Mixed Mode Simulation for IEEE 802.11-operated WLANs

As shown already in Chapters 3 and 4, *fluid model based simulation* and *network calculus* based simulation cannot provide packet mode dynamics such as the instantaneous queue length and the packet dropping probability, even though they significantly alleviate the computational overhead in large scale networks by approximating flows with theoretical models. In some sense, such the model based simulations trade some degree of accuracy and packet level dynamics for simulation performance. If packet level details are of concern to users, the best approach seems to simulate foreground traffic (whose packet dynamics are of interest) in the packet mode, and model the background traffic (that comprise of possibly hundreds of flows) with a fluid model. The notion of *mixed mode simulation* (a.k.a. hybrid simulation) was proposed to combine the performance gains of fluid model based simulation with the accuracy afforded by packet mode simulation. Figure 5.1 gives a blueprint on mixed mode simulation. One foreground flow is simulated at the packet mode, while the other background flows are approximated into a collection of fluid chunks and simulated in the theoretical mode. Note that these two types of flows influence each other at the point of interaction, e.g. a routing buffer or a wireless channel.

One important issue of mixed mode simulation is to accurately characterize the interaction between foreground traffic and background flows, e.g., when, and how many, packets of a foreground flow should be dropped due to the existence of background fluid chucks (that may represent a large number of background packets) at a router and vice versa. We deal with this mixed mode simulation in this chapter and the following chapter.

In this chapter, we investigate how to realize mixed mode simulation for IEEE 802.11-operated WLANs, by proposing the model of interaction at the wireless channel between the foreground

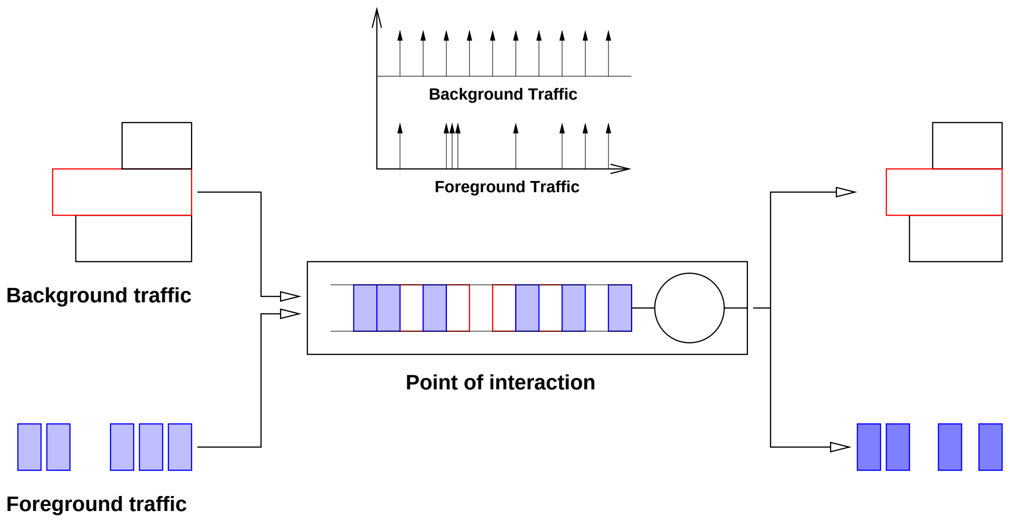


Figure 5.1: Conceptual description for *mixed mode simulation*.

flow and the other background flows, in view of their achievable throughput. This enables packet mode simulation to co-exist and interact with fluid model based simulation within one simulation framework. In order to accomplish the goal, we need two analytical models: one that describes background flows in view of their data transmission activities, and the other that characterizes data transmission of the foreground flow, as well as its interaction with background flows. We then implement mixed mode simulation in *ns-2* [4], and conduct a comprehensive simulation study to evaluate mixed mode simulation with respect to accuracy in terms of error discrepancy and efficiency in terms of speed-up in conducting simulation.

5.1 Throughput Model That Characterize Interactions Between Foreground and Background Traffic

Based on the fluid model [28], we derive the throughput model for one tagged flow so as to characterize the interaction between the tagged flow and the other (background) flows in view of the achievable throughput. In this model, the tagged flow is generated by one or multiple applications. (In the latter case, packets generated by multiple applications are multiplexed into one foreground flow.)

The throughput that a foreground flow can achieve depends on its interaction with the other background flows. Figure 5.2 depicts how flows interacts/affects one another. Let $\overline{T}_{fg}$ denote the

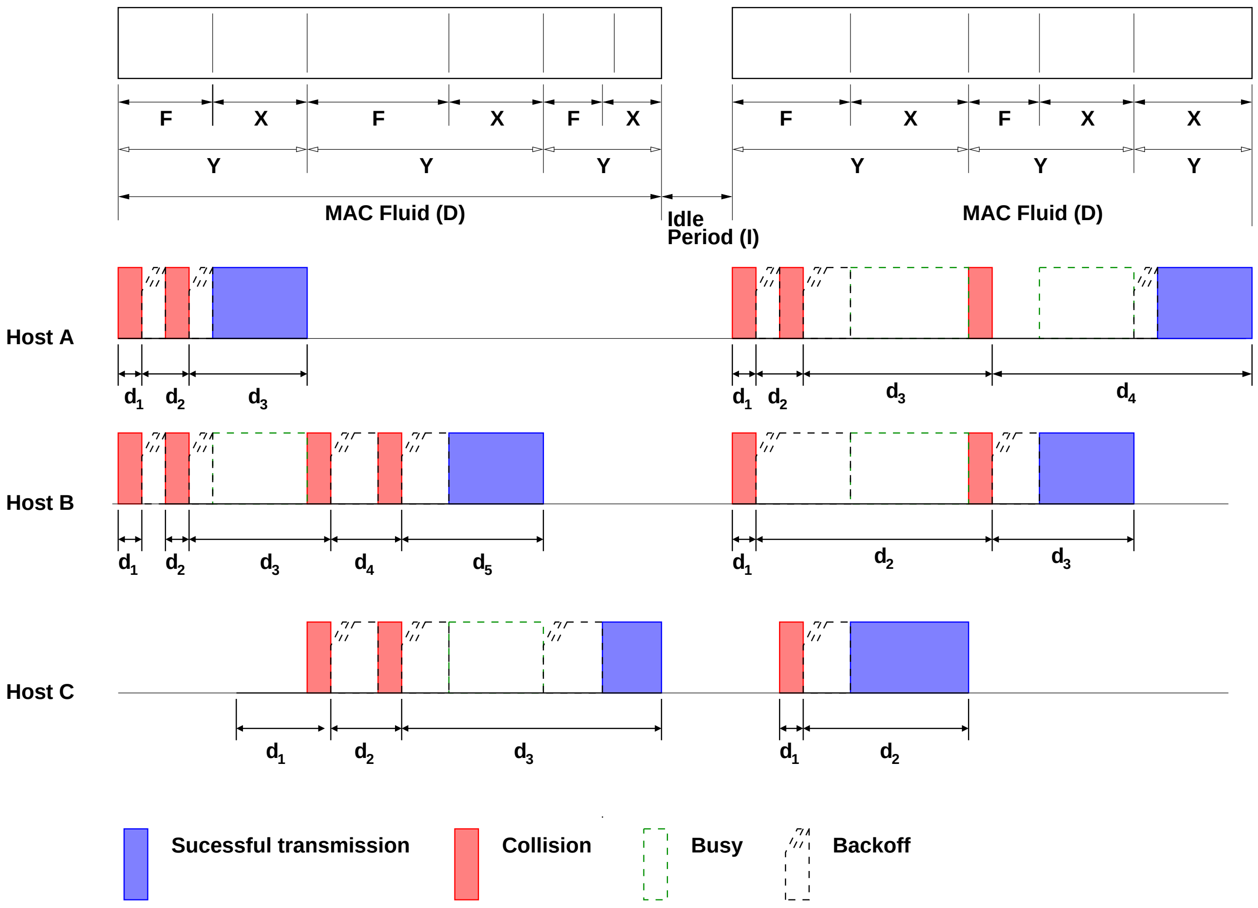


Figure 5.2: Delay experienced by a tagged node.

expected throughput of the foreground flow, $\overline{d}_{fg}$ the expected delay that a frame in the tagged, foreground flow experiences, and $\overline{x'}$ the average frame size. The throughput of a foreground flow can then be expressed as

$$\overline{T}_{fg} = \frac{\overline{x'}}{\overline{d}_{fg}}. \tag{5.1}$$

To derive $\overline{T}_{fg}$, we have to derive $\overline{d}_{fg}$. To facilitate the analysis of $\overline{d}_{fg}$, we define the following random variables

- D_{fg} : the r.v. representing the total delay experienced by a frame in the tagged flow;
- R : the r.v. representing the residual service time as seen by a frame of the foreground flow at its arrival;
- X_{fg} : the r.v. representing the current frame size in the tagged node;
- b_i : the r.v. representing the ith backoff time after the ith collision for $i \geq 0$;

- d_i : the r.v. representing the ith *deferred* backoff time after the ith collision for $i \geq 0$. According to IEEE 802.11, a node cannot decrease its backoff timer when the transmission medium is in use, i.e., $d_i = b_i +$ the time interval during which the medium is in use (Figure 5.2).

With the above notations, we can express D_{fg} as

$$D_{fg} = R + \sum_{i=0}^{\infty} d_i + X_{fg}. \tag{5.2}$$

Note that the deferred time d_i instead of the backoff time b_i is used in Eq. (5.2). By taking the expectation of D_{fg}, R, and X_{fg} in Eq. (5.2), we have

$$E\left[D_{fg}\right] = E[R] + E\left[\sum_{i=0}^{\infty} d_i\right] + E[X_{fg}]. \tag{5.3}$$

Let $\overline{r} \stackrel{\triangle}{=} \lim_{i \to \infty} E[R_i]$, and let $\overline{d}$ and $\overline{x}_{fg}$ denote the expected deferred time till the transmission of the tagged frame and the expected frame size, respectively. Then, we have

$$\overline{d}_{fg} = \overline{r} + \overline{d} + \overline{x}_{fg}. \tag{5.4}$$

The term $\overline{d}$ in Eq. (5.4) can be derived as follows. Let $\overline{b}$ denote the expected backoff window size, T_{slot} a physical slot time defined in Table 3.1, and λ the rate at which the background flows attempts to transmit their frames. (As shown in Chapter 3, λ is approximated to be $\lambda = N \cdot \frac{1}{\overline{b}}$, where N is the number of background nodes.) Then the term $\overline{d}$ in Eq. (5.4) can be written as

$$\begin{aligned}
\overline{d} &= \ell_{size} \cdot \overline{b}, \qquad \text{where} \\
\ell_{size} &= P_{idle,bg} \cdot T_{slot} + P_{collision,bg} \cdot (T_{slot} + \overline{c}_{bg}) \\
&+ P_{success,bg} \cdot (T_{slot} + \overline{x}_{bg}), \\
P_{idle,bg} &= e^{-\lambda}, \\
P_{success,bg} &= \lambda e^{-\lambda}, \\
P_{collision,bg} &= 1 - e^{-\lambda} - \lambda e^{-\lambda},
\end{aligned} \tag{5.5}$$

where $\overline{c}_{bg}$ and $\overline{x}_{bg}$ are, respectively, the expected length of collision period (due to collision of two or more background frames) and successful transmission period, both of which are caused by background flows.

The terms yet to be determined in Eq. (5.5) are $\overline{c}_{bg}$, $\overline{x}_{bg}$, and $\overline{b}$. To derive the former two terms, we define their corresponding random variables: C_{bg} is the r.v. representing the length of background collision period, and X_{bg} the length of successful background frame transmission time. When the RTS/CTS mechanism is employed, we have

$$\begin{aligned} C_{bg} &= RTS + DIFS, \\ X_{bg} &= RTS + SIFS + CTS + SIFS + X' \\ &\quad +SIFS + t_{ACK} + DIFS, \end{aligned}$$

and when the RTS/CTS mechanism is not employed, we have

$$\begin{aligned} C_{bg} &= CF + DIFS, \\ X_{bg} &= X' + SIFS + t_{ACK} + DIFS, \end{aligned}$$

where

- CF: the r.v. representing the size of a collided frame;
- X': the r.v. representing the size of a frame (note that the distribution of CF is the same as that of X');
- $DIFS$, $SIFS$: the system parameters whose values are given in Table 3.1.
- t_{ACK}: another system parameter defined as $t_{ACK} = ACK/1(Mb/s)$.

Let $C^*_{bg}(s)$, $X^*_{bg}(s)$ and $X'^*(s)$ denote, respectively, the Laplace transform of the probability density function associated with C_{bg}, X_{bg}, and X'. Then, we have

$$\begin{aligned} X^*_{bg}(s) &= X'^*(s) \cdot \\ &\quad e^{-(RTS+SIFS+CTS+SIFS+SIFS+ACK+DIFS)s} \end{aligned}$$

in the case that the RTS/CTS mechanism is used, or

$$X^*_{bg}(s) = X'^*(s) \cdot e^{-(SIFS+ACK+DIFS)s}$$

in the case that the RTS/CTS mechanism is not used. In addition, we have

$$C^*_{bg}(s) = e^{-(RTS+DIFS)s}$$

in the case that the RTS/CTS mechanism is used, or

$$C^*_{bg}(s) = CF^*(s) \cdot e^{-DIFSs}$$

in the case that the RTS/CTS mechanism is not used. Since $\overline{c^k_{bg}} = (-1)^k \cdot C^{*(k)}_{bg}(s)|_{s=0}$, we have

$$\overline{c_{bg}} = -C^{*(1)}(s)|_{s=0} = RTS + DIFS \tag{5.6}$$

in the case that the RTS/CTS mechanism is used, or

$$\overline{c_{bg}} = -C^{*(1)}(s)|_{s=0} = \overline{cf} + DIFS \tag{5.7}$$

in the case that the RTS/CTS mechanism is not used, where $\overline{cf}$ is the expectation of CF. Note that the distribution of CF is the same as that of X' and is given.

Similarly, we have the followings by using $\overline{x^k_{bg}} = (-1)^k \cdot X^{*(k)}_{bg}(s)|_{s=0}$

$$\begin{aligned} \overline{x}_{bg} &= RTS + SIFS + CTS + SIFS + \overline{x'} \\ &\quad + SIFS + t_{ACK} + DIFS, \end{aligned} \tag{5.8}$$

when the RTS/CTS mechanism is used; and

$$\overline{x}_{bg} = \overline{x'} + SIFS + t_{ACK} + DIFS \tag{5.9}$$

in the other case.

Finally, we can determine $\overline{d} = \ell_{size} \cdot \overline{b}$ in Eq. (5.5) as follows. Let p represent the probability that collision occurs between one foreground frame and one or more background frames. Recall that b_i is the average backoff window size when the contention window is $\widetilde{CW}_i$; i.e., $b_i = \frac{1}{\widetilde{CW}_i} \sum_{i=0}^{\widetilde{CW}_i - 1} i = \frac{\widetilde{CW}_i - 1}{2}$, and m is the index for the maximum contention window size (in Chapter 3). Then we have

$\widetilde{CW}_i = 2^i \cdot \widetilde{CW}_0$ for $1 \leq i \leq m$, and

$$\begin{aligned}\overline{d} =& \ell_{size} \cdot \overline{b} = \ell_{size} \cdot E\left[\sum_{i=0}^{\infty} b_i\right] \\ =& \ell_{size} \cdot \left[\sum_{i=0}^{m-1}\left\{\frac{\widetilde{CW}_i - 1}{2} \cdot p^i(1-p)\right\} + \right. \\ & \left. \sum_{i=m}^{\infty}\left\{\frac{\widetilde{CW}_m - 1}{2} \cdot p^i(1-p)\right\}\right] \\ =& \ell_{size} \cdot \left[\frac{\widetilde{CW}_0}{2(1-2p)} \cdot (1 - p - p \cdot (2p)^m) - \frac{1}{2}\right].\end{aligned} \tag{5.10}$$

Since the collision between foreground and background traffic occurs when one or more background nodes are attempting to transmit their frames when the backoff timer of the foreground node expires. Hence, the collision probability, p, can be expressed as

$$p = P_{collision,bg} + P_{success,bg} = 1 - e^{-\lambda}. \tag{5.11}$$

The term $\overline{r}$ in Eq. (5.4) is the expected residual service time for background traffic, which can be a collision period or a successful transmission time. Therefore, $\overline{r}$ can be expressed as

$$\overline{r} = P_{collision,bg} \cdot \left(\frac{\overline{c}_{bg}}{2} + \frac{\sigma^2_{C_{bg}}}{2\overline{c}_{bg}}\right) + P_{success,bg} \cdot \left(\frac{\overline{x}_{bg}}{2} + \frac{\sigma^2_{X_{bg}}}{2\overline{x}_{bg}}\right). \tag{5.12}$$

Conclusively, we can express the expected value of d_{fg} as follows:

$$\overline{d}_{fg} = \overline{r} + \ell_{size} \cdot \left\{\frac{\widetilde{CW}_0}{2(1-2p)} \cdot (1 - p - p \cdot (2p)^m) - \frac{1}{2}\right\} + \overline{x}_{fg}, \tag{5.13}$$

where $\overline{r}$, ℓ_{size}, and $\overline{x}_{fg} = \overline{x}_{bg}$ are given in Eqs. (5.12), (5.5), and (5.8) (or (5.9)), respectively. The throughput attained by foreground traffic is then given by Eqs. (5.1) and (5.13).

5.2 Throughput Model for Background Traffic

The aggregate throughput for background traffic should be determined in both the cases in which foreground traffic is present and is not. Fortunately the throughput under both cases can be determined in the same manner as in the work Chapter 3 and [28] (Chapter 3). The only difference is that the number, M, of active nodes (which is needed to compute the attempt rate, λ) is $N-1$ in the latter case, and is N in the former case. The remaining derivation has been given in Chapter 3.

5.3 Model Validation

In this section, we validate the interaction model derived in Section 5.1 via simulation, and compare results against those obtained via both packet mode simulation and fluid model based simulation.

5.3.1 Validation w.r.t. Network Dynamics

We investigate whether or not the interaction model correctly reflects the interaction among one foreground node and other background nodes at the wireless medium, by investigating the dynamics of the TCP behavior in a WLAN. To this end, we implement both the interaction model (Section 5.1) and the TCP fluid model that characterizes the TCP dynamics [36] with MATLAB/Simulink.

The network configuration under which we carry out the experiment is in Figure 5.3. In the figure, a total of 10 wireless nodes exist in the WLAN, each of which has a buffer size of 50 packets, and establishes 20 FTP/TCP flows (with the default TCP packet size set to 500 bytes). The RTS/CTS mechanism is used as the floor acquisition mechanism. Assume that the wireless link has a maximum bandwidth of 1 Mb/s and a constant delay of 25 μs.

Figure 5.4 gives the TCP dynamics and queue dynamics calculated using the TCP dynamics model and the interaction model. Note that the TCP dynamics model [36] expresses the queue length, the RTT, and the window size as functions of the link capacity of the bottleneck link. We use the interaction model to calculate the throughput attained by a node (Eq.(5.1) and divide the value by the number of TCP connections (10). This gives the "link capacity" of the wireless channel as viewed by a TCP connection. This value is then plugged into the TCP dynamics model to calculate the queue length of a node, the RTT, and the window size of a TCP connection. Finally the TCP sending rate is simply obtained by dividing the window size by the RTT.

We now inspect whether or not the results presented in Figure 5.4 are reasonable. As shown in Figure 5.4 (a), the queue length approaches the maximum buffer capacity quickly, as the traffic generated by the 20 wireless nodes (each with 10 TCP/FTP connections) is quite heavy and quickly saturates the wireless link of 1 Mb/s.

Part of the wireless capacity is consumed in (i) transporting the MAC header and the control

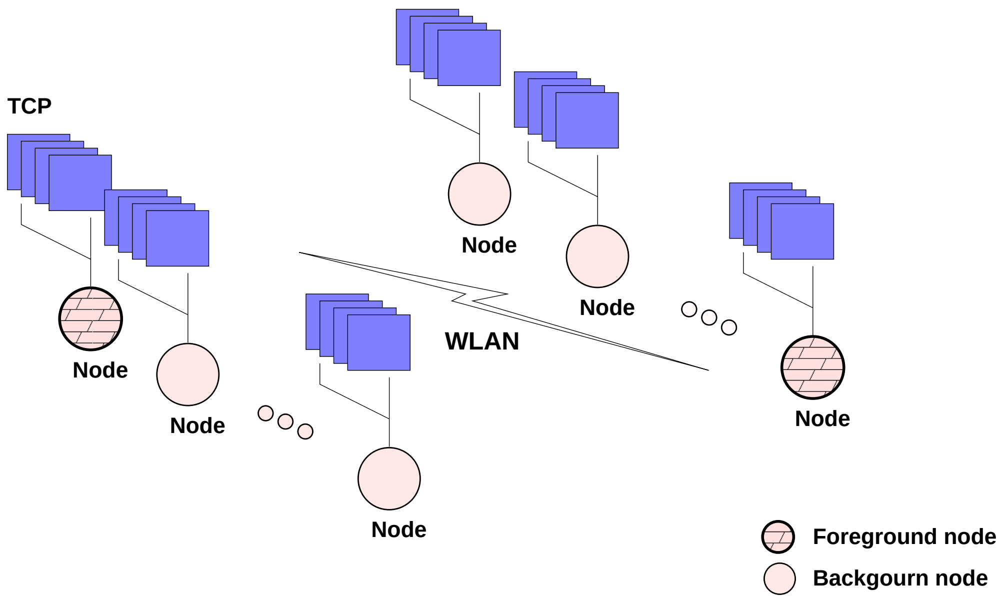


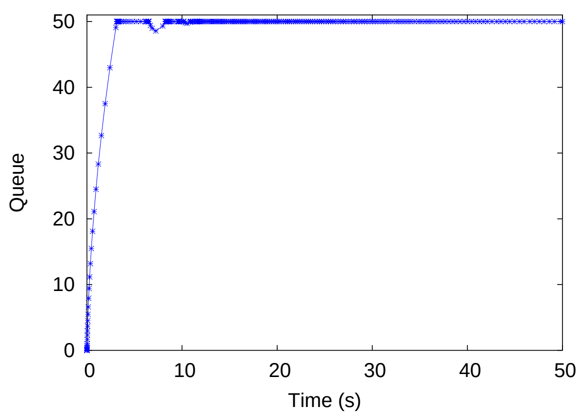
Figure 5.3: Networks for TCP dynamics

packets such as FCS, RTS/CTS/ACK frame, and the inter-frame space such as SIFS and DIFS, (ii) transporting the physical layer PLCP preamble and header, and (iii) collisions and subsequent retransmissions. According to the parameters given in Table 3.1, the overhead incurred due to (i) and (ii) is 1724 bits per packet in IEEE 802.11-operated WLANs. As will be seen in Section 5.3.2, when the number of nodes is 10, the capacity utilization is 70%, excluding all the MAC and physical overheads. Therefore, the RTT of a TCP connection is simply $50 \times (\frac{(500\times8)}{(10^6\times0.70\times\frac{1}{10})} + 25 \times 10^{-6} \times 2) \approx 2.85(s)$, where 50 is the buffer size, 10 is the number of wireless nodes in the WLAN, and $25\times10^{-6}\times2$ is the constant link delay consumed at both the receiver and the sender. This result is corroborated by the result given in Figure 5.4 (b).

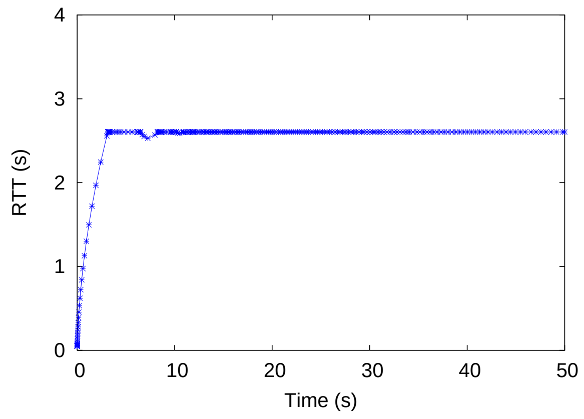
Recall the TCP window size represents the maximum number of packets that can be in transit without being acknowledged the equilibrium state. It is simply the bandwidth-delay product. Therefore, the window size can be calculated as $\frac{(0.70\times10^6\times\frac{1}{10}\times\frac{1}{25})}{(500\times8)} \times 2.85(RTT) \approx 1.99$, where 10 is the number of wireless nodes in the WLAN, and 25 is the number of TCPs per node. This result coincides with the value given in Figure 5.4 (c) in the equilibrium state.

Finally, the TCP sending rate can be simply determined by dividing the window size by the RTT value, and in this case is equal to $\frac{1.99}{2.85} \approx 0.70$. This value is corroborated by the result given in Figure 5.4 (d).

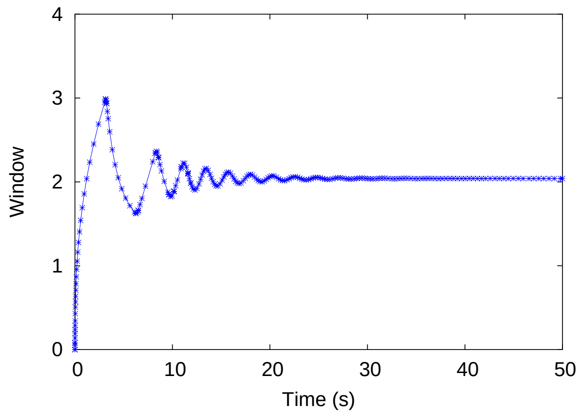
Based on the above calculation, we conclude that the interaction model indeed accurately char-



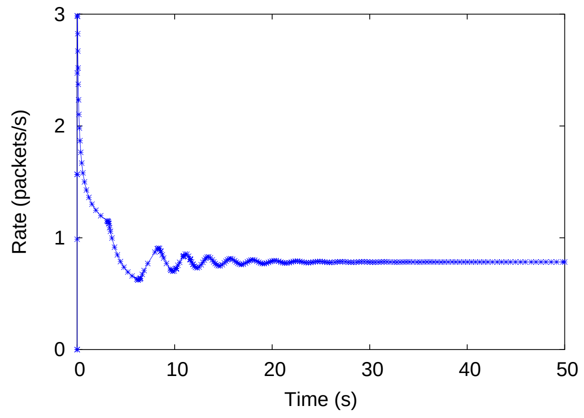
(a) Queue length at a node



(b) RTT



(c) Congestion window



(d) Attainable throughput

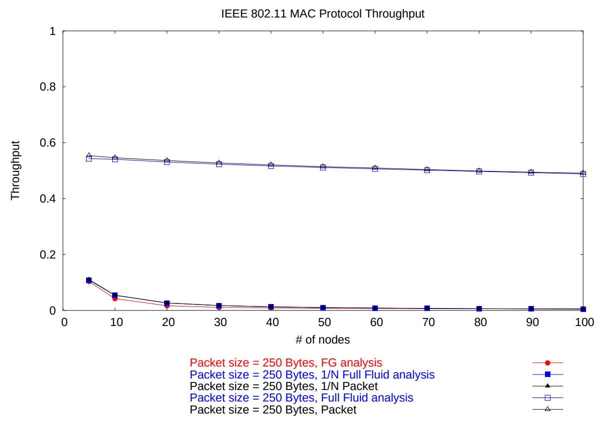
Figure 5.4: TCP dynamics and queue dynamics calculated using the interaction model derived in Section 5.1 and the TCP dynamics model. Note that the queue length ((a)) refers to the number of packets queued at a node, while the other parameters are with respect to a TCP connection.

acterizes the interaction between the foreground traffic and the background traffic in the wireless channel in a WLAN.

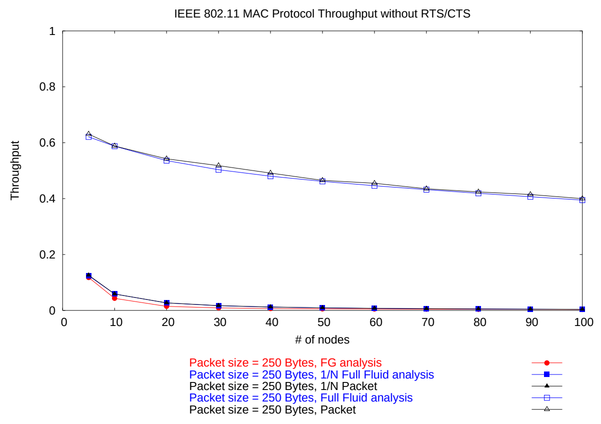
5.3.2 Validation w.r.t. Attainable Throughput

Figure 5.5 depicts the throughput versus the total number of nodes, N, in a IEEE 802.11-operated WLAN with the RTS/CTS mechanism ((a)) and without the RTS/CTS mechanism ((b)). Traffic from one of the nodes is considered as the foreground traffic. The packet size 250 bytes (100 slot time).

The upper two curves in Figure 5.5 represent, respectively the throughput attained by the aggregate traffic (that includes both the background and foreground traffic) in the simulation and



(a) With the RTS/CTS



(b) Without the RTS/CTS

Figure 5.5: Aggregate throughput and throughput attained by the foreground traffic obtained in the simulation and the analytic model. The packet size is 250 bytes.

the analytic model. We observe that they agree extremely well with each other. The bottom three curves depict, respectively, the analytical result of the throughput attained by the foreground traffic (calculated by Eq. (5.1)), and the corresponding throughput obtained in packet mode simulation and fluid model based simulation. Three curves agree well with one another.

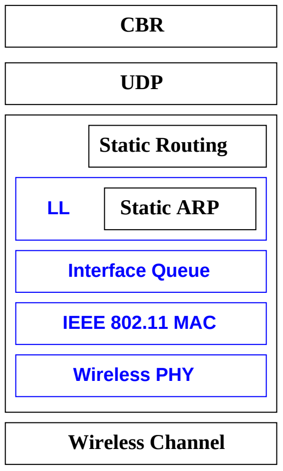
5.4 Simulation Study

We have conducted a simulation study in a variety of network configurations to evaluate the performance of the proposed mixed mode simulation approach in terms of the accuracy and performance as compared with both fluid model based simulation and packet mode simulation.

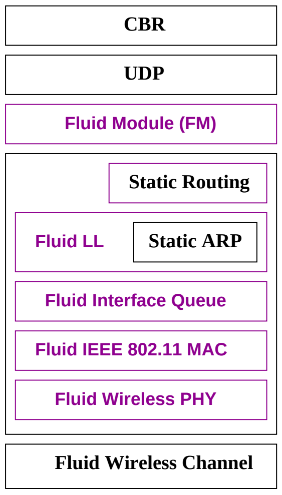
5.4.1 Implementation and Configuration

To realize mixed mode simulation, we extend *ns-2* to conduct both packet mode simulation and fluid model based simulation. Fluid model based simulation is implemented based on the fluid model derived in Chapter 3 and [28]. Traffic that is simulated at the packet mode competes, in compliance with the interaction model derived in Section 5.1, with traffic that is simulated in the fluid mode for the current bandwidth available in the WLAN.

Extension to *ns-2* simulator: To focus on the effect of data transmission-related activities and to filter out other second-order effects in the simulation study, we deliberately leave out several protocol operations in IEEE 802.11, e.g., power saving, beaconing, association and re-association



(a) Foreground node



(b) Background node

Figure 5.6: Protocol stacks for packet mode simulation and fluid model based simulation.

between wireless nodes and access points, and hidden terminal effects. Specifically, we extend *ns-2* as follows.

First, we introduce a virtual wireless LAN node, with which all the wireless nodes communicate with each other through this virtual node. The wireless LAN node uses a static routing algorithm, and also uses a static ARP table. The control overhead considered in the simulation study is therefore solely due to data transmission-related activities (overheads incurred in transmitting RTS/CTS/ACK packets). Second, in order to construct fluid chunks as the abstract simulation units, we exploit the time stepping techniques introduced in [28, 45]. With the time stepping technique, we introduce a new packet type, called the *fluid rate* packet which describes fluid chunks of one flow within one time step. Third, we also include in *ns-2* (i) several new protocol modules that correspond to the fluid model based version of existing link, MAC, physical, and channel layer modules, and (ii) a new module, called *fluid module*, that translates a cluster of packets into a *fluid rate* packet or vice versa (see Chapter 3 and [28]). Last, we extend *ns-2* to include a *modified* packet mode version of existing link, MAC, physical, and channel modules, all of which are adapted to interact with fluid model based simulation according to the proposed interaction model (derived in Section 5.1).

Figure 5.6 (a) and (b) gives, respectively, the protocol stack in the nodes under packet mode simulation and under fluid model based simulation. Note that both the protocol stacks exist and operate simultaneously in mixed mode simulation. Figure 5.7 depicts the interaction between fluid

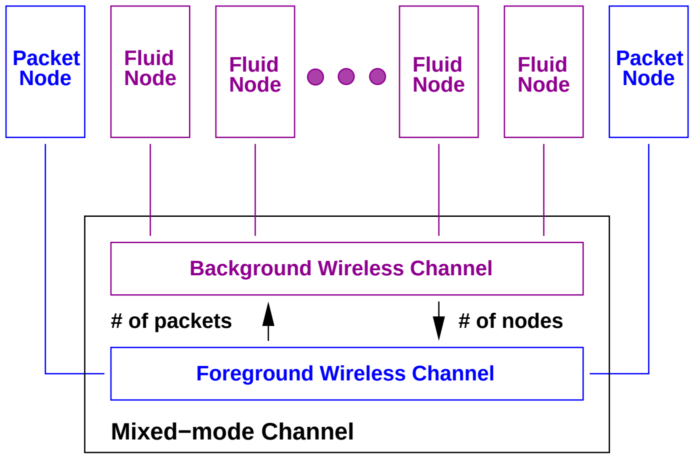


Figure 5.7: The configuration for mixed mode simulation.

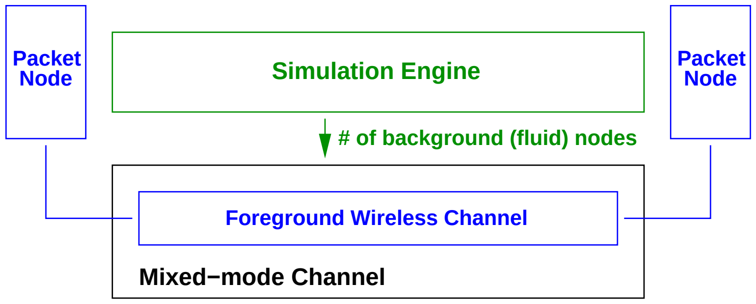


Figure 5.8: The configuration for foreground mode only simulation.

model based simulation and packet mode simulation in the mixed mode simulation framework. The interaction takes place in the *mixed mode channel*, which consists of the fluid model based wireless channel (for background traffic) and the packet mode wireless channel (for foreground traffic). The former channel computes the throughput (allocated to background traffic) according to the total number of active nodes (including the foreground nodes if it exists in a time step). The latter channel is based on the throughput model derived in Section 5.1, and computes the throughput attained by the foreground traffic (which is then used to schedule foreground frames).

Figure 5.8 gives a simplified version of mixed mode simulation, called *foreground only mixed mode simulation*. In this simulation mode, background traffic virtually exist within *ns-2*. With this configuration, only the packet mode wireless channel exists, and the simulation engine *virtually* provides the number of active background nodes for the foreground wireless channel, so as for the latter to estimate the throughput attained by the foreground traffic.

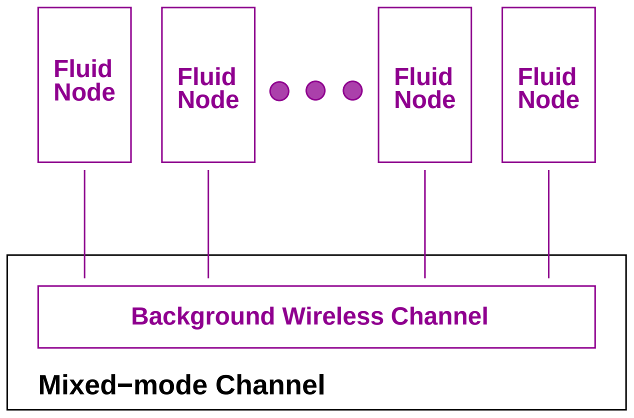


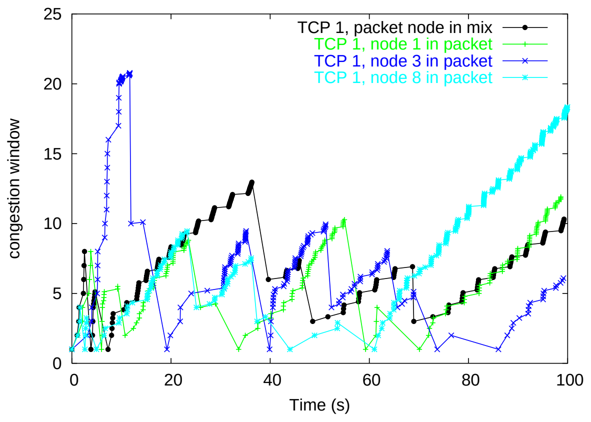
Figure 5.9: The configuration for fluid model based simulation.

Figure 5.9 depicts fluid model based simulation. The detailed explanation is referred to Chapter 3 and [28]. We use this configuration to compare the fluid model based simulation with the mixed mode simulation, in addition to comparison with the packet mode simulation.

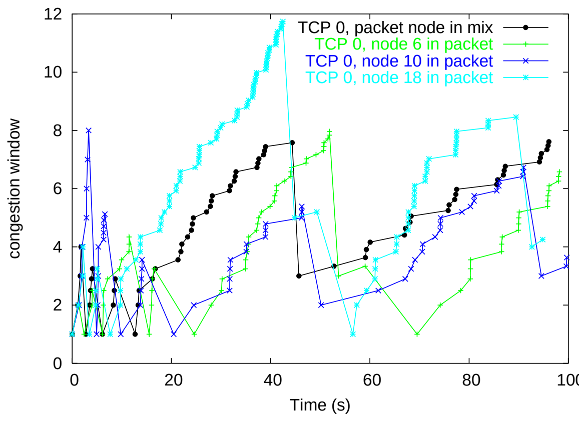
The experiments have been carried out under two different IEEE 802.11 operational modes: one with the RTS/CTS mechanism and the other without, and in each instance of fluid model based simulation, with a variety of time step values. However, due to the space limit, we only present results with the RTS/CTS mechanism. All the simulations are conducted on Linux 2.4.18 on a Pentium 4-1.9 *Ghz* PC with 1 *GBytes* memory memory and with 2 *GBytes* swap memory. We use *ns-2.1b9a*, but upgrade the code of the IEEE 802.11 MAC layer with that available in *ns-2.26*. Each simulation run lasts for 60 simulation seconds for UDP traffic while it lasts for 100 seconds for TCP traffic.

5.4.2 Capability in Retaining Network Dynamics

In this section, we verify whether or not mixed-mode simulation indeed retains accurate packet level dynamics, by keeping track of the congestion window size of TCP connections in both packet mode simulation and mixed-mode simulation. Figure 5.10 depicts the congestion window sizes of 3 (randomly selected) TCP connections in packet mode simulation and that of the foreground TCP connection in mixed-mode simulation (with Figure 5.7 as the mixed-mode configuration). The number of nodes in the WLAN is set to 10 ((a)) and 20 ((b)), respectively. As shown in Figure 5.10, the foreground TCP connection exhibits similar (increasing or decreasing) trends in the window dynamics to those in packet mode simulation.



(a) The number of nodes is 10



(b) The number of nodes is 20

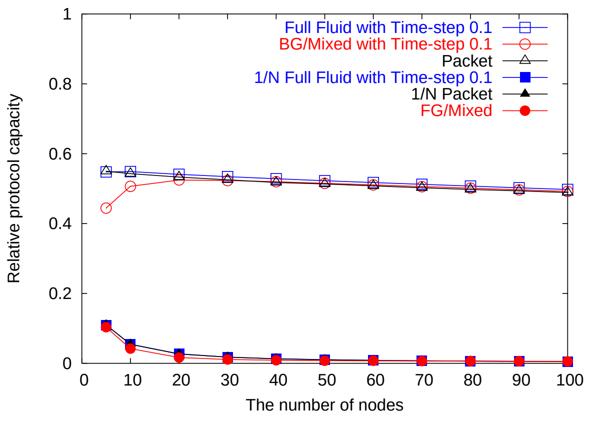
Figure 5.10: TCP congestion window sizes of TCP connections in packet mode simulation (denoted by {TCP #, node # in packet}) and that of the foreground TCP connection in mixed-mode simulation (denoted by { TCP #, packet node in mix }). The simulation is carried out in an IEEE 802.11-operated WLAN with the RTS/CTS mechanism. A total of 10 and 20 nodes exist, each having 5 TCP connections. The TCP packet size is 500 bytes.

5.4.3 Performance w.r.t. Relative Errors and Execution Times

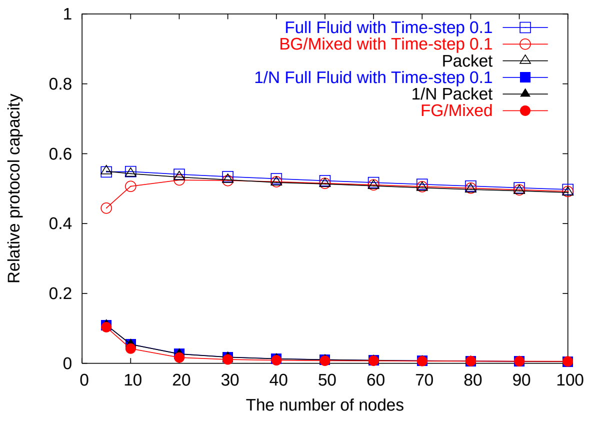
In this section, we evaluate mixed mode simulation in terms of accuracy in simulation results and performance gain in reducing execution time.

Performance in terms of relative errors

Mixed-Mode simulation inevitably trades accuracy for performance efficiency since all the traffic except the foreground traffic is abstracted in fluid model based simulation. In this section, we quantitatively evaluate the discrepancy between results obtained in mixed mode simulation and those respectively obtained in packet mode simulation and in fluid model based simulation. The comparison is made in two steps: in the first step, we study the discrepancy in the aggregate throughput (as a percentage of the maximum bandwidth of the WLAN) between results obtained in fluid model based simulation and those in packet mode simulation. The purpose of this step is to verify that fluid model based simulation accurately characterizes the background traffic. In the second step, we study the discrepancy in the throughput attained by the foreground flow, between results obtained in mixed mode simulation, packet mode simulation, and fluid mode simulation. The purpose of the second step is to validate that the foreground flow (simulated in the packet mode) attains in a fair manner the throughput in the wireless LAN with multiple competing flows.



(a) 25 bytes (10 slot times)

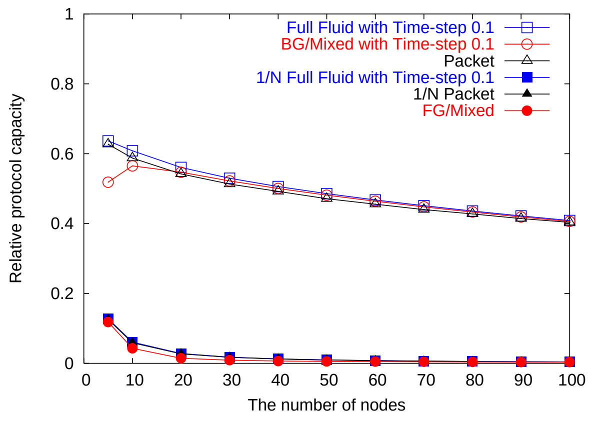


(b) 250 bytes (100 slot times)

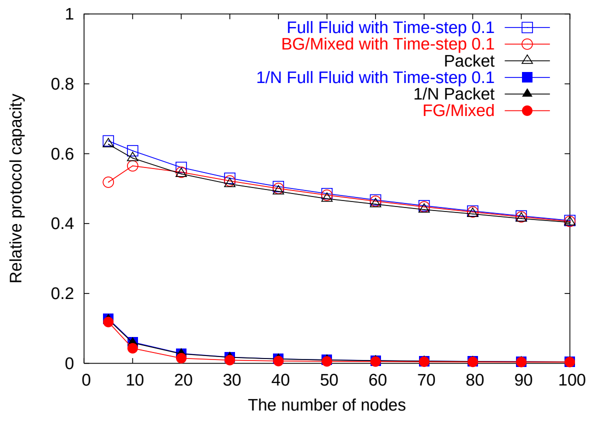
Figure 5.11: Aggregate throughput and throughput attained by the foreground traffic versus the number of nodes in an IEEE 802.11-operated WLAN with the RTS/CTS mechanism. The time step value is 0.1 (s). Packet size are 25 ((a)) and 250 bytes ((b)), respectively. Curves labeled with "Full Fluid," "BG/Mixed," and "Packet" corresponds to the aggregate throughput results in fluid model based simulation, mixed mode simulation, and packet mode simulation, respectively. Curves labeled with "$\frac{1}{N}$ Full Fluid," "$\frac{1}{N}$ Packet," and "FG/Mixed" correspond to the foreground throughput results in fluid model based simulation, packet mode simulation, and mixed mode simulation.

Figure 5.11 compares (i) the aggregate throughput (relative to protocol capacity) between packet mode and fluid model based simulation and (ii) the throughput attained by the foreground traffic (in mixed mode simulation) and the average throughput obtained by a flow in packet mode simulation and fluid model based simulation. In both cases, the error discrepancy is less than 2 % of the protocol capacity as far as the time step value is appropriately chosen. In particular, the throughput attained by the foreground flow agrees extremely well with the average per-flow throughput in pure packet mode simulation or pure fluid model based simulation. We also observe that for simulation results without the RTS/CTS mechanism, the error discrepancy also falls within 2 %. Figure 5.12 shows the simulation results in the same configuration as in Figure 5.11, except that the RTS/CTS mechanism is not used. The same observation is made.

Results in WLANs of extremely large sizes To investigate whether or not the error discrepancy is still within the 2 % bound, when the size of the WLAN grows, the same experiments have been conducted in a WLAN with a (perhaps unreasonably) large number of nodes. Figure 5.13 gives the protocol capacity (both fluid model based and mixed mode simulation) versus the number of nodes in a WLAN of size up to 1000 nodes, when the packet size is fixed at 25 (10) and 250



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

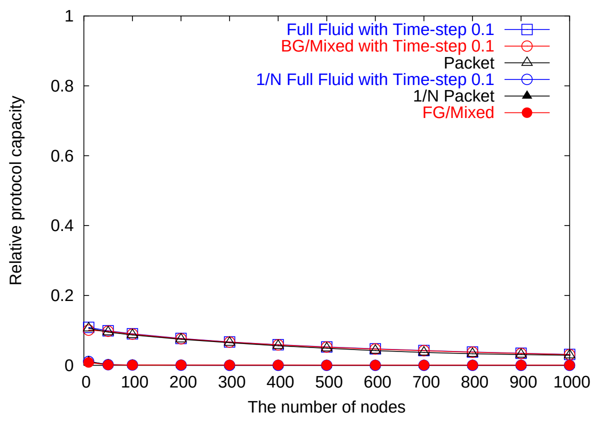
Figure 5.12: Aggregate throughput and throughput attained by the foreground traffic versus the number of nodes in an IEEE 802.11-operated WLAN without the RTS/CTS mechanism. The time step value is 0.1 (s). Packet size are 25 ((a)) and 250 bytes ((b)), respectively.

bytes (100 slot times) respectively, and when the time step value is 0.1 (s). Again, the relative error discrepancy in both the aggregate throughput and the foreground throughput (both relative to the maximum protocol capacity) falls within 2 %. Figure 5.14 shows the simulation results in the same configuration as in Figure 5.13, except that the RTS/CTS mechanism is not used. The same observation as done in Figure 5.13 is made.

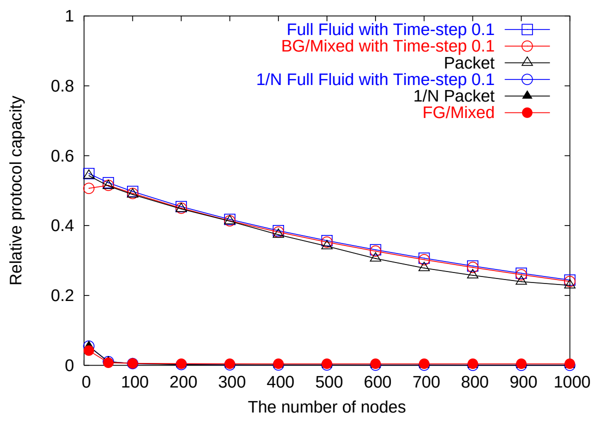
Performance in terms of execution time

This section presents the performance improvement (in terms of execution time) that mixed mode simulation makes, as compared with packet mode simulation. We also study the overhead incurred in mixed mode simulation which does not exist in fluid model based simulation.

Figure 5.15 depicts the execution time versus the number of nodes among packet mode simulation, fluid model based mixed mode simulation (Figure 5.7), and the simplified version of mixed mode simulation (Figure 5.8). The RTS/CTS mechanism is used, the time-step value is set to 0.1 (s) and the packet size is 25 and 250 bytes, respectively. Packet level simulation incurs the most execution time. Both mixed mode simulation and fluid model based simulation achieve about two orders of magnitude of improvement as compared to packet mode simulation. Mixed-Mode simulation (labeled as BG/MIXED) is slightly slower than fluid model based simulation (labeled as Full Fluid), due to the overhead incurred in its interaction with the packet mode simulation. On the other hand, the simplified version of mixed mode simulation (labeled as FG/MIXED), in



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

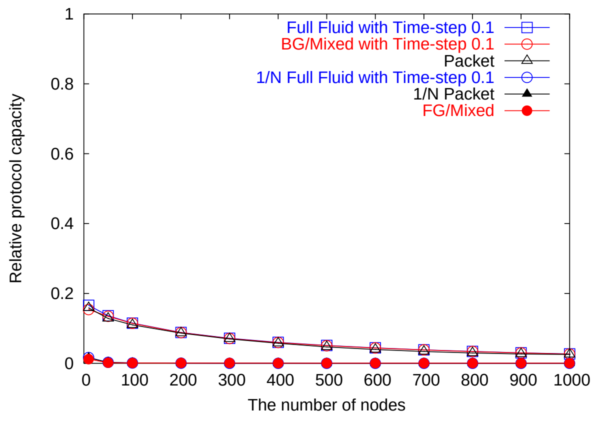
Figure 5.13: Aggregate throughput and throughput attained by the foreground traffic versus the number of nodes in an IEEE 802.11-operated WLAN with the RTS/CTS mechanism. The time step value is 0.1 (s). Packet size are 25 ((a)) and 250 bytes ((b)), respectively.

which background traffic virtually exists, performs the best.

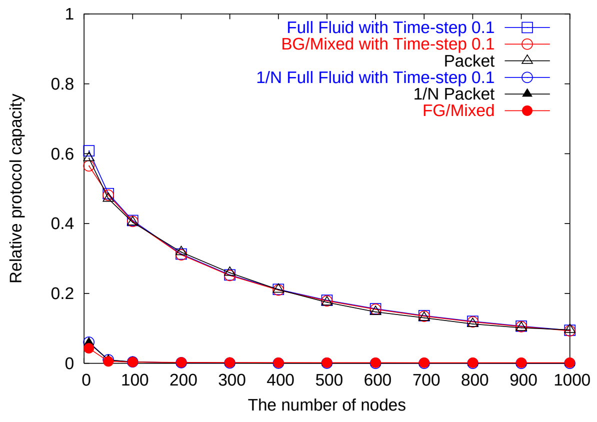
Figure 5.16 presents the simulation results obtained in the same configuration as in Figure 5.15 except that the RTS/CTS mechanism is not employed. The level of improvement in the cases without RTS/CTS is a little bit lower than that in the cases with RTS/CTS. This observation results form the fact that events incurred in processing RTS/CTS frames themselves can be saved in the cases without RTS/CTS.

Figure 5.17 evaluates the effect of time step values on the performance of mixed mode simulation in the same configuration used in Figure 5.15. Four different values of time steps are used. We observe approximately two orders of magnitude improvement in mixed mode simulations (with the time step values varying from 0.01 to 1.0), and more than two orders of magnitude improvement in the simplified version of mixed mode simulation. The improvement is especially pronounced when the number of nodes increases or the packet size decreases. This is because under these conditions the packet mode simulation generates more events to be processed.

Figure 5.18 conducts the same evaluation as done in Figure 5.17 except that the RTS/CTS is not used. The same observation is made: about two orders of magnitude improvement in the mixed mode simulations is made, and furthermore, more than two orders of magnitude improvement in the simplified mixed mode simulation (foreground only simulation) is observed. Due to the same reason inferred in the results of Figure 5.16, the level of improvement in the cases without RTS/CTS is a little bit lower than that in the cases with RTS/CTS.



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

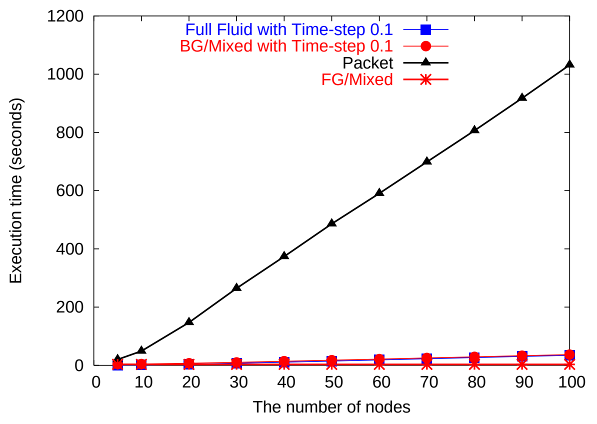
Figure 5.14: Aggregate throughput and throughput attained by the foreground traffic versus the number of nodes in an IEEE 802.11-operated WLAN without the RTS/CTS mechanism. The time step value is 0.1 (s). Packet size are 25 ((a)) and 250 bytes ((b)), respectively.

Results in WLANs of extremely large sizes Figure 5.19 gives the execution time versus the number of nodes in WLANs (of up to 1000 nodes) with the RTS/CTS mechanism, under packet mode simulation, fluid model based simulation, mixed mode simulation, and the simplified version of mixed mode simulation. The time-step value is set to 0.1 (s), and the packet size is 25 (10) and 250 bytes (100 slot times), respectively. The same observation as in Figure 5.15 can be made.

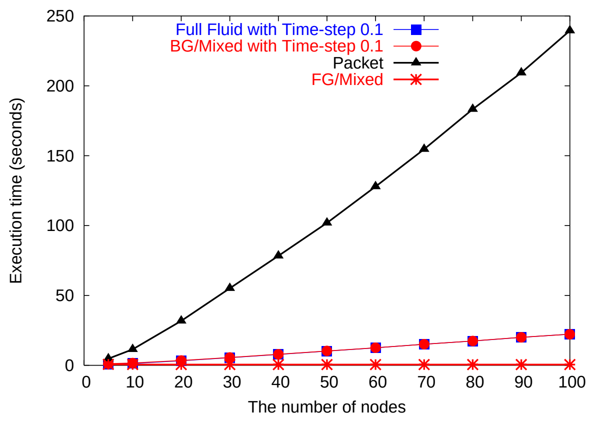
Figure 5.20 shows the results in the cases without the RTS/CTS mechanism. The same observation as done in Figure 5.19 is made.

Figure 5.21 evaluates the effect of time step values on the performance of mixed mode simulation in the same configuration used in Figure 5.19. Four different values of time steps are used. The same observation as in Figure 5.17 can be made: as compared to packet mode simulation, approximately two orders of magnitude improvement has been made in mixed mode simulations (with the time step values varying from 0.01 to 1.0), and more than two orders of magnitude improvement in the simplified version of mixed mode simulation. Figure 5.22 presents the results in the cases without the RTS/CTS mechanism. The similar trend is observed.

Results in multiple-WLAN, multiple-application scenarios To study whether or not the performance improvement levels off as the network size further increases, we evaluate the performance for large scale networks in which multiple WLANs (each of which is made up of multiple nodes) exist and are interconnected via "bridge" wireless nodes. Bridge nodes are connected by wired links in a ring structure (Figure 5.23). In this configuration, both the number of applications



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

Figure 5.15: Execution time under (i) packet mode simulation, (ii) fluid model based simulation, (iii) mixed mode simulation, and (iv) a simplified version of mixed mode simulation, in IEEE 802.11 operated WLANs with the RTS/CTS mechanism. The number of nodes increases up to 100 nodes, the time-step value is 0.1 (s), the packet size is 25 (10) and 250 bytes (100 slot times), respectively.

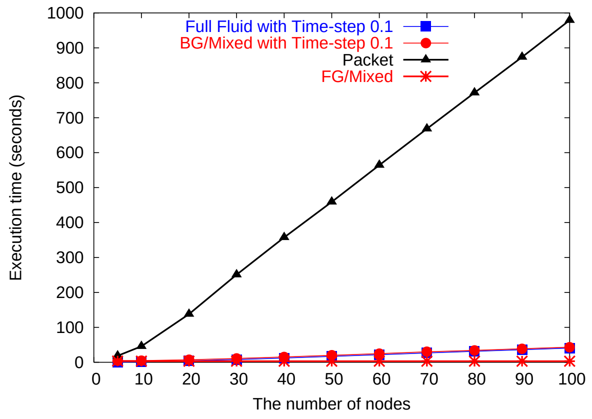
per node and the number of WLANs in the network to be simulated may vary.

Figure 5.24 gives the execution time versus the number of applications per node, in the large mixed-mode network (composed of 5 WLANs, each of which consists of 20 nodes), under packet mode simulation, fluid model based simulation, mixed mode simulation, and the simplified version of mixed mode simulation. The packet size is fixed at 25 (10) and 250 bytes (100 slot times), respectively. The same observation as in Figures 5.15 and 5.19 can be made.

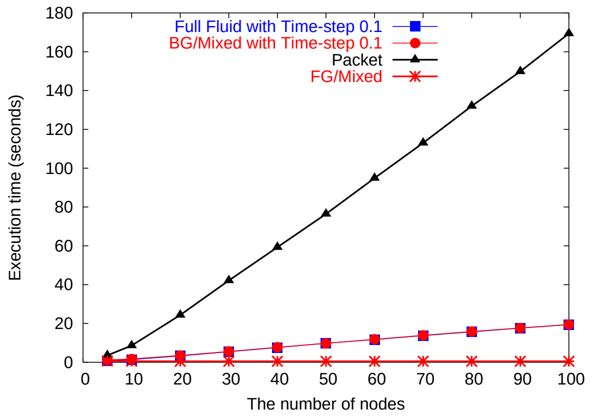
Figure 5.25 shows the results without the RTS/CTS mechanism. The same trend appears in this simulation.

Figure 5.26 evaluates the effect of time step values on the performance of mixed mode simulation in the same configuration used in Figure 5.24. Four different values of time steps are used. The same observation as in Figures 5.17–5.21 can be made.

Figure 5.27 presents the cases without the RTS/CTS mechanisms. We also observe more than two orders of magnitude improvement in the simulation.

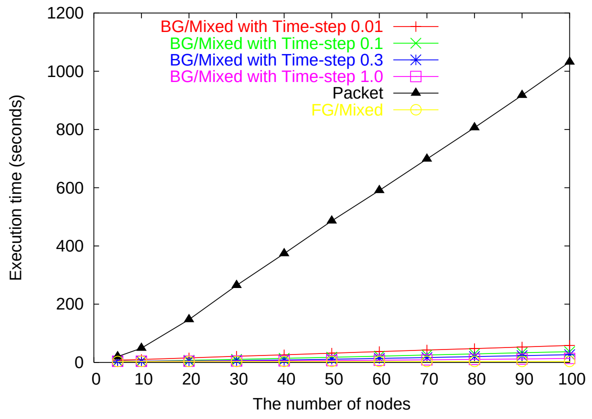


(a) 25 bytes (10 slot times)

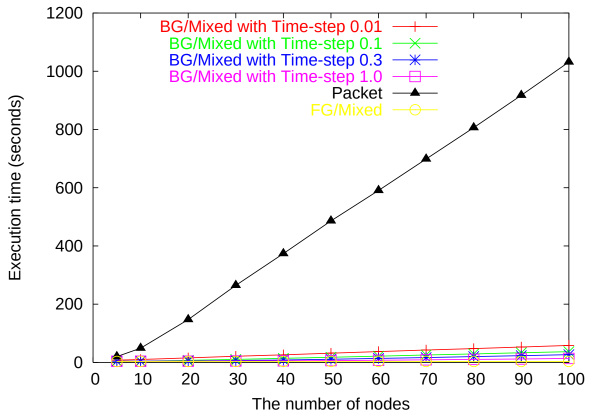


(b) 250 bytes (100 slot times)

Figure 5.16: Execution time under (i) packet mode simulation, (ii) fluid model based simulation, (iii) mixed mode simulation, and (iv) a simplified version of mixed mode simulation, in IEEE 802.11 operated WLANs without the RTS/CTS mechanism. The number of nodes increases up to 100 nodes, the time-step value is 0.1 (s), the packet size is 25 (10) and 250 bytes (100 slot times), respectively.

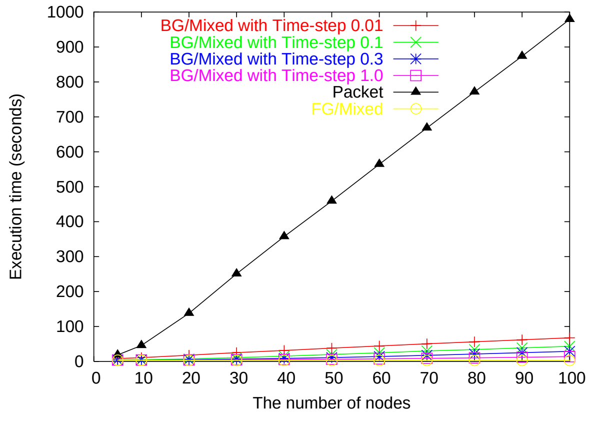


(a) 25 bytes (10 slot times)

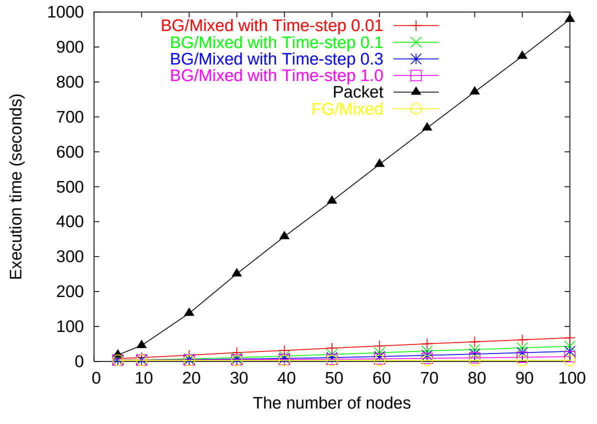


(b) 250 bytes (100 slot times)

Figure 5.17: Effect of time step values on the performance of mixed mode simulation. The number of nodes increases up to 100 nodes, the packet size is 25 (10) and 250 bytes (100 slot times), respectively, and the IEEE 802.11-operated WLAN is equipped with the RTS/CTS mechanism.

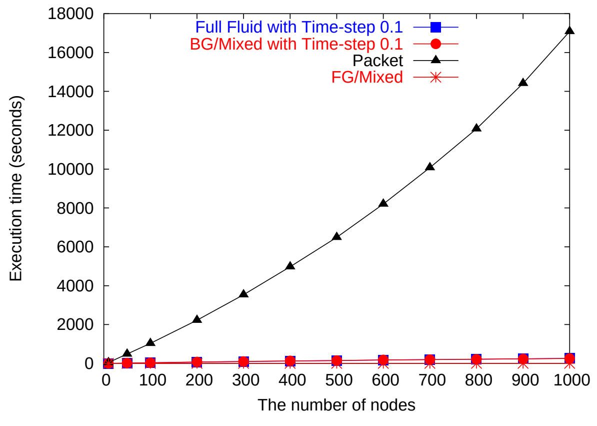


(a) 25 bytes (10 slot times)

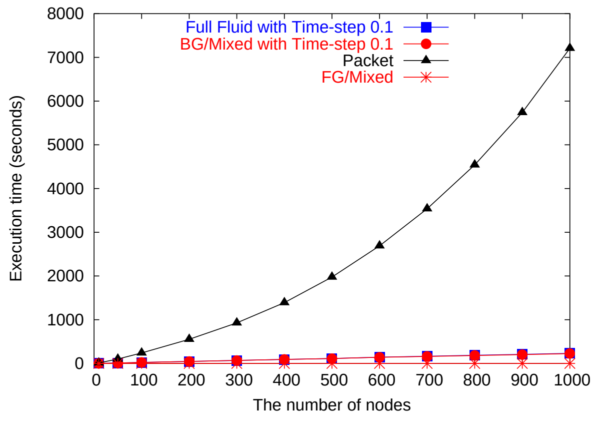


(b) 250 bytes (100 slot times)

Figure 5.18: Effect of time step values on the performance of mixed mode simulation. The number of nodes increases up to 100 nodes, the packet size is 25 (10) and 250 bytes (100 slot times), respectively, and the IEEE 802.11-operated WLAN is equipped without the RTS/CTS mechanism.

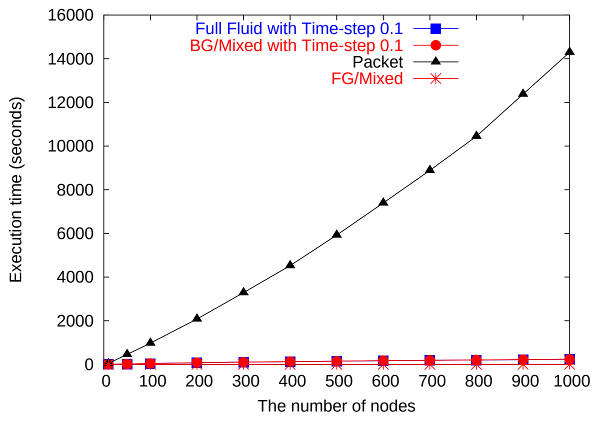


(a) 25 bytes (10 slot times)

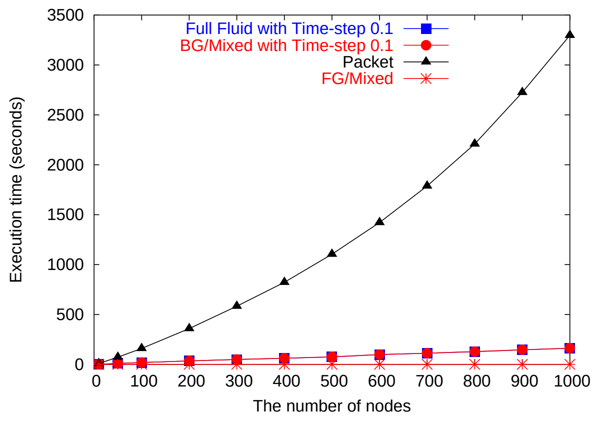


(b) 250 bytes (100 slot times)

Figure 5.19: Execution time under (i) packet mode simulation, (ii) fluid model based simulation, (iii) mixed mode simulation, and (iv) a simplified version of mixed mode simulation, in IEEE 802.11 operated WLANs (up to 1000 nodes) with the RTS/CTS mechanism. The time-step value is 0.1 (s), and the packet size is 25 (10) and 250 bytes (100 slot times), respectively.

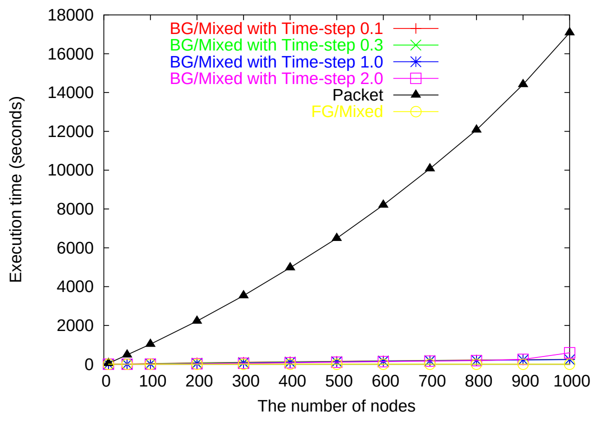


(a) 25 bytes (10 slot times)

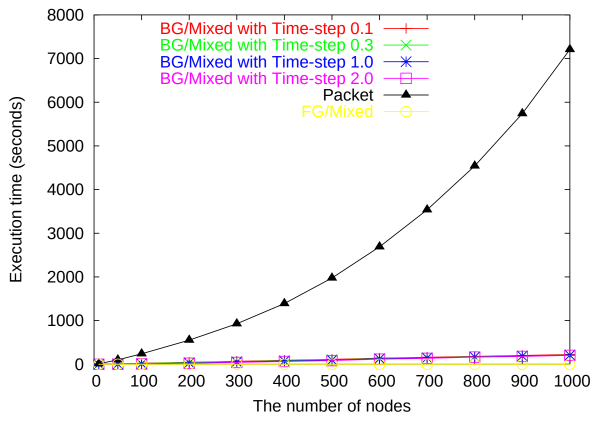


(b) 250 bytes (100 slot times)

Figure 5.20: Execution time under (i) packet mode simulation, (ii) fluid model based simulation, (iii) mixed mode simulation, and (iv) a simplified version of mixed mode simulation, in IEEE 802.11 operated WLANs (up to 1000 nodes) without the RTS/CTS mechanism. The time-step value is 0.1 (s), and the packet size is 25 (10) and 250 bytes (100 slot times), respectively.

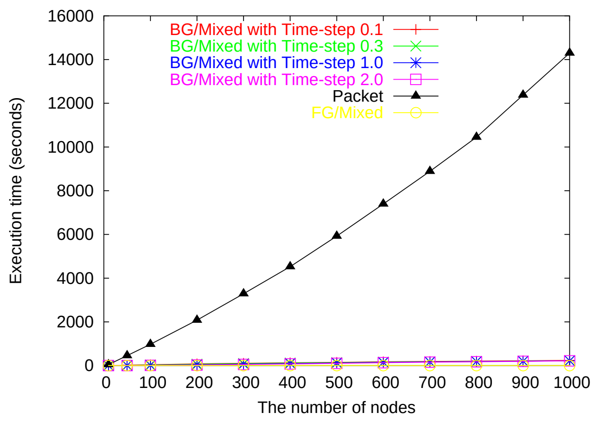


(a) 25 bytes (10 slot times)

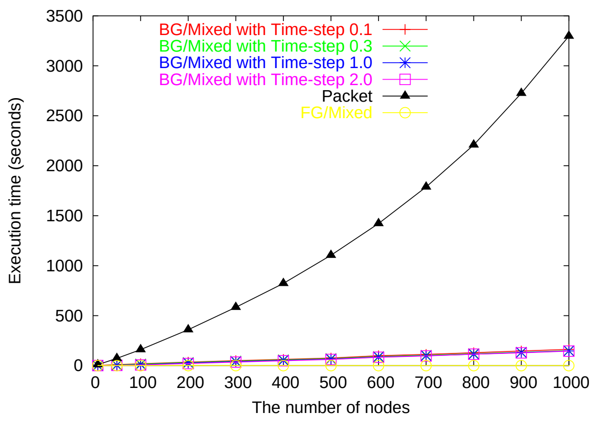


(b) 250 bytes (100 slot times)

Figure 5.21: Effect of time step values on the performance of mixed mode simulation. The number of nodes increases up to 1000 nodes, the packet size is 25 (10) and 250 bytes (100 slot times), respectively, and the IEEE 802.11-operated WLAN is equipped with the RTS/CTS mechanism.



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

Figure 5.22: Effect of time step values on the performance of mixed mode simulation. The number of nodes increases up to 1000 nodes, the packet size is 25 (10) and 250 bytes (100 slot times), respectively, and the IEEE 802.11-operated WLAN is equipped with the RTS/CTS mechanism.

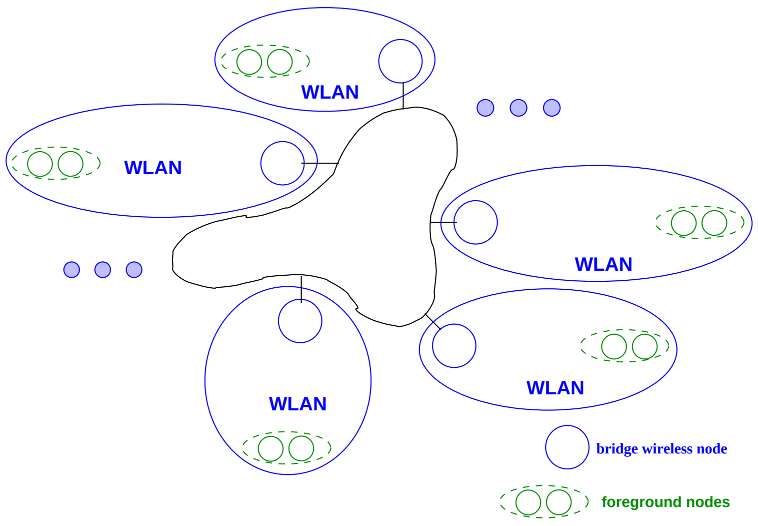
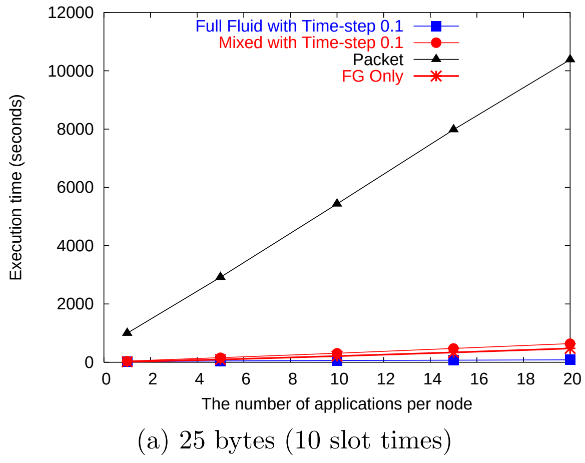
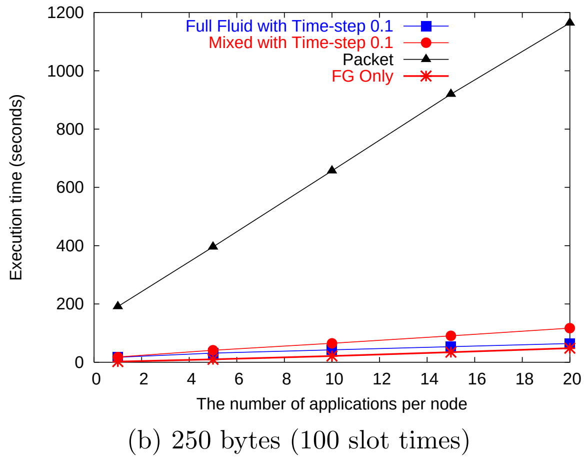


Figure 5.23: Multiple WLANs interconnected via bridge wireless nodes.

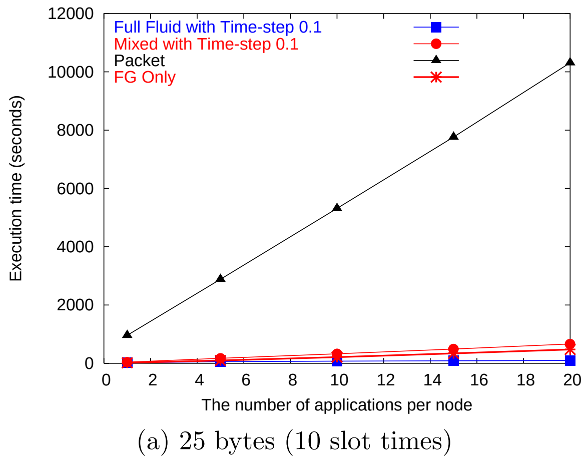


(a) 25 bytes (10 slot times)

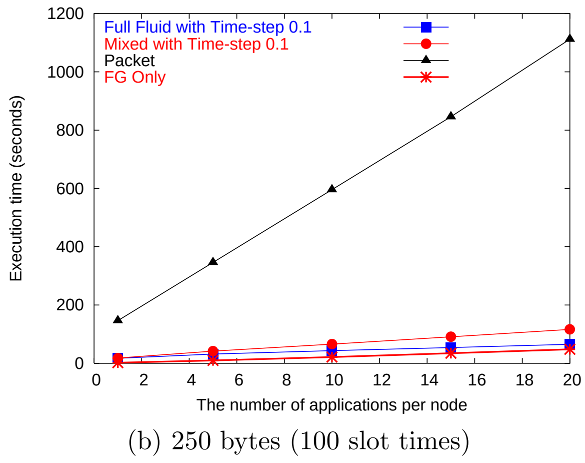


(b) 250 bytes (100 slot times)

Figure 5.24: Execution time under (i) packet mode simulation, (ii) fluid model based simulation, (iii) mixed mode simulation, and (iv) a simplified version of mixed mode simulation, in a large mixed-mode network (composed of 5 WLANs, each of which consists of 20 nodes). The time-step value is 0.1 (s), and the packet size is 25 (10) and 250 bytes (100 slot times), respectively.

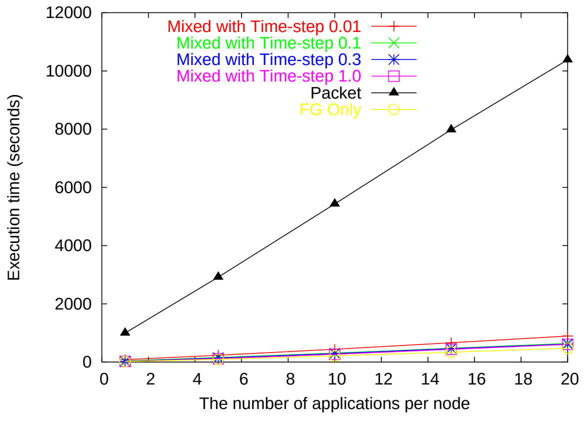


(a) 25 bytes (10 slot times)

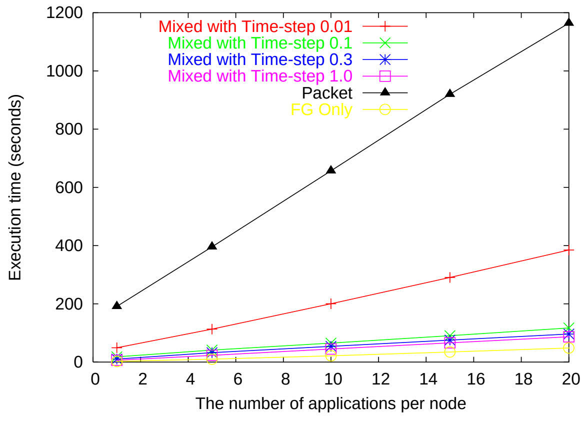


(b) 250 bytes (100 slot times)

Figure 5.25: Execution time under (i) packet mode simulation, (ii) fluid model based simulation, (iii) mixed mode simulation, and (iv) a simplified version of mixed mode simulation, in a large mixed-mode network (composed of 5 WLANs, each of which consists of 20 nodes). The time-step value is 0.1 (s), and the packet size is 25 (10) and 250 bytes (100 slot times), respectively. Both do not use the RTS/CTS mechanism.

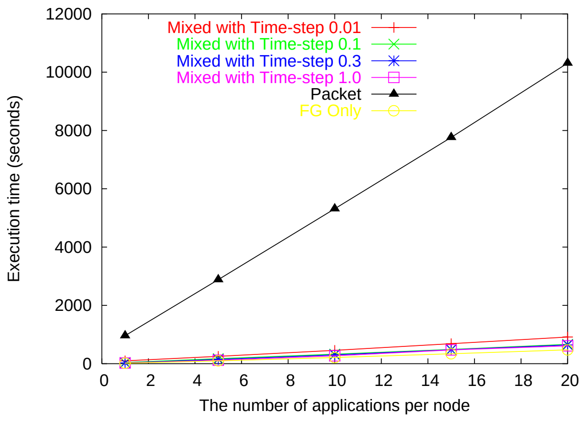


(a) 25 bytes (10 slot times)

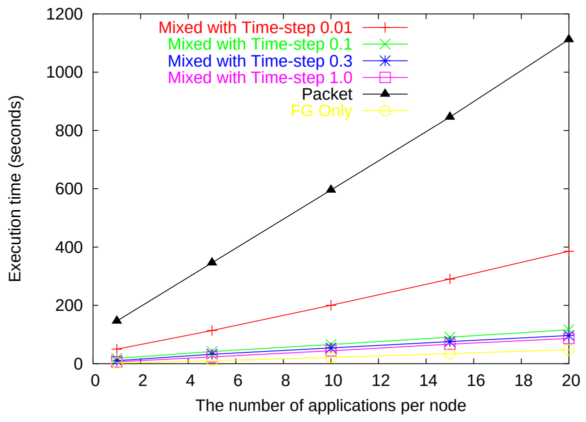


(b) 250 bytes (100 slot times)

Figure 5.26: Effect of time step values on the performance of mixed mode simulation. The mixed-mode network is composed of 5 WLANs (equipped with the RTS/CTS mechanism), each of which contains 20 wireless nodes. The packet size is 25 (10) and 250 bytes (100 slot times), respectively.



(a) 25 bytes (10 slot times)



(b) 250 bytes (100 slot times)

Figure 5.27: Effect of time step values on the performance of mixed mode simulation. The mixed-mode network is composed of 5 WLANs (which do not employ the RTS/CTS mechanism), each of which contains 20 wireless nodes. The packet size is 25 (10) and 250 bytes (100 slot times), respectively.

Chapter 6

Mixed Mode Simulation for TCP Networks

In this chapter, we investigate how to integrate packet level simulation with network calculus based simulation for large scale TCP/IP networks. Network calculus simulation interacts with packet level simulation in the buffer at the point of bottleneck link. Based on existing network calculus simulation, we propose a simple analytical model to implement such the interaction between two mode simulations. Then we implement mixed-mode simulation simulation in *ns-2*, conduct simulation in both the packet mode and mixed-mode simulation, and measure the performance gain in terms of the execution time thus reduced and the error discrepancy in terms of the discrepancy between NC-based simulation results and packet-level simulation results.

6.1 Interaction Model

In this section, we define the model of interaction among a foreground flow and the other network calculus based flows (NC-based flows).

Figure 6.1 presents the interaction between packet mode and network calculus based simulation in mixed mode simulation framework. According to [29], each flow in network calculus based simulation estimates the amount of TCP traffic to be sent during an interval, T, and then sends only the information of the amount at the beginning point of the interval. The top portion in Figure 6.1 presents the behavior of the NC-based flows, and the bottom portion shows an arrival sequence of packets belong to foreground flow.

At each time step, simulation engine estimates the amount of TCP traffic to be sent at each TCP sender, and computes the amount of packets dropped, queued, or transmitted at each link,

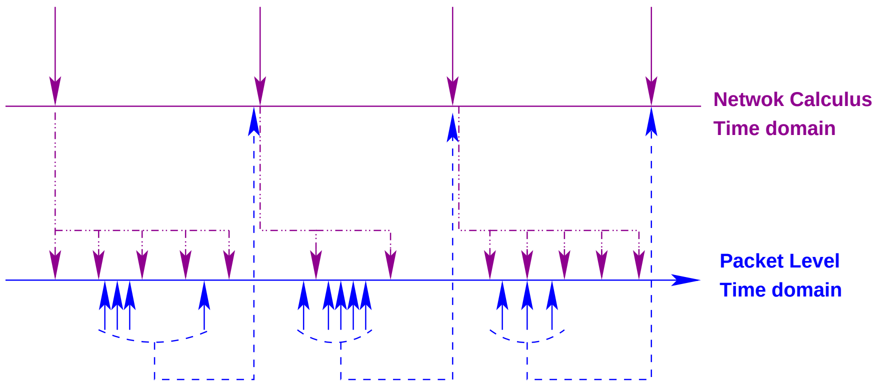


Figure 6.1: Interaction among one packet flow and the other NC-based flows

according to network calculus rules in [29]. However, if there might be some activities in the foreground flow during last time interval, the simulation engine should consider buffer occupation, link capacity, and amount of packet droppings consumed by the foreground flow.

On the contrary, whenever a foreground packet arrives at the link (let t be the arrival time), the simulation engine uses the information collected from NC-based flows which arrived at the previous time step. The engine assumes that all the packets belong to NC-based flows are evenly spaced over the time step, and computes the number of packets that arrives from NC-based flows until t. Then, it computes the current buffer size, the available link capacity, and the number of dropped packets incurred until t. Based on the computation, the simulation engine decides whether or not it sends the packet to next hop, whether or not it drops the packet, or whether or not it queues the packet.

6.1.1 The Behavior of Foreground Traffic

A foreground packet, P_{fg}, arrives at the point of bottleneck in an interval, $T_i = [t_i, t_{i+1}]$, and sees the current queue length, which are determined by both background flows and foreground flow arrived ahead of its arrival.

$\widehat{q}_{fg}(t)$ when $t_i \leq t \leq t_{i+1}$, *packet mode queue*, denotes the current queue the foreground packet sees. $q_{bg}(t_i)$, *network calculus based queue*, represents the queue length at the beginning point of the interval, T_i. r^j_{bg} denotes the rate of one background flow, j, during T_i. N presents the number of background flows. $\widehat{n}_{fg}(t)$ represents the number of foreground packets transmitted during sub-interval $[t_i, t]$. C_l, P_l, and Q denotes the capacity, the propagation delay, and the maximum queue length of the bottleneck link, l, respectively.

Then, $\widehat{q}_{fg}(t)$ is as follows:

$$\widehat{q}_{fg}(t) = \min\left[Q, q_{bg}(t_i) + \sum_{j=1}^{N} r_{bg}^{j} \cdot (t - t_i) + \widehat{q}_{fg}(t^-) - (C_l \cdot (t - t_i) - \widehat{n}_{fg}(t))\right], \tag{6.1}$$

where t^- denotes the time just before the foreground packet arrives at t.

Based on Eq. (6.1), the foreground packet firstly decides whether or not it drops itself, and then decides whether or not it queues or sends itself: in other words, it is dropped when $\widehat{q}_{fg}(t)$ is equal to Q; otherwise, it is queued and then transmitted.

The packet experiences the next queuing delay,

$$d_q = \frac{\widehat{q}_{fg}(t)}{C_l},$$

and takes the following transmission delay also:

$$d_t = \frac{size(P_{fg})}{C_l},$$

where $size(\cdot)$ is a function of determining the packet size. Then, the packet arrives at the next link after P_l link delay.

In order to correctly reflect the effect of foreground flows into background flows, all the resource usages of foreground flow during the current interval should be kept track of. Let $n_{fg}(t)$, $q_{fg}(t)$, and $l_{fg}(t)$ denote respectively the number of transmitted packets, the number of queued packets, and the number of dropped packets observed from the foreground flow at time t when $t_i \leq t \leq t_{i+1}$. The specific algorithm is as follows: (i) when $(t_{i+1} - t) > (d_q + d_t + P_l)$ and $\widehat{q}_{fg}(t) < Q$,

$$n_{fg}(t^+) = n_{fg}(t) + 1, q_{fg}(t^+) = q_{fg}(t), l_{fg}(t^+) = l_{fg}(t);$$

when t^+ denotes the time just after the packet arrived at t is processed; (ii) when $(t_{i+1} - t) < (d_q + d_t + P_l)$ and $\widehat{q}_{fg}(t) < Q$,

$$n_{fg}(t^+) = n_{fg}(t), q_{fg}(t^+) = q_{fg}(t) + 1, l_{fg}(t^+) = l_{fg}(t);$$

(iii) when $\widehat{q}_{fg}(t) = Q$,

$$n_{fg}(t^+) = n_{fg}(t), q_{fg}(t^+) = q_{fg}(t), l_{fg}(t^+) = l_{fg}(t) + 1;$$

(iv) when the time, $(d_q + d_t + P_l)$, in case (ii) expires,

$$n_{fg}(t^+) = n_{fg}(t) + 1, q_{fg}(t^+) = q_{fg}(t) - 1, l_{fg}(t^+) = l_{fg}(t);$$

(v) when the current time step expires

$$n_{fg}(t_{i+1}^+) = 0, q_{fg}(t_{i+1}^+) = q_{fg}(t_{i+1}), l_{fg}(t_{i+1}^+) = 0.$$

Note that $\widehat{q}_{fg}(t)$ and $\widehat{n}_{fg}(t)$ differ from $q_{fg}(t)$ and $n_{fg}(t)$ in that $\widehat{q}_{fg}$ is $q_{fg}(t)$ plus the number of foreground packets not transmitted yet and $\widehat{n}_{fg}(t)$ is the number of foreground packets really transmitted until t, so that $\widehat{q}_{fg}(t) \geq q_{fg}(t)$, $\widehat{n}_{fg}(t) \leq n_{fg}(t)$, and $\widehat{q}_{fg}(t) + \widehat{n}_{fg}(t) = q_{fg}(t) + n_{fg}(t)$.

6.1.2 The Behavior of Background Traffic

All the background flows cannot fully exploit C_l when they arrive at the link, l, as far as some foreground packets appear and use some of it (Figure 6.1). In addition to link capacity, the foreground flow also consumes buffer at the link (Eq. (6.1). Therefore, even though background traffic exactly follows the method introduced in [29], it should use the remained amount of capacity and queue length as follows.

Following the notations in [29], let R, Y, L, and q denote, respectively, the cumulative amount received, the cumulative amount transmitted, the cumulative amount of dropped packets, and the buffer size at a link.

$$Y(t_{i+1}) = Y(t_i) + \min\left(\sum_{j=1}^{N} R^j(t_i, t_{i+1}), (C_l(t_{i+1}, t_i) - n_{fg}(t_{i+1}))\right), \tag{6.2}$$

$$q(t_{i+1}) = \min\left(Q,\ [q(t_i) + R(t_i, t_{i+1}) - (C_l(t_i, t_{i+1}) - n_{fg}(t_{i+1})) + q_{fg}(t_{i+1})]^+\right), \tag{6.3}$$

$$L(t_{i+1}) = L(t_i) + [q(t_i) + R(t_i, t_{i+1}) - (C_l(t_i, t_{i+1}) - n_{fg}(t_{i+1})) - (Q - q_{fg}(t_{i+1}))]^+ . \tag{6.4}$$

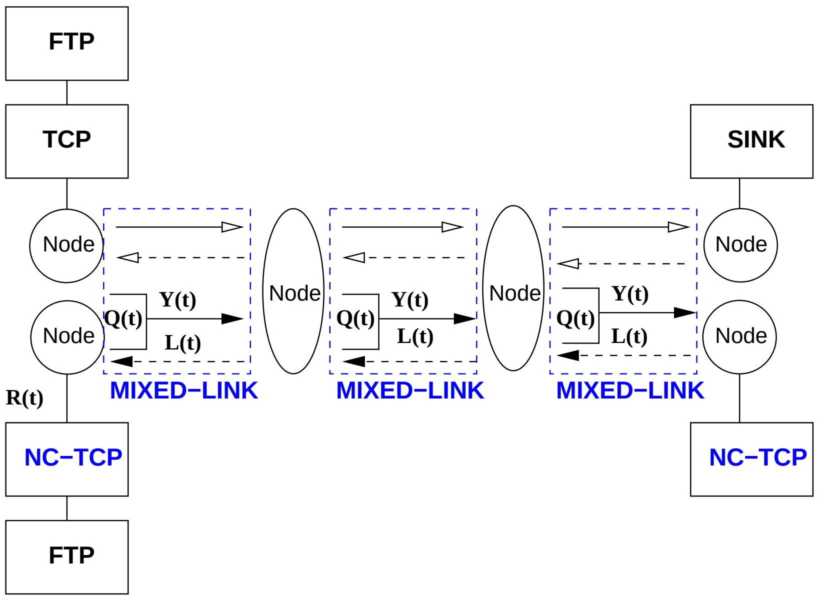


Figure 6.2: Network components implemented for mixed mode simulation.

6.2 Simulation Study

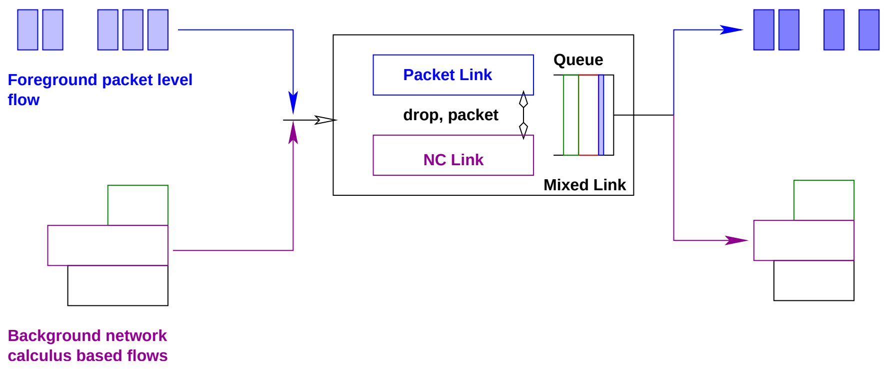
In this section, we discuss how we implement mixed mode simulation, and evaluate it in terms of execution time and error discrepancy.

6.2.1 Implementation

According to Section 6.1 and [29], we implement mixed mode simulation in *ns-2* as shown in Figure 6.2. In the figure, there are *NC-TCP*, *MIXED-LINK*, and existing modules such as FTP module and Node. The upper portion in Figure 6.2 shows packet mode simulation and the lower portion presents the network calculus based simulation.

A *NC-TCP* module at the sending side performs the following tasks: it (i) computes a round trip time on the basis of the loss and delay information fed back by the receiving *NC-TCP* module, and (ii) determines the amount of traffic to be transmitted, and then sends the information of the traffic amount via a *nc-tcp* packet. The detailed explanation is referred to [29].

MIXED-LINK consists of two submodules: *NC-LINK* submodule and *Packet-LINK* submodule and its graphical representation is shown in Figure 6.3. *NC-LINK* submodules are used to deliver NC-based flows, and determine the output, loss, and queue function using Eqs. (6.2), (6.4), and (6.3), respectively. *Packet-LINK* submodule serves foreground flows based on the model of



Foreground packet level flow
Packet Link
Queue
drop, packet
NC Link
Mixed Link
Background network calculus based flows

Figure 6.3: Mixed mode link for mixed mode simulation.

interaction specified in Section 6.1.

6.2.2 Simulation

We have conducted a simulation study to evaluate mixed mode simulation and compare its performance in terms of execution time and error discrepancy in a wide variety of network topologies, compared to packet mode simulation. The packet size is set to 1000 bytes, *TCP Reno* and *TCP Tahoe* are used for both packet mode and mixed mode simulation. Due to space limit, in what follows we only present the results in case of TCP Reno.

All the experiments are conducted in Linux 2.4.18 on a Pentium 4/1.9 *Ghz* PC with 1 *GBytes* memory and with 2 *GBytes* swap memory, and *ns-2.1b9a* is chosen as the underlying simulator.

Performance of mixed mode Simulation w.r.t. Error Discrepancy: To begin with, we examine whether or not one foreground flow in mixed mode simulation under-utilizes or over-utilizes network bandwidth, competing with other NC-based flows. For the purpose, we compare the throughput that the foreground flow achieves in mixed mode simulation with (i) the throughput range which is measured in packet mode simulation (where all the per-flow throughputs exist), i.e. the maximum, the minimum, and the average throughput of flows in packet mode simulation, and (ii) the average throughput of the other NC-based flows in mixed mode simulation.

The first simulation topology is a simple dumbbell network structure in Figure 6.4 with the varying number of flows (nodes) where each router has the buffer of 50 packets length. As marked in the figure, one flow represents one packet flow which really transmits a sequence of TCP packets. Figure 6.5 gives the relative throughput to the bottleneck link capacity every 100 seconds in the system in a 1000-second simulation under both packet mode and mixed mode simulation, when the

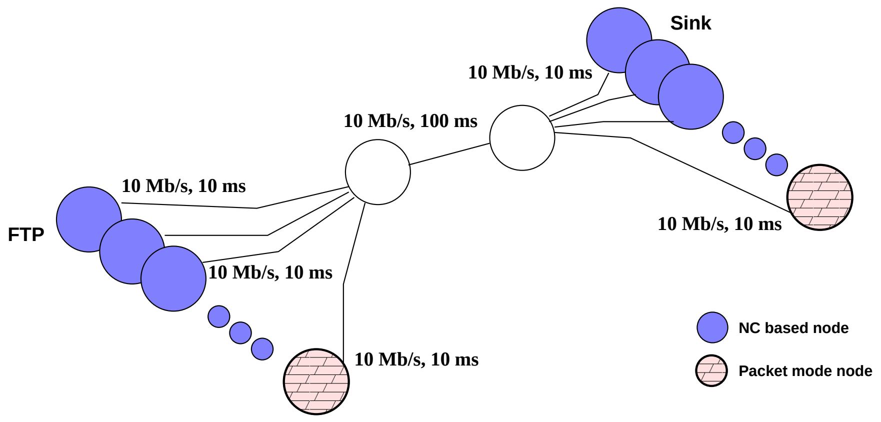
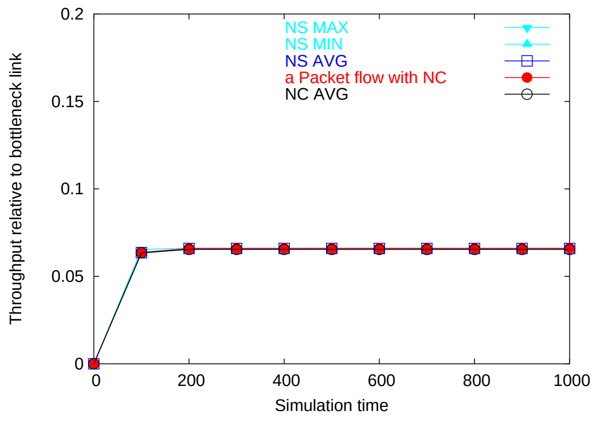


Figure 6.4: The network configuration used in the first set of experiments.

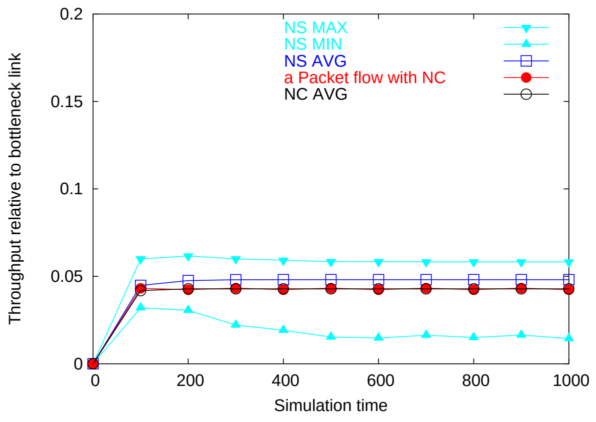
number of nodes is 10, 20, 40, and 60, respectively, and the bandwidth of the bottleneck link is 10 Mb/s in Figure 6.4. Figure 6.5 firstly presents the throughput that one foreground flow achieves in mixed mode simulation. We simply call the throughput *foreground throughput*. Additionally, it shows its comparison with the maximum, the minimum, the average throughput of all the flows in packet mode simulation, and also with the average throughput of the other NC-based flows in mixed mode simulation.

As shown in Figure 6.5, the throughput of the foreground flow oscillates in the range between the maximum and the minimum throughput in packet mode simulation during every 100 seconds. In some cases, the throughput is deviated from the range, but the error discrepancy is negligible, compared with the bottleneck link capacity. Specifically, the error discrepancy in the results of the foreground throughput against both (i) the achievable throughput range of packet mode flows and (ii) the average throughput of the other NC-based flows falls within 1 % of the bottleneck link capacity. Therefore, mixed mode simulation correctly provide a bandwidth environment for the foreground flow by approximating the other flows with NC-based flows, so that the foreground flow accurately mimics one packet mode flow in packet mode simulation in perspective of performance.

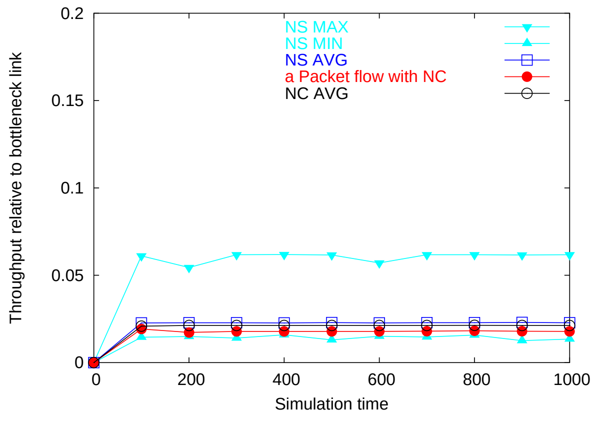
To verify whether or not mixed mode simulation still produces high-fidelity simulation results in more complicated network configuration, we repeat the same experiments as we increasingly make network configuration complicated. The second network configuration is in Figure 6.6 where buffer size at each link is 50 packets. Figure 6.7 gives the relative throughput to the bottleneck link capacity every 100 seconds in 1000-second simulation under both packet mode and mixed mode



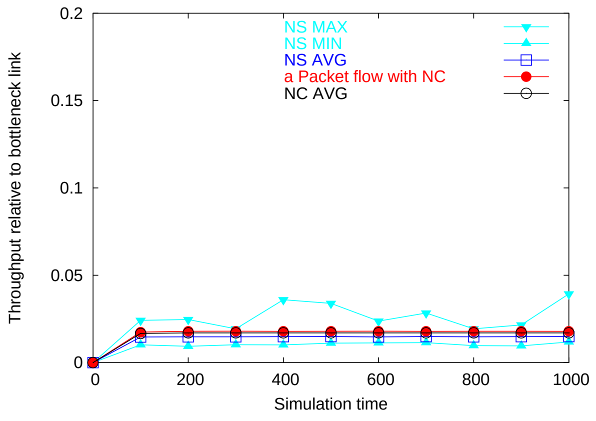
(a) The number of nodes is 10



(b) The number of nodes is 20



(c) The number of nodes is 40



(d) The number of nodes is 60

Figure 6.5: The relative throughput to the bottleneck link capacity that one foreground packet flow achieves every 100 seconds in mixed mode simulation, compared with the maximum, the minimum, and the average throughput of all the flows in packet mode simulation, and also with the average throughput of the other NC-based flows in mixed mode simulation. The network configuration is given in Figure 6.4, where the number of nodes is 10, 20, 40, and 60, respectively.

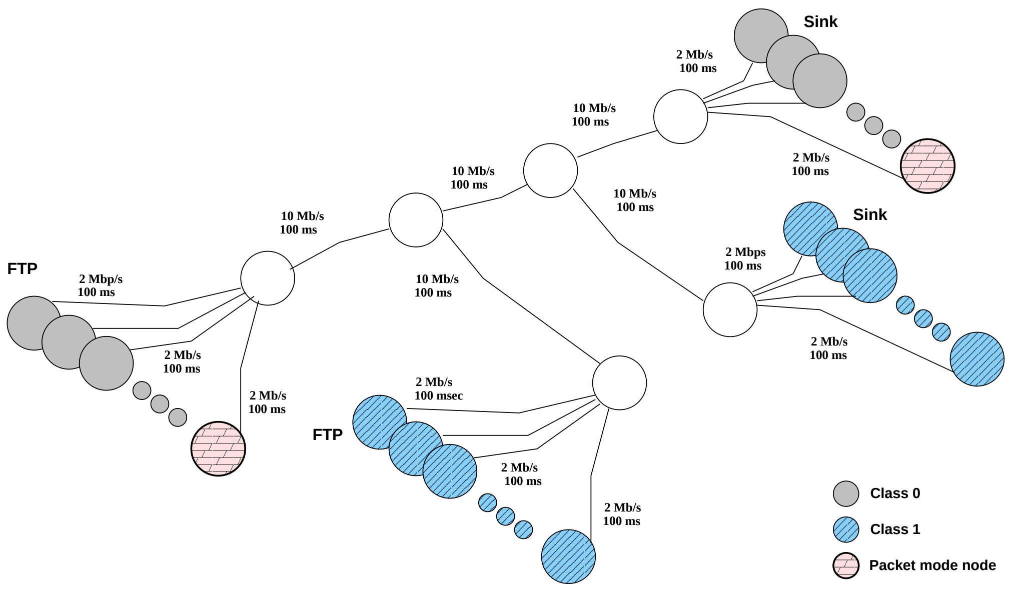
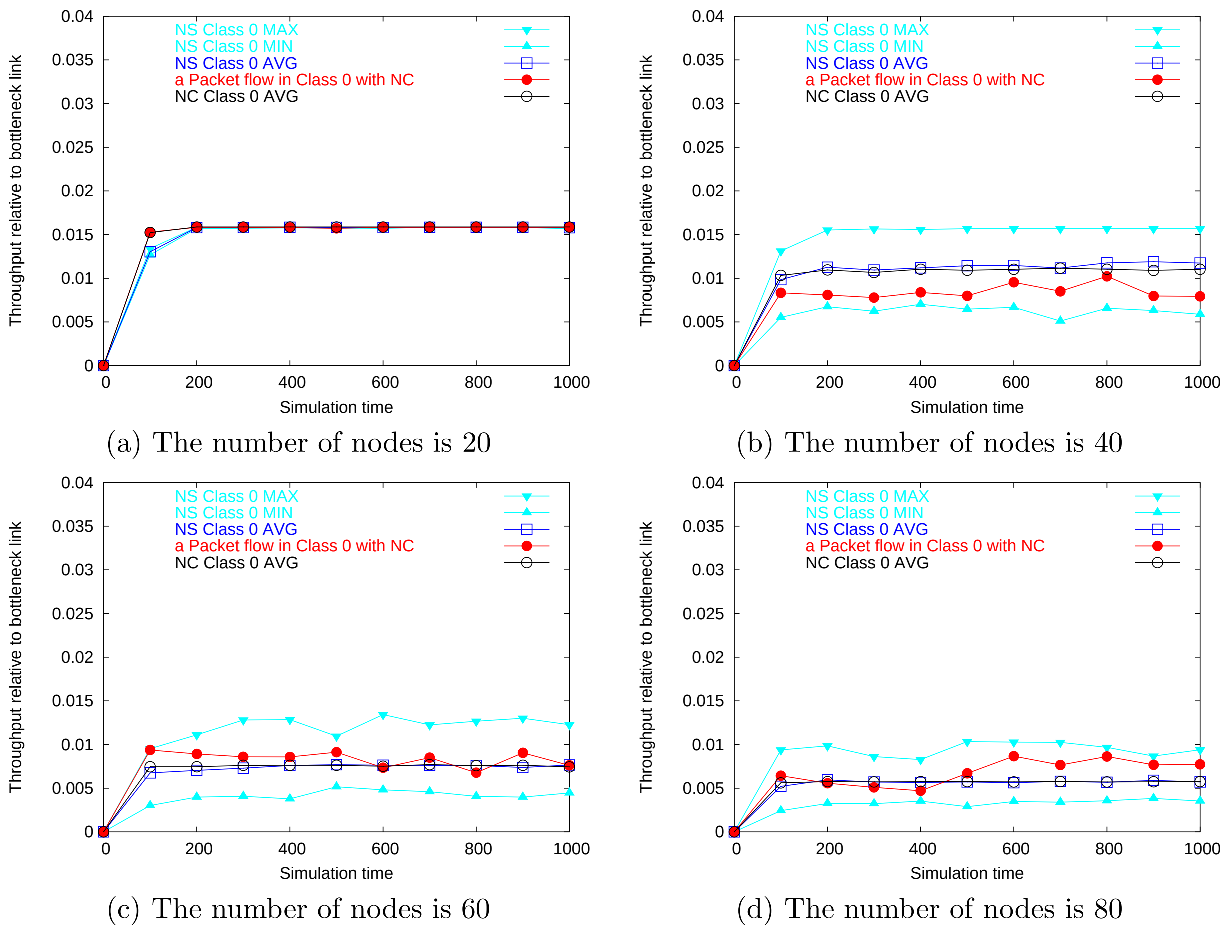


Figure 6.6: The network configuration used in the second set of experiments.

simulation, when the number of nodes is 20, 40, 60, and 80, respectively, and the bandwidth of the bottleneck link is 10 Mb/s in Figure 6.6. Figure 6.7 presents the foreground throughput in mixed mode simulation, and then shows its comparison with the maximum, the minimum, the average throughput of all class 0 flows in packet mode simulation, and also with the average throughput of the other NC-based flows in the same class (class 0) in mixed mode simulation.

As shown in Figure 6.5, the same observation is made: the error discrepancy in the results of the foreground throughput, compared to both the achievable throughput range of class 0 flows in packet mode simulation and the average throughput of the other class 0 NC-based flows, falls within less than 1 % of the bottleneck link capacity; and therefore the foreground flow does not over-utilize nor under-utilize the link bandwidth in mixed mode simulation.

The last network configuration is in Figure 6.8, in which the buffer length at each router is still 50 packets. Figure 6.9 gives the relative throughput to the bottleneck link capacity every 100 seconds in a 1000-second simulation under both packet mode and mixed mode simulation, when the number of nodes is 20, 40, 60, and 80, respectively, and the bandwidth of the bottleneck link is 10 Mb/s in Figure 6.6. Figure 6.9 presents the foreground throughput in mixed mode simulation achieves, and additionally it shows its comparison with the maximum, the minimum, and the average throughput of all class 0 flows in packet mode simulation, and also with the average throughput of the other class 0 NC-based flows in mixed mode simulation.



(a) The number of nodes is 20

(b) The number of nodes is 40

(c) The number of nodes is 60

(d) The number of nodes is 80

Figure 6.7: The relative throughput to the bottleneck link capacity that one foreground packet flow achieves every 100 seconds in mixed mode simulation, compared with the maximum, the minimum, and the average throughput of class 0 flows in packet mode simulation, and also with the average throughput of the other class 0 NC-based flows in mixed mode simulation. The network configuration is given in Figure 6.4, where the number of nodes is 20, 40, 60, and 80, respectively.

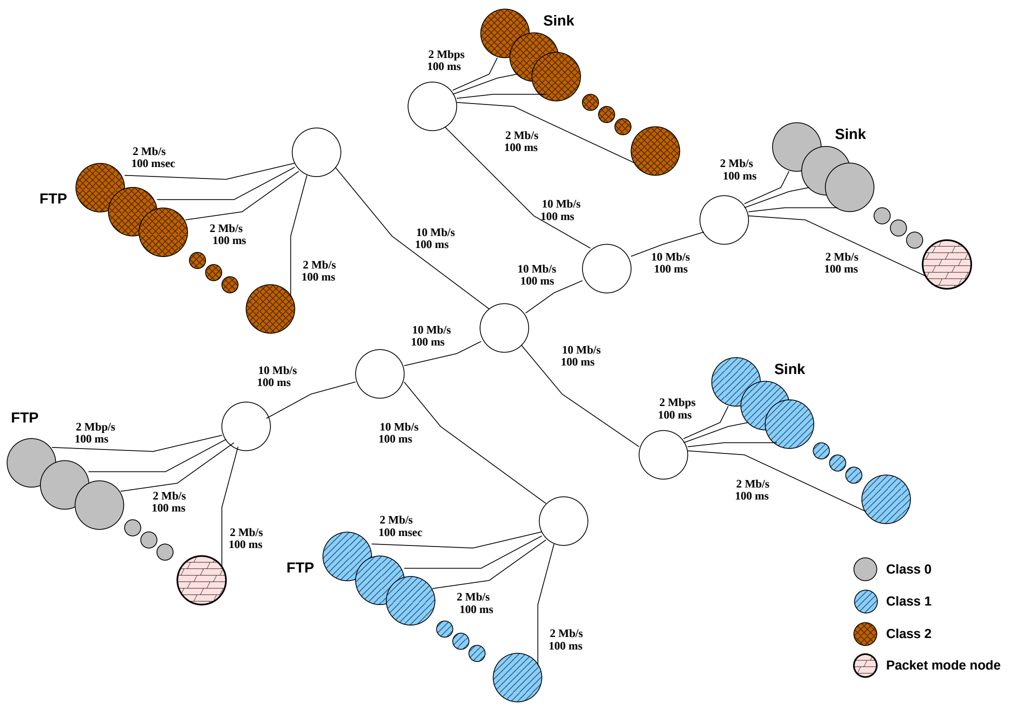
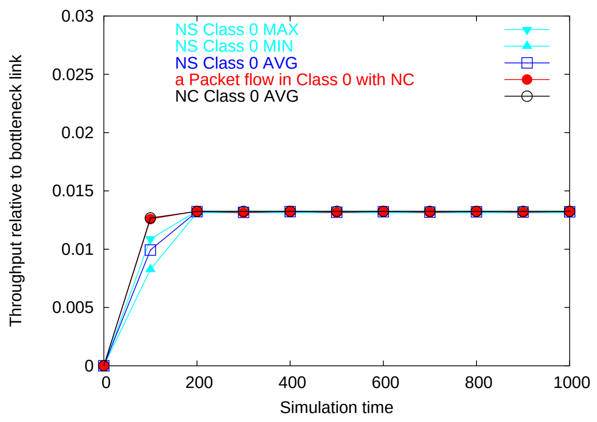


Figure 6.8: The network configuration used in the third set of experiments.

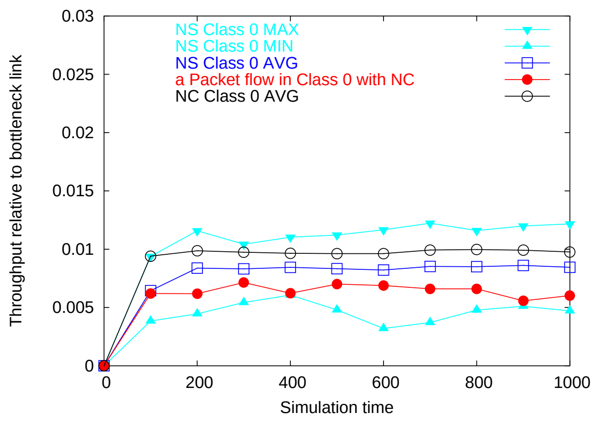
As shown in Figures 6.5 and 6.7, the similar trend is observed: the error discrepancy in the results of the foreground throughput from both the achievable throughput range in packet mode simulation and the average throughput of the other NC-based flows falls within about 1 % of the bottleneck link capacity, so that the foreground flow in mixed simulation correctly represents one packet flow in perspective of throughput.

In addition to the throughput dynamics every 100 seconds, we also investigate the aggregate throughput measured in 1000-second run in mixed mode simulation. Figures 6.10– 6.12 compare the aggregate foreground throughput with the maximum, the minimum, the average throughput of all the flows in the same class in packet mode simulations, and also with the average throughput of the other NC-based flows in the same class in mixed mode simulation.

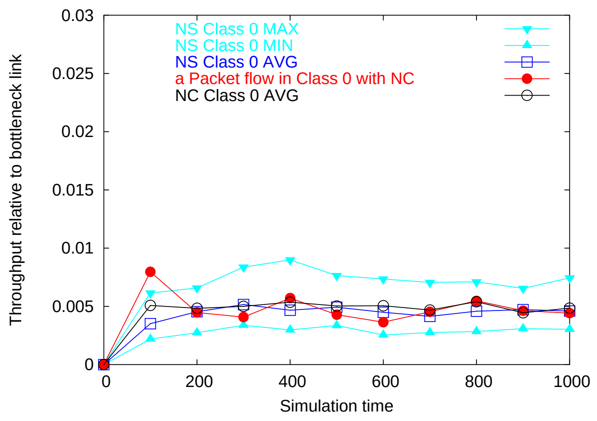
Figure 6.10 presents the relative throughput that the foreground flow achieves during 1000 seconds in Figure 6.4, and compares it with other four (4) throughputs: the maximum, the minimum, and the average throughput of all the flows in packet mode simulation, and also the average throughput of the other NC-based flows in mixed mode simulation. As the errors are accumulated during 1000 seconds, the error discrepancy gets a little slightly larger than the one in Figure 6.5, but still falls within 1 % in almost cases.



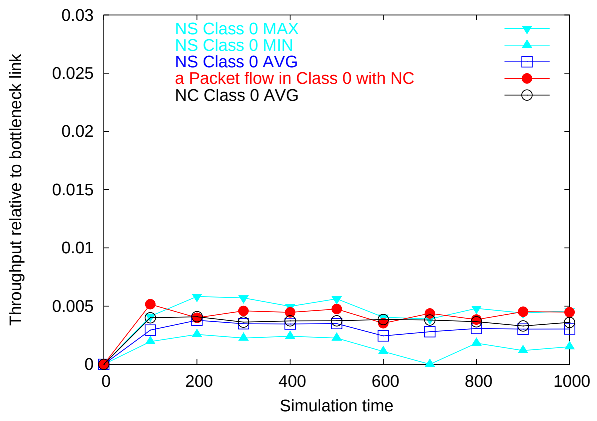
(a) The number of nodes is 20



(b) The number of nodes is 40



(c) The number of nodes is 60



(d) The number of nodes is 80

Figure 6.9: The relative throughput to the bottleneck link capacity that one foreground packet flow achieves every 100 seconds in mixed mode simulation, compared with the maximum, the minimum, and the average throughput of class 0 flows in packet mode simulation, and also with the average throughput of the other class 0 NC-based flows in mixed mode simulation. The network configuration is given in Figure 6.4, where the number of nodes is 20, 40, 60, and 80, respectively.

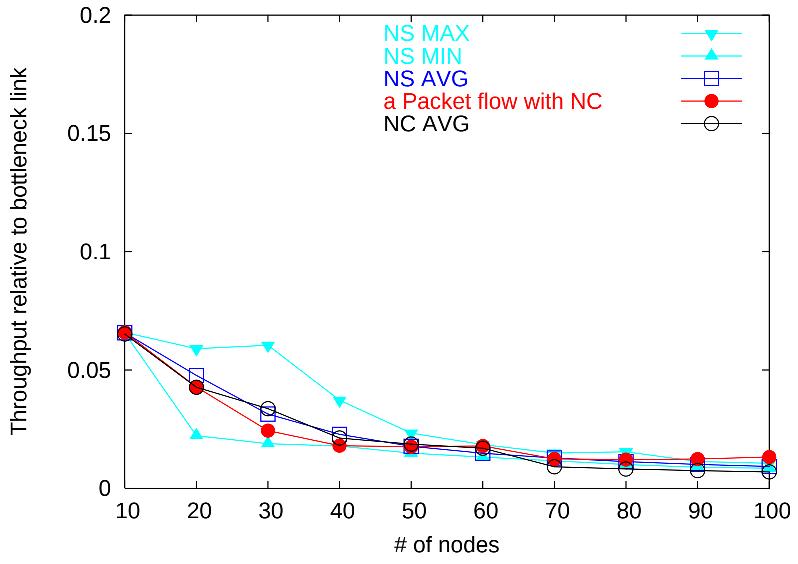


Figure 6.10: The throughput (relative to bottleneck link capacity) that one foreground flow achieves during 1000-second run in mixed mode simulation when the number of nodes varies in the network in Figure 6.4. The throughput is compared with the maximum, the minimum, and the average throughput of all the flows in packet mode simulation, and also with the average throughput of the other NC-based flows in mixed mode simulation.

Figure 6.11 conducts the similar throughput comparison in the network of Figure 6.6. The error discrepancy is still confined within 1 % for the most time.

Figure 6.12 also shows the same trendy in the network of Figs 6.8. The error discrepancy still falls within 1 % in the most cases. In some cases, the foreground throughput gets a little larger than the maximum throughput measured in packet mode simulation, but the error is small enough to be ignored, compared to the bottleneck link capacity.

Conclusively, from Figures 6.10–6.12, we can also observe that the foreground flow in mixed mode simulation shows a similar behavior in perspective of throughput.

Performance of mixed mode Simulation w.r.t. Execution Time: After we verify that mixed mode simulation generates acceptably small error, we study to what extent mixed mode simulation save execution time in network simulation for TCP-operated networks.

Figure 6.13 shows the execution time taken to carry out 1000-second simulation versus the number of nodes under both packet mode and mixed mode simulation in the network configuration shown in Figure 6.4. Figures 6.14–6.15 presents the execution time taken to carry out 1000-second simulation versus the number of nodes per class in the network configuration given in Figure 6.6

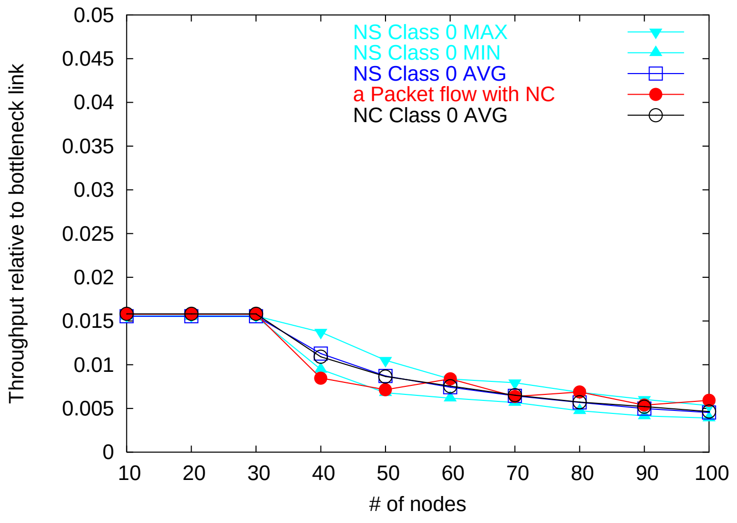


Figure 6.11: The throughput (relative to bottleneck link capacity) that one foreground flow achieves during 100-second run in mixed mode simulation when the number of nodes varies in the network in Figure 6.6. The throughput is compared with the maximum, the minimum, and the average throughput of all class 0 flows in packet mode simulation, and also with the average throughput of the other class 0 NC-based flows in mixed mode simulation.

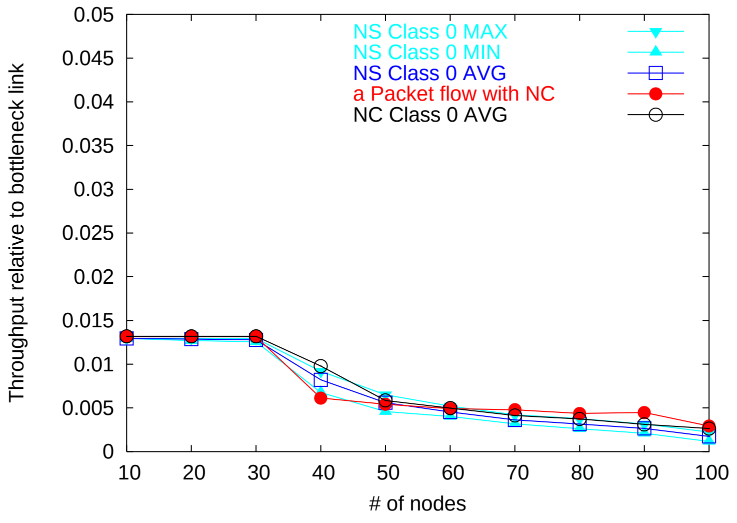


Figure 6.12: The throughput (relative to bottleneck link capacity) that one foreground flow achieves during 1000-second run in mixed mode simulation when the number of nodes varies in the network in Figure 6.8. The throughput is compared with the maximum, the minimum, and the average throughput of class 0 flows in packet mode simulation, and also with the average throughput of the other class 0 NC-based flows in mixed mode simulation.

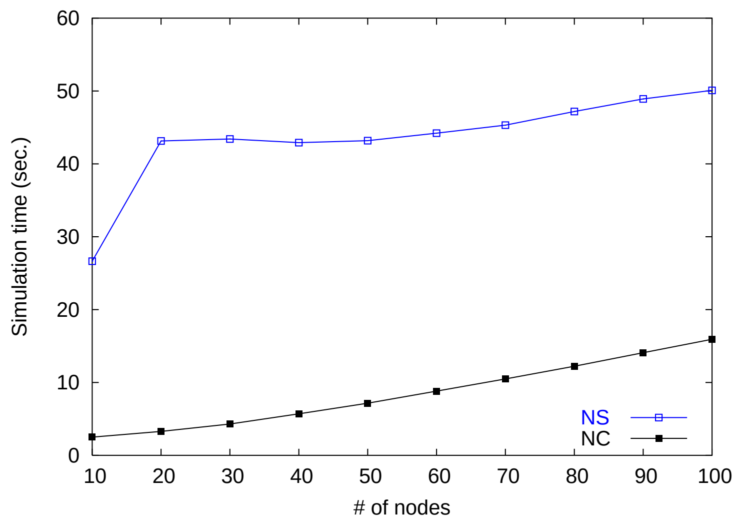


Figure 6.13: The execution time taken to carry out 1000-second simulation versus the number of nodes, under both packet mode and mixed mode simulation in the network configuration in Figure 6.4.

and 6.8, respectively.

As shown in Figures 6.13–6.15, the execution time incurred in mixed mode simulation is maximally 10 times less than that in packet mode simulation. (The speed-up can be further improved with a larger time step value.)

Effects of the Time Step Value: As NC-based flows in mixed mode simulation operates in time-stepped manner, we need to investigate how the time step value affects the performance in mixed based simulation. Thus, we carry out simulation with the varying time step value T.

Tables 6.1–6.3 give the results with varying time step values for the network configurations given in Figures6.4, 6.6, and 6.8, respectively. In all the configurations, the buffer size at each router is set to 50 packets.

As shown in the tables, the performance of mixed mode simulation can be improved (up to about 20 times better than that of packet mode simulation) as the time step value increases. Note that the error discrepancy does not always increase with the time step value, and the execution time cannot be unlimitedly reduced by increasing the time step value. This is due to the facts that in the underlying implementation of network calculus based simulation implementation: (i) sometimes the error that results from the computation optimization in the matrix computation is larger than that incurred in the choice of larger time step values, and thus it dominates the total

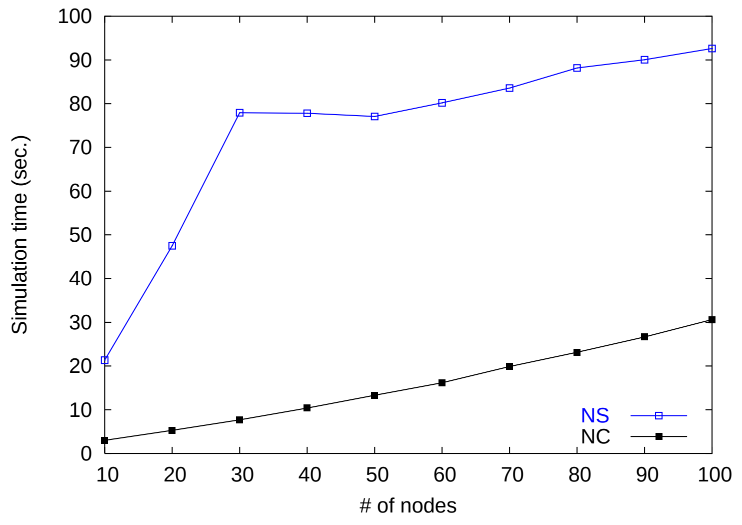


Figure 6.14: The execution time taken to carry out 1000-second simulation versus the number of nodes per class, under both packet mode and mixed mode simulation in the network configuration in Figure 6.6.

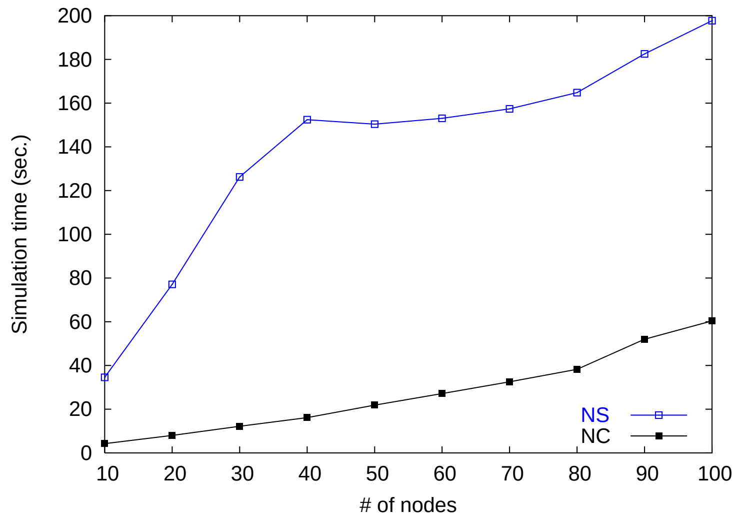


Figure 6.15: The execution time taken to carry out 1000-second simulation versus the number of nodes per class, under both packet mode and mixed mode simulation in the network configuration in Figure 6.8.

Table 6.1: The number of packets and execution time (s) that that the foreground flow sent and consumed in a 1000-second simulation run in mixed mode simulation, with the network configuration in Figure 6.4 when the number of nodes is 20. The number of packets is compared with the maximum, the minimum, and the average number of all the flows in packet mode simulation, and also with the average value of the other NC-based flows in mixed mode simulation.

Time step	packet mode simulation				mixed mode simulation		
value (s)	max	min	avg	time	foreground	avg	time
	73755	27807	59621	44.84			
0.5					34649	53460	8.71
1.0					42803	63775	6.34
2.0					53487	53419	3.48
4.0					54144	51648	2.57

Table 6.2: The number of packets and execution time (s) that that the foreground flow sent and consumed in a 1000-second simulation run in mixed mode simulation, with the network configuration in Figure 6.6 when the number of nodes is 40. The number of packets is compared with the maximum, the minimum, and the average number of packets of all class 0 flows in packet mode simulation, and also with the average value of the other class 0 NC-based flows in mixed mode simulation.

Time step	packet mode simulation				mixed mode simulation		
value (s)	max	min	avg	time	foreground	avg	time
	17159	11829	14090	82.54			
1.0					10754	15288	17.82
2.0					10598	13631	10.86
4.0					14252	12717	7.74

error; in other words, the error does not fully depends on time step values, and (ii) since the number of losses increases with the increase in the time step value and the performance of optimization depends on the number of losses, the optimization algorithm consumes more computation time as the number of losses increases. Note that both causes affect also the foreground flow since NC-based flows dynamically influence on the flow.

6.3 Analysis in Worst Case

In this section, we conduct an analysis for worst case error bounds which might be observed on the simulation results in the previous section.

Given an equation for the dropping probability, p_l, we can describe the instantaneous change

Table 6.3: The number of packets and execution time (s) that that the foreground flow sent and consumed in a 1000-second simulation run in mixed mode simulation, with the network configuration in Figure 6.8 when the number of nodes is 50. The number of packets are compared with the maximum, the minimum, and the average number of packets of all class 0 flows in packet mode simulation, and also with the average value of the other class 0 NC-based flows in mixed mode simulation.

Time step	packet mode simulation				mixed mode simulation		
value (s)	max	min	avg	time	foreground	avg	time
	8120	5734	6990	156.21			
1.0					5299	6939	35.55
2.0					6765	7309	22.87
4.0					9127	6926	16.76

in the rate with the fluid model in [26, 42]:

$$\dot{r_i}(t) = \alpha\frac{1}{t_{R_i}^2} - \beta r_i(t - t_{R_i}) \cdot p_l(r_i(t - t_{R_i})) \tag{6.5}$$

where $\dot{r_i}(t)$ is the rate change of a flow i passing through bottleneck link l, t_{R_i} being the round trip time for the flow i, p_l is the loss rate function which is a function of all flows, and α and β are an additive increase parameter and a multiplicative decrease parameter, respectively.

If we change the above rate equation (fluid model) into the equation for the dynamics of window size with $r_i(t) = w_i(t)/t_{R_i}$, we obtain the following:

$$\dot{w_i}(t) = \alpha\frac{1}{t_{R_i}} - \beta w_i(t - t_{R_i}) \cdot p_l\left(\frac{w_i(t - t_{R_i})}{t_{R_i}}\right). \tag{6.6}$$

Therefore, we need to determine the dropping probability function, $p_l(\cdot)$, in order to decide the throughput and the window dynamics of one foreground flow, and consequently we can investigate the difference between two throughputs, each of which is obtained by the flow in packet mode simulation and mixed mode simulation respectively.

In the case of DropTail, dropping probability is approximately determined as follows [25, 42],

$$p_l(t) = \frac{\left(\sum_{i=1}^{N'} r_i(t) - C_l\right)^+}{\sum_{i=1}^{N'} r_i(t)}, \tag{6.7}$$

where $N' = (N + 1)$ denotes the total number of flows, one of which is one foreground flow while the other N number of flows are NC-based background flows, sharing link l.

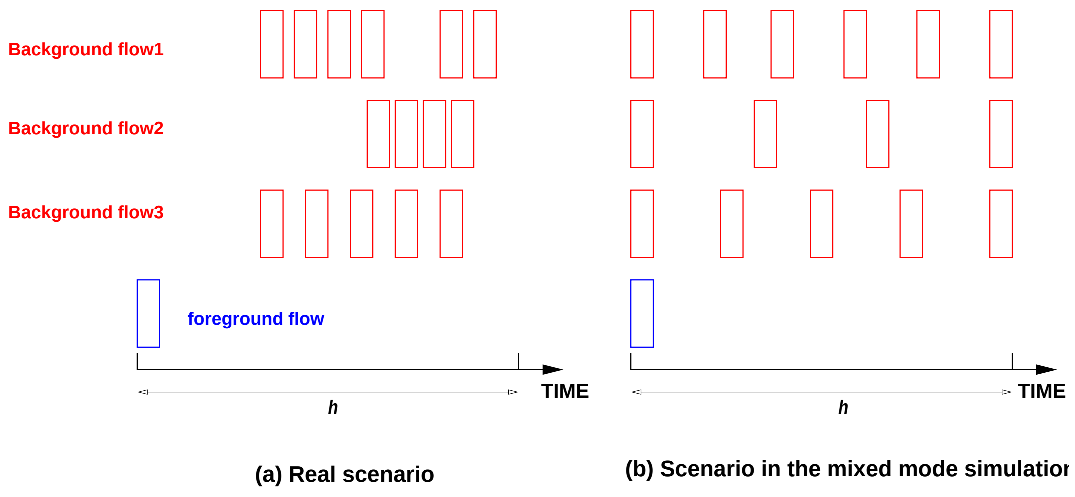


Figure 6.16: The worst scenario for the throughput for one foreground packet flows.

The biggest difference in two throughputs is perceived when the instantaneous queue length at the time t (when one new packet arrives at the link) is overestimated due to the increased rate of the NC-based flows by assuming that all the packets belong to NC-based flows are evenly spaced during a time interval (time step) [29]; for example, even if a buffer has an ample space to accommodate the newly arrived foreground packet in packet mode simulation, mixed mode simulation might decide there is no room for the packet and then decides to discard it. Specifically, as shown in Eq. (6.7), dropping probability in DropTail depends on the aggregate input rate. The problem is that mixed mode simulation might increase the current input rate, $\sum_{i=1}^{N'} r_i(t)$, by the assumed traffic amount (or rate) of the NC-based flows (the amount is proportional to the time step). Therefore, the worst case of the throughput difference happens when a dropping occurs in mixed mode simulation when a foreground packet arrives, even if any background flow does not cause any dropping in packet mode simulation.

Figure 6.16 represents one of the extreme cases; Figure 6.16 (a) presents the real scenario in which the foreground flow (the flow at the lowest position of the figure) has the first packet, p_{first}, among all the packets during an interval h. On the contrary, Figure 6.16 (b) shows another scenario made by mixed mode simulation in which the first packet p_{first} competes other three packets belong to background flows.

In this worst case, the dropping function in the real situation is still Eq. (6.7), but the function in mixed mode simulation is,

$$
\begin{aligned}
\widehat{p_l}(t) &= \frac{\left(\sum_{i=1}^{N'} r_i(t) + \frac{\sum_{i\in I_m} m_i}{h} - C_l\right)^+}{\sum_{i=1}^{N'} r_i(t)} \\
&\leq \frac{\left(\sum_{i=1}^{N'} r_i(t) + \frac{\sum_{i=1}^{N} m_i}{h} - C_l\right)^+}{\sum_{i=1}^{N'} r_i(t)}, \qquad (6.8)
\end{aligned}
$$

where m_i represents the number of packets sent by flow i during h, and I_m denotes a set of flows that do not have any packet which arrives at the time when the first packet in the foreground flow arrives but come to send their packet in mixed mode simulation.

Therefore, the relationship between p and $\widehat{p}$ is

$$\widehat{p_l}(t) \leq p_l(t) + \frac{\frac{\sum_{i=1}^{N} m_i}{h}}{\sum_{i=1}^{N'} r_i(t)}.$$

Let the focus return to the throughput window dynamics in Eq. (6.6) in order to see the throughput difference. Instead of general dropping function in Eq. (6.6), we can use the dropping function in Eq. (6.7) for this DropTail case as follows

$$\frac{\triangle w_i}{\triangle t} = \alpha \frac{1}{t_{R_i}} - \beta w_i(t - t_{R_i}) \cdot p_l \left(\frac{w_i(t - t_{R_i})}{t_{R_i}} \right). \qquad (6.9)$$

As for mixed mode simulation, the instantaneous throughput fluctuation in perspective of window size with the dropping function in Eq. (6.8) is described as

$$\frac{\triangle \widehat{w}_i}{\triangle t} \geq \alpha \frac{1}{t_{R_i}} - \beta w_i(t - t_{R_i}) \cdot \left(p_l(t - t_{R_i}) + \frac{\frac{\sum_{i=1}^{N} m_i}{h}}{\sum_{i=1}^{N'} \frac{w_i(t - t_{R_i})}{t_{R_i}}} \right). \qquad (6.10)$$

Comparing Eq. (6.9) and (6.10), we finally obtain

$$\frac{\triangle w_i}{\triangle t} - \frac{\triangle \widehat{w}_i}{\triangle t} \leq \beta w_i(t - t_{R_i}) \cdot \frac{\frac{\sum_{i=1}^{N} m_i}{h}}{\sum_{i=1}^{N'} \frac{w_i(t - t_{R_i})}{t_{R_i}}} \qquad (6.11)$$

From Eq. (6.11), we can see that the error discrepancy caused in mixed mode simulation depends on the additional dropping probability, which comes from an incorrect estimate about the traffic amount of NC-based flows arrived up to now.

Chapter 7

Rescaling Simulation Model for Large Scale IP Networks

In this chapter, we present a rescaling simulation methodology (RSM) to expedite simulation in large scale IP networks without loss of fidelity of simulation results. Conceptually, we scale down the network to be simulated in order to save the number of events, simulate the downscaled network for a short period of time, and then extrapolate the corresponding results for the original network by scaling up the simulation results obtained from the downscaled network. Both the operations of scaling down and rescaling up the network are conducted in such a manner that the network invariant, called the *bandwidth-delay* product, is preserved. In particular, since the dynamics of queues, such as the queue size, the dropping probability, and other parameters at every link are the same in both the original and the downscaled network, RSM can accurately infer the network dynamics behavior of the original network (equipped with various Active Queue Management (AQM) strategies). Figure 7.1 coarsely explains this underlying idea of the proposed rescaling simulation methodology. Now the key issue is how to preserve the properties that characterize the original network in the downscaled network so that accurate results can be extrapolated after the packet mode simulation. In particular, the network property of interest should remain invariant in the process of scaling down and rescaling up the network.

For the purpose of simulating large scale IP networks, we use the *bandwidth-delay* product as the network property to be preserved, as it represents the capacity of the "pipe," i.e., the amount of data packets that can be transmitted without waiting for the acknowledgment [41]. That is, we scale down the original network by reducing the link capacity by a fraction α ($0 < \alpha \leq 1$), increasing the link delay by $\frac{1}{\alpha}$, but keeping the bandwidth-delay product constant. (Note that we change neither the number of nodes/flows nor the queue. (the maximum buffer size or the

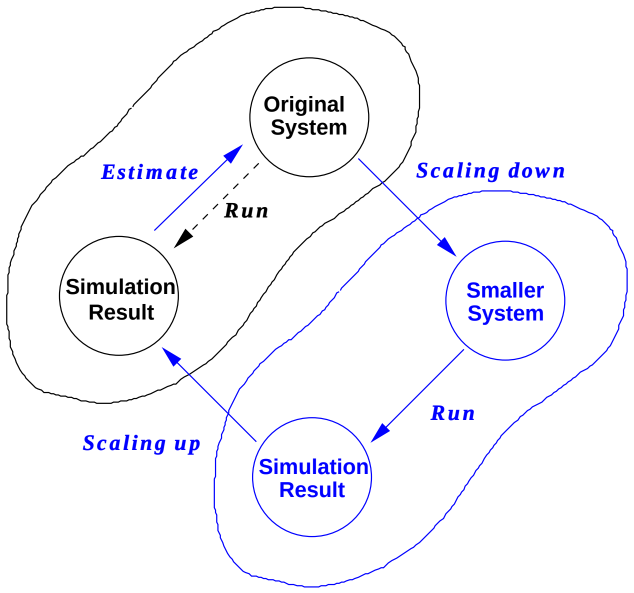


Figure 7.1: The underlying concept for the rescaling simulation methodology

AQM parameters.)) By preserving this product invariant during the down/up scaling operations, the network capacity as perceived by each TCP connection is preserved, and we can formally prove that the queue dynamics (e.g., the instantaneous queue length), the RTT dynamics, the TCP window size dynamics, and even the effect of TCP-friendly flows remain unchanged in the operations. Additionally, we will show that the event times, which change the dynamics (such as the number of flows or their bandwidth) in network, do not need to be adjusted in the course of scaling-down and scaling-up, even if the down-scaling delays all the events by $\frac{1}{\alpha}$. The proposed RSM based simulation focuses on the α portion of each time period, which is defined by two consecutive network events, then simulates those portion of period with the downscaled network (which takes shorter time), and finally extrapolates the results for the whole simulation period from the obtained results (The formal description and its proof appears in Section 7.1 and Figure 7.2.) By repeating this procedure each period until the end of simulation, the RSM based simulation can estimates all the simulation results, which consists of both queue dynamics and throughput.

7.1 Rescaling Simulation Methodology

In this section, we present the rescaling simulation methodology. The key idea of the methodology is to preserve the network capacity (as determined by the queue dynamics) during the down/up scaling operation. We first discuss the network invariant in simulating IP networks. Then we introduce the rescaling simulation model based on the network invariant.

7.1.1 Network Invariant

Our objective is to scale down a large scale network to a small one that can be simulated to produce sufficient results in a short time interval to infer the performance for the original network. The down scaling operation expedites network simulation by reducing the number of events generated/processed in the simulation. For example, if we reduce the link capacity at each link, we can reduce the number of sending and receiving events per unit time at the link. Similarly, if we increase the propagation delay at each link, we can reduce the sending rate of each TCP connection (as a result of increased round trip time.)

One important issue is then how to scale down the network so that the simulation results obtained from the down-scaled network can be used to accurately infer those corresponding to the original network. We claim that the down scaling operation should not change the network capacity as perceived by TCP connections. Note that the network capacity is determined by the network dynamics (such as the instantaneous queue length and dropping probability) and is reflected upon the throughput TCP connections can attain. Hence, we use the *bandwidth-delay* product as the network invariant to be preserved during the down/up scaling operations. Since the bandwidth-delay product represents the amount of data packets that can be in transit [41], the product can be regarded as the TCP window size in the perspective of a TCP connection.

Specifically, let B and D denote, respectively, the available bandwidth and delay along a path in the original network, and P_{BDP} the bandwidth-delay product along the path. For notational convenience, we denote the corresponding variables in the down-scaled network by attaching an apostrophe to the variables, i.e., B', D', and P'_{BDP}. The following constraint is used in the down/up scaling operations:

$$P_{BDP} = B \cdot D = B' \cdot D' = P'_{BDP}. \tag{7.1}$$

By Eq. (7.1), a scaling parameter, α, is determined as follows:

$$\begin{aligned} B' &= \alpha \cdot B, && (7.2) \\ D' &= \frac{D}{\alpha}, && (7.3) \end{aligned}$$

where $0 < \alpha \leq 1$.

Note that we do not change the number of nodes, the number of flows, or the queue-related parameters (e.g., the maximum buffer size or the AQM parameters). The only parameters that are scaled are the link capacity and the link delay. As will be formally proved in Section 7.1.2, by keeping P_{BDP} invariant, the queue dynamics (e.g., the instantaneous queue length), the RTT dynamics, the window size dynamics, and the effect of TCP-friendliness remain unchanged in the operations. As a result, the down-scaled network exhibits exactly the same behavior as the original network.

7.1.2 Rescaling Simulation Model

We now propose the rescaling simulation model. Let N denote the number of flows sharing a path with the bottleneck link of capacity C. Let $q(t)$ and $p(t)$ denote, respectively, the queue length and the packet dropping probability of the bottleneck link at time t, and T the propagation delay of the path. Let $W_i(t)$ and $R_i(t)$ denote, respectively, the window size and the round trip time of flow i at time t. Then the interaction between TCP flows and the bottleneck link can be characterized with the TCP model given in [36]:

$$\begin{aligned} R_i(t) &= T + \frac{q(t)}{C}, && (7.4) \\ \frac{dq(t)}{dt} &= \sum_{i=1}^{N} \frac{W_i(t)}{R_i(\tau_i)} - C, && (7.5) \\ \frac{dW_i(t)}{dt} &= a \cdot \frac{1}{R_i(t)} - b \cdot W_i(t) \cdot \frac{W_i(\tau_i)}{R_i(\tau_i)} \cdot p(\tau_i), && (7.6) \end{aligned}$$

where a is an additive increase parameter, b is a multiplicative parameter, and $\tau_i = t_i - R_i(t)$.

The rescaling simulation model reduces the link bandwidth and increases the link delay by the scaling parameter, α (Eqs. (7.2) and (7.3)). Since each packet is served at the rate of $\alpha \cdot B$ (which stretches its transmission time by the factor of $\frac{1}{\alpha}$) and experiences $\frac{1}{\alpha}$ times larger delay, the time instants at which packet events occur are also delayed by $\frac{1}{\alpha}$. Specifically, a packet event that occurs at time t in the original network is now delayed to t' in the down-scaled network as follows:

$$t' = \frac{1}{\alpha} \cdot t, \tag{7.7}$$

where $t (t \geq 0)$ and $t' (t' \geq 0)$ denote, respectively, a time instant for the original network and for the down-scaled network.

In what follows, we will investigate the dynamics of the queue length, the round trip time, and the TCP window size in both the down-scaled network and the original network. Again we denote the corresponding variables in the down-scaled network by attaching an apostrophe to the variables.

Queue Dynamics

We will show that queue length of the bottleneck link in the original network is the same to that in the down-scaled network:

$$q'(t') = q(t).$$

This implies that the queue responds to each TCP connection in the down-scaled network in exactly the same manner (including the packet dropping probability) as in the original network. Moreover, this is achieved without modifying or approximating any AQM parameter in the down-scaled network. We first look at $\frac{dq'(t')}{dt}$:

$$\begin{aligned} \frac{dq'(t')}{dt} &= \sum_{i=1}^{N} \frac{P'_{i,BDP}(t')}{D'_i(t')} - C' = \sum_{i=1}^{N} \frac{P_{i,BDP}(t)}{\frac{D_i(t)}{\alpha}} - \alpha \cdot C \\ &= \alpha \cdot \left\{ \sum_{i=1}^{N} \frac{P_{i,BDP}(t)}{D_i(t)} - C \right\} = \alpha \cdot \frac{dq(t)}{dt} = \frac{dq(t)}{\frac{1}{\alpha} \cdot dt}. \end{aligned} \tag{7.8}$$

Note that in Eq. (7.8), the change in the queue size is simply the difference between the arrival rate of all flows and the link capacity, and the arrival rate of a flow i is obtained by dividing its current bandwidth-delay product ($P_{i,BDP}(t)$) by its current delay ($D_i(t)$).

Since $t' = \frac{1}{\alpha} \cdot t$ (Eq. (7.7)), $dt' = \frac{1}{\alpha} dt$, and thus,

$$\frac{dq'(t')}{dt} = \frac{dq(t)}{dt'},$$

or

$$dq'(t') \cdot dt' = dq(t) \cdot dt. \tag{7.9}$$

Since $q'(0) = q(0) = 0$ and Eq. (7.9) is a separable first order equation [10], we obtain the following result by integrating both sides of Eq. (7.9) when $t, t' \geq 0$:

$$q'(t') = q(t). \tag{7.10}$$

By Eq. (7.10), we know as long as the down scaling operation preserves the *bandwidth-delay* product of the original network, the queue dynamics in both the original network and the down-scaled network are the same at each event time.

RTT Dynamics

We can compute the round trip time in the down-scaled time domain, t', as follows. Since $R_i(t) = T + \frac{q(t)}{C}$,

$$\begin{aligned} R'_i(t') &= T' + \frac{q'(t')}{C'} = \frac{T}{\alpha} + \frac{q(t)}{\alpha \cdot C} = \frac{1}{\alpha} \cdot (T + \frac{q(t)}{C}) \\ &= \frac{1}{\alpha} R_i(t). \end{aligned} \tag{7.11}$$

Note that we have used $q'(t') = q(t)$ in the second equality in Eq. (7.11). With Eq. (7.11), we can now re-express $\frac{dq'(t')}{dt}$ in Eq. (7.8) as follows:

$$\begin{aligned} \frac{dq'(t')}{dt} &= \sum_{i=1}^{N} \frac{W'_i(t')}{R'_i(\tau'_i)} - C' = \sum_{i=1}^{N} \frac{W_i(t)}{\frac{R_i(\tau_i)}{\alpha}} - \alpha \cdot C \\ &= \alpha \cdot \left\{ \sum_{i=1}^{N} \frac{W_i(t)}{R_i(\tau_i)} - C \right\} = \alpha \cdot \frac{dq(t)}{dt} = \frac{dq(t)}{dt'} \end{aligned}$$

where the first equality results from the fact that the current TCP window size ($W_i(t)$) can be represented by the current bandwidth-delay product ($P_{i,BDP}(t)$) as perceived by each TCP connection i.

Window Size Dynamics

Eq. (7.11) implies that each TCP connection in the down-scaled network exhibits the same window dynamics but the response is $\frac{1}{\alpha}$ times slower than that in the original network (since a TCP connection in the down-scaled network adjusts its rate per round trip time $R'_i(t)$). To further verify this, we express the window dynamics in the down-scaled network based on Eq. (7.6) as follows:

$$\frac{dW_i'(t')}{dt'} = a \cdot \frac{1}{R_i'(t')} - b \cdot W_i'(t') \cdot \frac{W_i'(\tau_i')}{R_i'(\tau_i')} \cdot p(\tau_i'), \tag{7.12}$$

where $\tau_i' = t' - R_i'(t')$.

By plugging Eq. (7.11) into Eq. (7.12), we have

$$\begin{aligned}\frac{dW_i'(t')}{dt'} &= a \cdot \frac{1}{\frac{1}{\alpha} \cdot R_i(t)} - b \cdot W_i(t) \cdot \frac{W_i(\tau_i)}{\frac{1}{\alpha} \cdot R_i(\tau_i)} p(\tau_i) \\ &= \alpha \cdot \left\{ a \cdot \frac{1}{R_i(t)} - b \cdot W_i(t) \cdot \frac{W_i(\tau_i)}{R_i(\tau_i)} \cdot p(\tau_i) \right\} \\ &= \alpha \cdot \frac{dW_i(t)}{dt}, \end{aligned} \tag{7.13}$$

where $\tau_i = t - R_i(t)$. Note that in the first equality of Eq. (7.13), we use the current window size as the current bandwidth-delay product of the path, and hence $W_i'(x') = W_i(x)$, for $\forall x' = \frac{1}{\alpha} \cdot x$. Additionally, $\tau_i' = \frac{1}{\alpha} \cdot \tau_i$ since $R_i'(t') = \frac{1}{\alpha} \cdot R_i(t)$ (Eq. (7.11)), and $p(t) = p'(t')$ since $q'(t') = q(t)$ (Eq. (7.10)), for $\forall t' = \frac{1}{\alpha} \cdot t$.

As implied in Eqs. (7.11) and (7.13), all the events that occur in the original network are delayed by the factor of $\frac{1}{\alpha}$ in the down-scaled network. This is consistent with Eq. (7.7).

TCP Dynamics in the Slow Start Phase

In the TCP *slow start* phase, a TCP connection exponentially increases its TCP window until packet loss occurs. Let $W_{i,ss}(t)$ be the instantaneous window size at time t in the *slow start* phase. Then, $W_{i,ss}$ can be expressed as:

$$\begin{aligned} W_{i,ss}(0) &= 1, \text{ and} \\ W_{i,ss}(t) &= 2 \cdot W_{i,ss}(t - R_i(t)) = 2^{\left\lfloor \frac{t}{R_i(t)} \right\rfloor} \text{ for } t > 0. \end{aligned}$$

We first investigate the TCP window dynamics in the slow start phase. The TCP window size at time t depends on the number of successful transmissions until time t. We appropriate $R_i(t_i) \approx R_i(t_j)$ for all $0 < t_i \le t$ and $0 < t_j \le t$ since in the slow start phase no packet loss is incurred, and all the packets experience approximately the same queuing time (if any). The window dynamics in the down-scaled network can be expressed as follows:

$$\begin{aligned}\frac{dW'_{i,ss}(t')}{dt'} &= \frac{d}{dt'}\left(2^{\left\lfloor \frac{t'}{R'_i(t')}\right\rfloor}\right) = \frac{d}{\frac{1}{\alpha}dt}\left(2^{\left\lfloor \frac{\frac{1}{\alpha}\cdot t}{\frac{1}{\alpha}\cdot R_i(t)}\right\rfloor}\right) \\ &= \alpha\cdot\frac{d}{dt}\left(2^{\left\lfloor \frac{t}{R_i(t)}\right\rfloor}\right) = \alpha\cdot\frac{dW_{i,ss}(t)}{dt}. \qquad (7.14)\end{aligned}$$

By Eq. (7.14), we know that the dynamics of the TCP window in the slow start phase in the down-scaled networks is scaled by the parameter of $\frac{1}{\alpha}$, as compared to that in the original network.

Next we prove the cumulative throughput attained by a TCP connection in the slow start phase in the down-scaled network is equal to that in the original network. Note that we use "cumulative throughput", $T_i(t_i, t_{i+1})$, to denote the number of packets sent by the flow, i, in time interval $[t_i, t_{i+1}]$, which is $\int_{t_i}^{t_{i+1}} dr_i(t)$ when $r_i(t)$ is the rate function. Let $T_{i,ss}(t)$ be the cumulative throughput attained until time t in the slow start phase. (By means of $T_{i,ss}(t)$, we simply denote $T_{i,ss}(0,t)$.) Then $T'_{ss}(t')$ can be expressed as

$$\begin{aligned}T'_{i,ss}(t') &\approx 1+2+2^2+\cdots+2^{\left\lfloor \frac{t'}{R'_i(t')}\right\rfloor} = \sum_{k=0}^{\left\lfloor \frac{t'}{R'_i(t')}\right\rfloor} 2^k \\ &= 2^{\left\lfloor \frac{t'}{R'_i(t')}\right\rfloor} - 1 = 2^{\left\lfloor \frac{\frac{1}{\alpha}t}{\frac{1}{\alpha}R_i(t)}\right\rfloor} - 1 \\ &\approx T_{i,ss}(t). \qquad (7.15)\end{aligned}$$

Down-scaling of Unresponsive Traffic

In the case that unresponsive flows (e.g., UDP traffic) exist in the network, as no rate adaptation scheme such as TCP window dynamics (Eq. (7.13) is used, we have to forcefully adjust the rate of unresponsive traffic in the down-scaled network, so as to handle them in a consistent manner with Eqs. (7.7), (7.11), (7.13), and (7.14).

Specifically, let $\sigma_i(t)$ denote the rate function of a unresponsive flow i used in the original network. Since we down-scale the network bandwidth by a scaling factor of α, the rate function of flow i in the down-scaled network should also be down scaled by the same factor, i.e., $\alpha\cdot\sigma_i(t)$. With such an adjustment, unresponsive flows introduce the same degree of aggressiveness in the down-scaled network as they do in the original network.

Event Times in Down-scaled/Original Networks

Sometimes one would like to obtain the cumulative throughput attained by a TCP connection during a *specific* interval in steady state, say $[t_0, t_n]$, in the original network. As all the events in the down-scaled network are delayed by a factor of $\frac{1}{\alpha}$, one would expect to measure the throughput attained by the TCP connection in $[\frac{t_0}{\alpha}, \frac{t_n}{\alpha}]$. In what follows, we prove this is not the case, i.e., from the perspective of the attainable throughput, one can make the measurement in the same interval $[t_0, t_n]$ in the down-scaled network and then extrapolate the results in the original network.

Let t_k for $k = 0, 1, 2, \cdots, n$ represent the n event times at which events are respectively scheduled, and let $T_i(t_{k-1}, t_k)$ denote the cumulative throughput attained by a TCP connection i in the interval $[t_{k-1}, t_k]$. Then the total cumulative throughput attained by flow i can be expressed as

$$T_i(t_0, t_n) = \sum_{k=1}^{n} T_i(t_{k-1}, t_k). \tag{7.16}$$

The cumulative throughput, $T'_i(t_{k-1}, t_k)$ attained by a TCP flow during $[t_{k-1}, t_k]$, in the down-scaled network can be expressed:

$$\begin{aligned} T'_i(t_{k-1}, t_k) &= \int_{t_{k-1}}^{t_k} \left(\frac{dW'_i(t')}{R'_i(t')} \right) \\ &= \int_{t_{k-1}}^{t_k} \left(\frac{dW_i(t)}{\frac{1}{\alpha} R_i(t)} \right) = \alpha \cdot \int_{t_{i-1}}^{t_k} \left(\frac{dW_i(t)}{R_i(t)} \right) \\ &= \alpha \cdot T_i(t_{k-1}, t_k). \end{aligned}$$

Therefore, we can have the following as the total cumulative throughput attained by TCP flow i during $[t_0, t_n]$:

$$\begin{aligned} T'_i(t_0, t_n) &= \sum_{k=1}^{n} T'_i(t_{k-1}, t_k) \\ &= T'_i(t_0, t_1) + T'_i(t_1, t_2) + \cdots + T'_i(t_{n-1}, t_n) \\ &= \alpha \cdot (T_i(t_0, t_1) + T_i(t_1, t_2) + \cdots + T_i(t_{n-1}, t_n)) \\ &= \alpha \cdot \sum_{k=1}^{n} T_i(t_{k-1}, t_k) = \alpha \cdot T_i(t_0, t_n). \end{aligned} \tag{7.17}$$

Eq. (7.17) implies that from the perspective of obtaining the cumulative throughput attained by TCP connections in steady state, one does not have to scale event times. Instead, s/he simply simulates activities in the down-scaled network for the given set of event times, obtains the simulation results in each interval, and then infers (by up-scaling) simulation results for the same period in the original network.

7.1.3 Summary

With the proposed rescaling simulation model (RSM) (Eqs. (7.9)–(7.11), (7.13)–(7.15), and (7.16)–(7.17)), one does not have to scale **the event times** in the down-scaled network in order to obtain the throughput (Eqs (7.15) and (7.17)) and the dynamics (Eqs. (7.10), (7.11), (7.13), and (7.14)), since RSM can extrapolate the queue dynamics, the throughput changes, and the cumulative throughput in a specific interval in the original network simply by simulating activities in the α portion of the interval (of the original network) in the down-scaled network and then scaling up the results. For example, suppose two events e_1 and e_2 occur at t_1 and t_2 in the original network. The results observed in $[t_1, t_2]$ in the down-scaled network are actually the results in $[t_1, t_1 + \alpha \cdot (t_2 - t_1)]$ in the original network. As the results in $[t_1, t_1 + \alpha \cdot (t_2 - t_1)]$ represent the α portion of the results in $[t_1, t_2]$, the latter can be inferred accordingly.

In summary, one can measure both the throughput and the dynamics in $[t_i, t_{i+1}]$, $0 \leq i \leq n-1$, in the down-scaled network. The obtained results correspond to those in the α portion of each interval $[t_i, t_{i+1}]$ in the original network. The results in each interval in the original network are inferred by scaling up the obtained results. The entire set of simulation results can be obtained by repeating the procedure in each interval, and inferring the results accordingly. Figure 7.2 conceptually shows the throughput relationship between the original and the down-scaled networks.

7.1.4 Comparison with SHRiNK

As compared to SHRiNK [40], RSM possesses several desirable features: (i) RSM does not make any assumption on the input traffic, while SHRiNK relies on the Poisson assumption for the input traffic; (ii) RSM preserves the queue dynamics and hence the network capacity as perceived by TCP connections. In contrast, SHRiNK approximates the queue dynamics in the down-scaled network in order to preserve the queuing delay (round trip time). As a result, the down-scaled network may respond differently to TCP connections; (iii) RSM can work together with any AQM strategy, while SHRiNK cannot be used if the modified queue dynamics cannot keep the queuing delay invariant in the down-scaled network. For example, as reported in [40], if the AQM strategy is

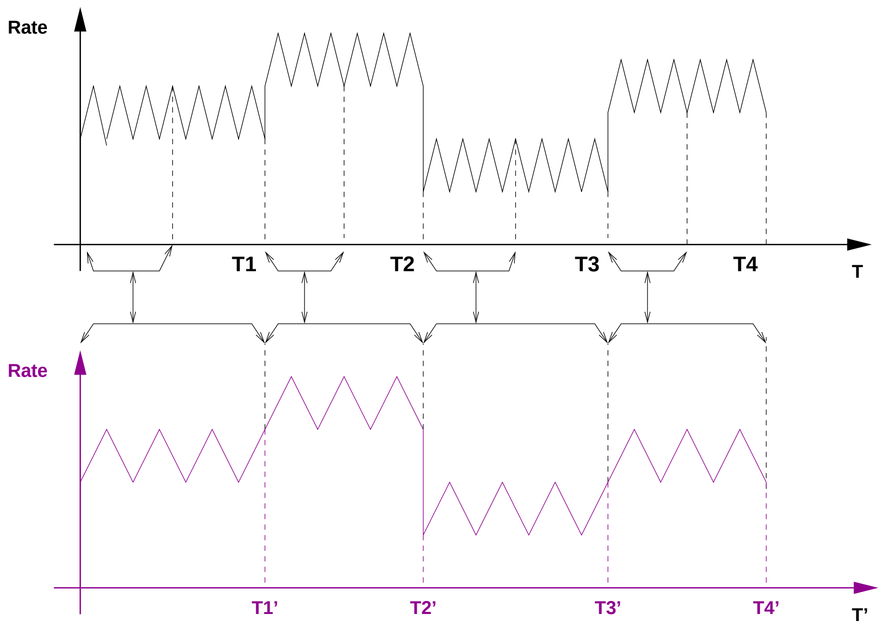


Figure 7.2: The relationship between the instantaneous throughput in the original network and that in the down-scaled network.

Drop Tail, SHRiNK cannot produce accurate results; (iv) RSM is universally effective irregardless of the value of the scaling factor, while the scaling factor in SHRiNK cannot be too small, as SHRiNK reduces the number of flows according to the factor; and (v) the correctness of RSM has been established for short-lived flows (which lasts only in the slow start phase), long-lived flows, TCP-friendly rate-controlled flows, and unresponsive flows (UDP flows).

Table 7.1 gives a quantitative comparison between RSM and SHRiNK. The network configuration under which the simulation is carried out is given in Figure 7.8. The buffer size is set to 100 packets and RED is used as the buffer management strategy. The minimum threshold, the maximum threshold, the maximum drop probability, and the gentle option of RED are set to 30, 70, 1, and enabled, respectively. As shown in Table 7.1, the execution time incurred in RSM reduces as the scaling factor decreases while SHRiNK cannot produce correct results and even cannot proceed when the scaling factor becomes small. This is because RSM does not change the number of connections in its rescaling operation, while SHRiNK reduces the number of connections.

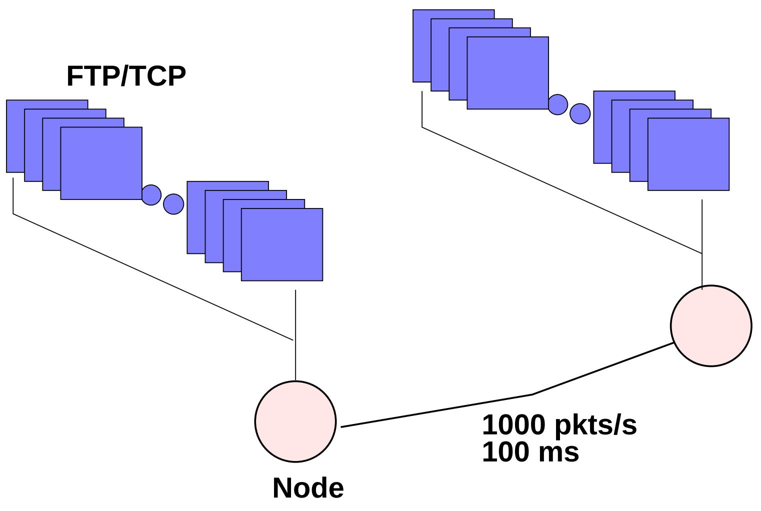
Table 7.1: The number of packets received per class and the execution time (s) in a 1000-second simulation run in RSM and SHRiNK. The network configuration is given in Figure 7.8, in which the buffer size is set to 100 and RED is used as the buffer management strategy. The minimum threshold, the maximum threshold, the maximum drop probability, and the gentle option of RED are set to 30, 70, 1, and enabled, respectively.

# of nodes per class	scaling factor	RSM				SHRiNK			
		class 0	class 1	class 2	time	class 0	class 1	class 2	time
20	1.0	520131	1671982	1676905	166.11	520131	1671982	1676905	166.11
	0.2	491415	1492095	1512170	32.29	506540	1570700	1604325	24.27
	0.1	490290	1537490	1529750	17.47	132890	1819520	1812290	10.24
	0.01	482000	1506000	1535100	3.80	N/A			
40	1.0	522755	1671982	1676905	189.99	522755	1671982	1676905	189.99
	0.2	473870	1596565	1620595	38.97	526885	1684530	1711410	28.16
	0.1	455530	1665950	1695290	22.75	170330	1959670	1965020	12.20
	0.01	421100	1671300	1724200	6.59	N/A			

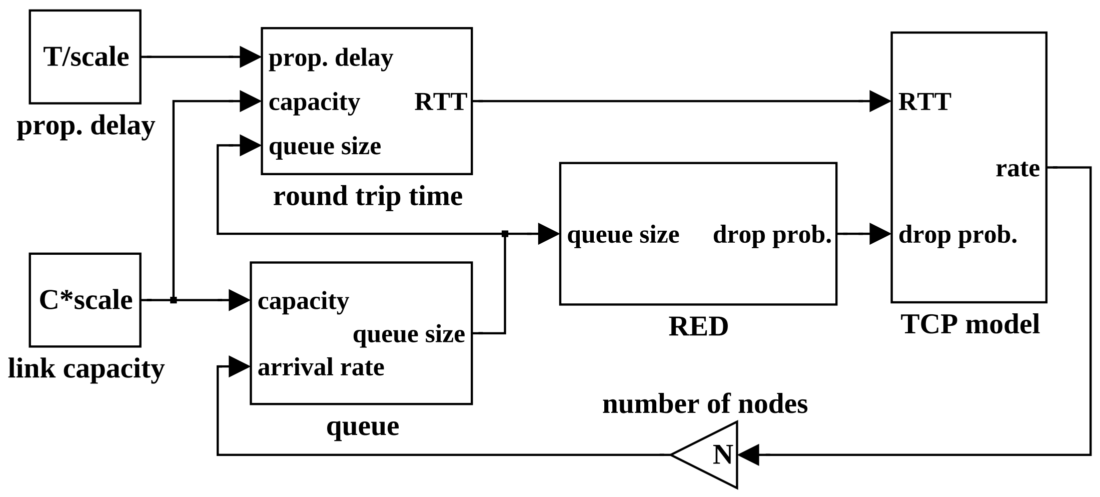
7.2 Validation of RSM with MATLAB

In this section, we validate RSM with *MATLAB* [43]. The specific effect when we use the RSM model with a scaling parameter α is that all the events that occur at time $t(t \geq 0)$ in the original network are delayed to $t'(t' \geq 0) = \frac{1}{\alpha} \cdot t$ in the down-scaled network. This is attributed to the fact that the link bandwidth is scaled down by the factor of α and the link delay is scaled up by the factor of $\frac{1}{\alpha}$ during the down-scaling operation. The dynamic network behavior that the original network exhibits at any time instant can be observed in the down-scaled network, except that the time instant at which each event occurs is delayed by the factor $\frac{1}{\alpha}$ (Eqs. (7.13) and (7.7)). In what follows we inspect the trajectory of the queue length and transient behavior of the dropping probability at a link to verify Eqs. (7.13) and (7.7) under the assumption that TCP dynamics in Eqs. (7.4)–(7.6) are correct.

The network topology in which we validate RSM is a simple dumbbell network topology, as shown in Figure 7.3 (a). The link capacity, the link delay, and the buffer, of the bottleneck link are set to 1000 packets/second, 100 *ms*, and 100 packets, respectively. RED is used as the AQM strategy. The minimum threshold, the maximum threshold, the maximum drop probability, and the gentle option of RED are set to 50, 100, 1, and enabled, respectively. A total of 100 TCP connections traverse the bottleneck link. Figure 7.3 (b) gives a corresponding *MATLAB Simulink* diagram of Figure 7.3 (a) implements the interaction between a bottleneck link (equipped with RED) and multiple TCP flows. In the figure, the *RTT module* executes the operation given in



(a) A network topology

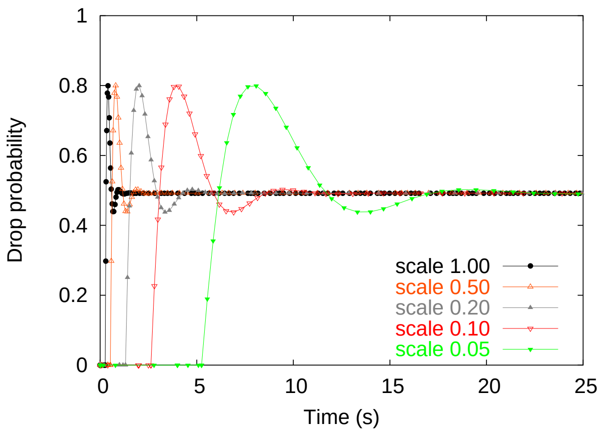


(b) MATLAB/Simulink representation

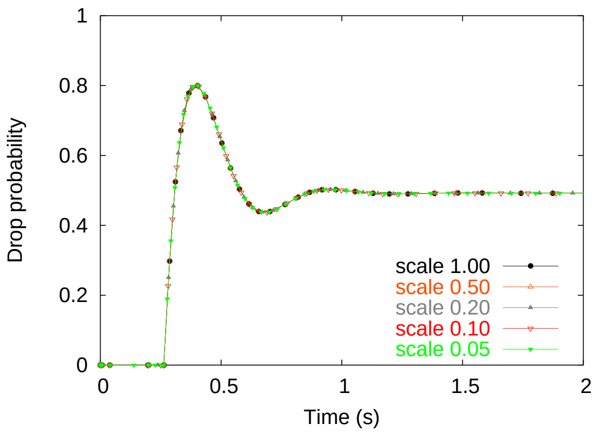
Figure 7.3: Network configuration and MATLAB/Simulink representation for the validation

Eq. (7.4), the *queue module* executes the function given in Eq. (7.5), the *RED module* performs the RED operations at the link [17] and the *TCP module* carries out the dynamics given in Eq. (7.6). Additionally, the *prop. delay* module and the *link capacity* module represent, respectively, the delay and the capacity of the link, and N is the number of flows.

Figures 7.4 and 7.5 give, respectively, the kinetic dropping probability and the instantaneous queue length on the bottleneck link. Figure 7.4 (a) gives the dynamic trajectories of the packet dropping probability of the link in the down-scaled network, where each trajectory corresponds to a different value of the scaling parameter α. Note that events are delayed by the factor of $\frac{1}{\alpha}$. Figure 7.4 (b) gives the trajectories of the packet dropping probability after the network is re-scaled



(a) In the down-scaled network



(b) After re-scaling up

Figure 7.4: Trajectories of the packet dropping probability in the down-scaled network and after the network is re-scaled up with the same scaling parameter α.

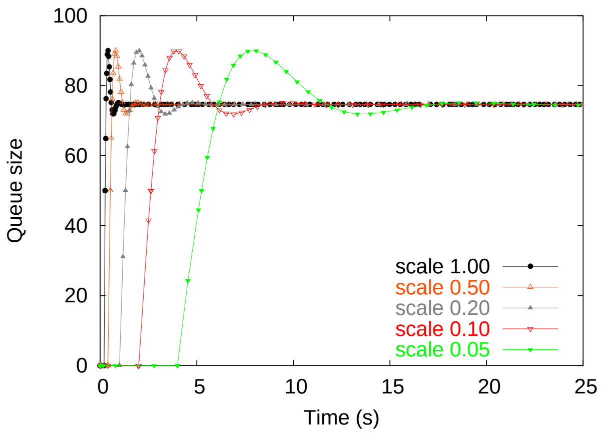
up with α. Note that all the trajectories agree with one another. Figure 7.5 gives the instantaneous queue length of the link in the down-scaled network ((a)) and after the network is re-scaled up ((b)). Conclusions similar to those made for Figure 7.4 can be drawn.

7.3 Simulation Study

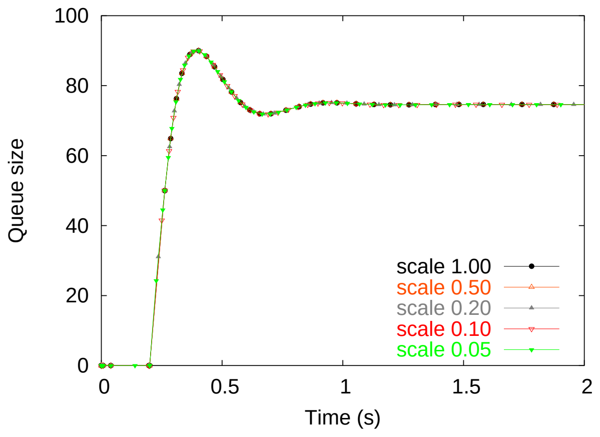
In this section, we discuss how we implement RSM in *ns-2* [4] and conduct a simulation study to compare the network behaviors (e.g., the queue dynamics) and the steady state behavior (e.g., the TCP throughput) obtained in RSM-based simulation and packet level simulation.

7.3.1 Implementation

Recall that the key idea of RSM is to reduce the number of events in the simulation by scaling down the original network (i.e., decreasing the link capacity and increasing the link delay by the same parameter), while preserving the bandwidth-delay product invariant in the down-scaled network (Section 7.1). As a result, neither the number of flows nor the queue-related parameters (e.g., the maximum buffer size and AQM parameters) need to be changed. As a matter of fact, the implementation of RSM is quite simple and straightforward. The network simulator takes as input the network topology and the scaling parameter, scales down the original network, carries out packet mode simulation in the down-scaled network, and then extrapolates the results corresponding to the original network. Only a light-weight preprocessor and a post-processor are needed to scale



(a) In the down-scaled network



(b) After re-scaling up.

Figure 7.5: Trajectories of the instantaneous queue length in the down-scaled network and after the network is re-scaled up with the same scaling parameter α.

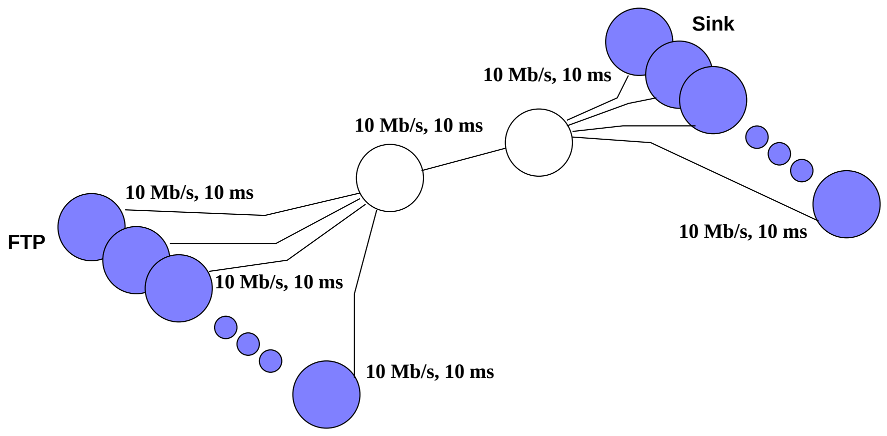


Figure 7.6: The network configuration used in the first set of experiments.

down the network and to extrapolate the simulation results.

7.3.2 Simulation

We have carried out an extensive *ns-2.1b9a* simulation study in a wide variety of network topologies and traffic loads, to assess the effectiveness of RSM in reducing the execution time and in capturing transient, packet level network dynamics. However, due to the space limit, in what follows we only present representative simulation results obtained in the three network configurations given in Figures 7.6, 7.7, and 7.8 Both the link and delay are labeled in the figures, unless otherwise specified. The major differences between the last two configurations lie in (i) the configuration in

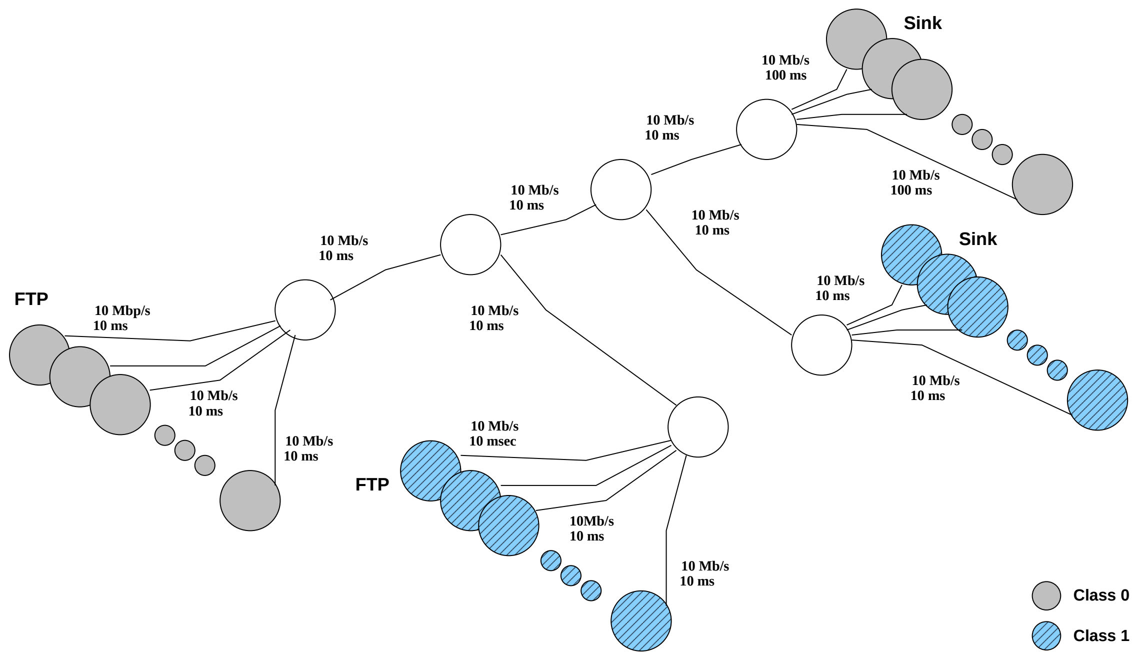


Figure 7.7: The network configuration used in the second set of experiments.

Figure 7.7 has one bottleneck, while that in Figure 7.8 has two bottleneck links; (ii) the number of input links (not flows) at each of the bottleneck links in Figure 7.8 is larger than that at the bottleneck link in Figure 7.7. As a result, the impact of flows of short RTTs on the flows of large RTTs becomes more severe than in the other case.

Different variations of TCP are used in the transport layer, but due to the space limit, we only present results with *TCP Reno* connections. Each router is equipped with a buffer of size 100 packets, with the default packet size set to 500 bytes. Different AQM strategies are employed at routers in different simulation runs. The various parameters in the AQM strategies are selected as follows. The minimum threshold, maximum threshold, maximum drop probability, and gentle option of RED are set to 30, 70, 1, and enabled, respectively. The update interval, reference value, and gain of REM are set to 10 ms, 50, and 0.1, respectively. The reference queue size and sampling frequency of PI are set to 50 and 100 times per second, and the remaining parameters, such as kp and ki (which in turn determine a and b) are determined in compliance with [23]. The desirable utilization, γ, of AVQ is set to 0.98, while the damping factor, α, is determined in compliance with Theorem 1 in [30] to ensure system stability ($\alpha = 0.15$). Parameters other than those mentioned above are set to their default values that come with the *ns-2* distribution. All the experiments are conducted in Linux 2.4.18 on a Pentium 4/1.9 *Ghz* PC with 1 *GBytes* memory and with 2 *GBytes*

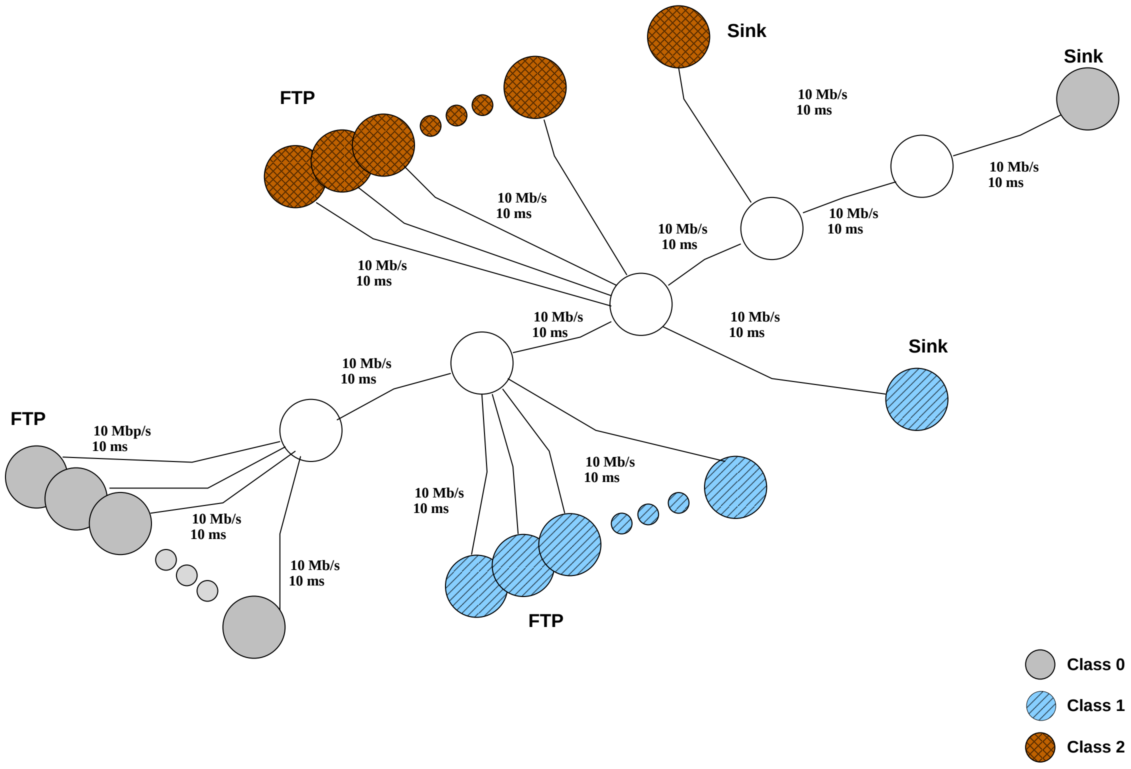


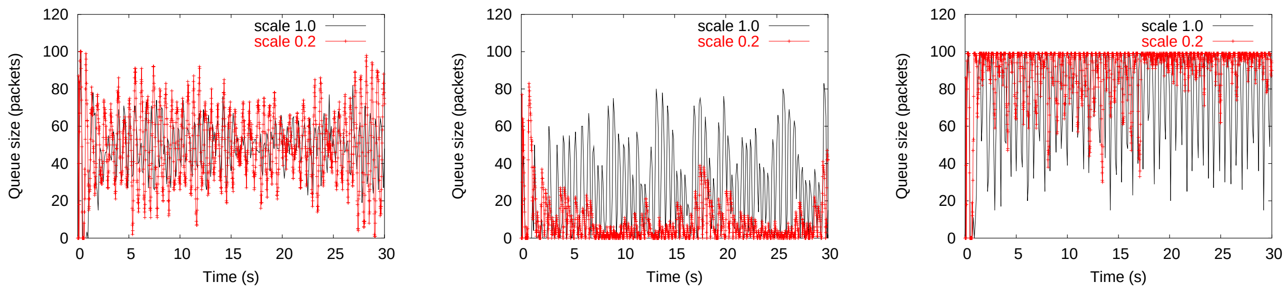
Figure 7.8: The network configuration used in the third set of experiments.

swap memory.

Performance in the presence of long-lived TCP connections

In this section, we study how effective RSM based simulation is in simulating long-lived TCP flows in perspective of network dynamics, throughput, and execution time.

Performance of RSM based simulation w.r.t. network dynamics First, we examine how close the network dynamics under RSM based simulation are to those under packet mode simulation. Figure 7.9 presents the instantaneous queue length versus time in the network configuration in Figure 7.6. The number of nodes in each class varies from 5 to 100, and the scaling parameter varies from 0.02 to 1.0 in these simulation runs. Also, router nodes employ a different AQM strategy (among Drop Tail, RED, REM, PI, and Virtual Queue) in each simulation run. We only present in Figure 7.9 the case that each class has 100 nodes, routers employ one of the three randomly chosen AQMs (which are PI, Virtual Queue, Drop Tail), and only show the initial 30 seconds (although each simulation run lasts for 1000 seconds). Regardless of the AQM strategy employed, the queue



(a) PI (b) Virtual Queue (c) Drop Tail

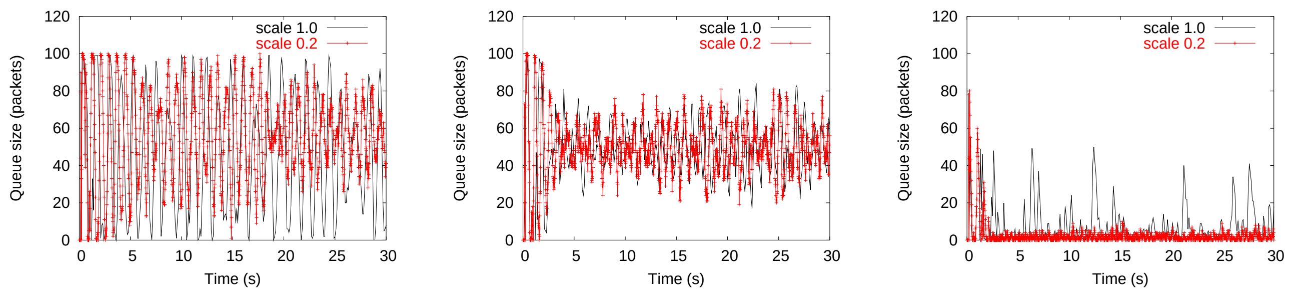
Figure 7.9: Instantaneous queue length versus time in the network configuration given in Figure 7.6 when the number of nodes is 100. The curve labeled with "scale 1.0" is the instantaneous queue length in the original network and that labeled with "scale 0.2" is the instantaneous queue length extrapolated from the down scaled network.

length extrapolated from the down-scaled network (the curves labeled with "scale 0.2") agrees extremely well with that observed in the original network (the curves labeled with "scale 1.0").

Figure 7.10 depicts the instantaneous queue length versus time in the network configuration given in Figure 7.7. The number of nodes in each class varies from 5 to 100, and the scaling parameter varies from 0.02 to 1.0 in these simulation runs. Also, router nodes employ a different AQM strategy (among Drop Tail, RED, REM, PI, and Virtual Queue) in each simulation run. Due to the space limit, we only present in Figure 7.10 the case that each class has 100 nodes and routers employ one of the three randomly chosen AQMs, and only show the initial 30 seconds (although each simulation run lasts for 1000 seconds). Regardless of the AQM strategy employed, the queue length extrapolated from the down-scaled network (the curves labeled with "scale 0.2") agrees extremely well with that observed in the original network (the curves labeled with "scale 1.0").

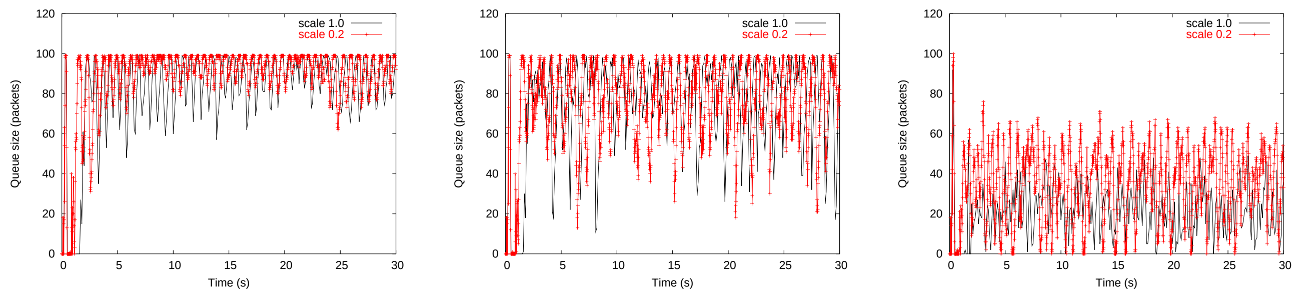
Figure 7.11 depicts the instantaneous queue length of the second bottleneck link versus time in the network configuration in Figure 7.8. Due to the space limit, we only present the case in which each class has 20 nodes and routers employ one of the three AQMs, Drop Tail, REM, and PI. Again the instantaneous queue length extrapolated from the down-scaled network agrees extremely well with that in the original network.

Performance of RSM based simulation w.r.t. error discrepancy We now quantitatively evaluate the discrepancy between results obtained in RSM based simulation and those in packet mode simulation. In both simulation modes, the TCP throughput is measured at a receiver and



(a) RED (b) PI (c) Virtual Queue

Figure 7.10: Instantaneous queue length versus time in the network configuration given in Figure 7.7 when the number of nodes per class is 100. The curve labeled with "scale 1.0" is the instantaneous queue length in the original network and that labeled with "scale 0.2" is the instantaneous queue length extrapolated from the down scaled network.



(a) Drop Tail (b) REM (c) PI

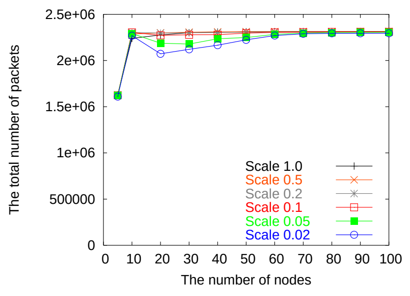
Figure 7.11: Instantaneous queue length versus time in the network configuration given in Figure 7.8 when the number of nodes per class is 20.

summed up to give the total throughput per class and the total throughput for the network.

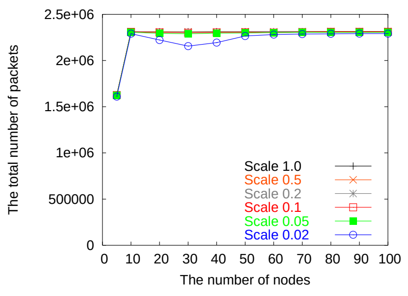
Figure 7.12 gives the total number of packets received in the network configuration given in Figure 7.6. Each simulation run lasts for 1000 seconds. The error discrepancy observed between two simulation modes is at most approximately 10 % of the capacity of the bottleneck link.

Figure 7.13 only gives the total number of packets received at class 0 nodes in the network configuration given in Figure 7.7. Each simulation run lasts for 1000 seconds. The error discrepancy observed between two simulation modes is at most approximately 10 % of the capacity of the bottleneck link. (The error discrepancy measured in the total throughput as well as the aggregate throughput of all the class 1 nodes are also less than 10 %.)

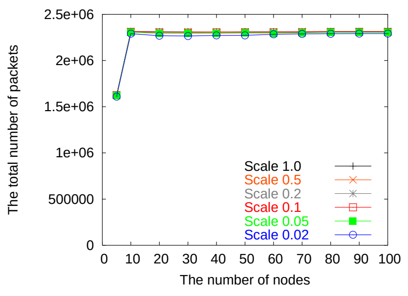
Figure 7.14 gives the total number of packets received at class 1 nodes, in the the network configuration given in Figure 7.8. Again each simulation run lasts for 1000 seconds. The error discrepancy is at most approximately 10 % of the capacity of the bottleneck link.



(a) PI

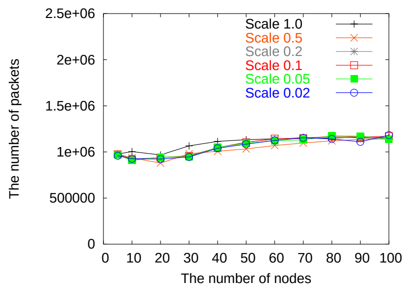


(b) Virtual Queue

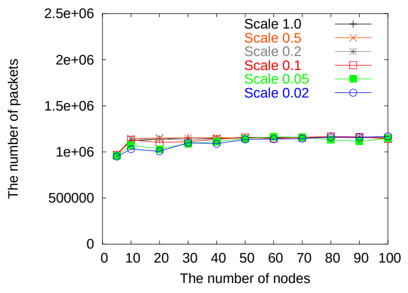


(c) Drop Tail

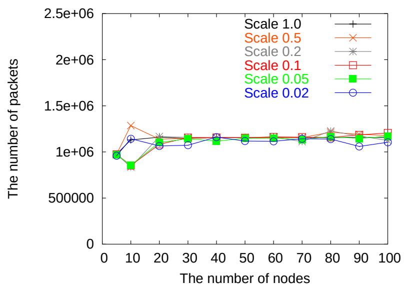
Figure 7.12: The total number of packets received at class 0 nodes in a 1000-second simulation run in the network configuration given in Figure 7.6.



(a) Class 0 with RED



(b) Class 0 with PI



(c) Class 0 with Virtual Queue

Figure 7.13: The total number of packets received at class 0 nodes in a 1000-second simulation run in the network configuration given in Figure 7.7.

Performance of RSM based simulation w.r.t. execution Time We now evaluate the performance gain of RSM (as compared to packet level simulation) in terms of the execution time required to carry out the simulation. Figure 7.15 depicts the execution time versus the number of nodes in a 1000-second simulation run in the network configuration given in Figure 7.7. The simulation results indicate more than an order of magnitude improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases (in terms of the number of nodes and the link capacity) or as the scaling parameter decreases.

Figure 7.16 depicts the execution time versus the number of nodes in a 1000-second simulation run in the network configuration given in Figure 7.7. The simulation results indicate more than an order of magnitude improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases (in terms of the number of nodes and the link capacity) or as the scaling parameter decreases.

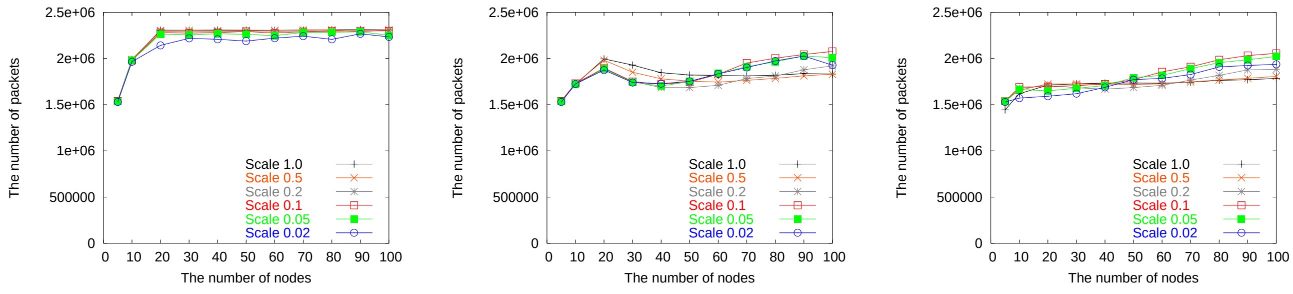

(a) Class 1 with Drop Tail (b) Class 1 with REM (c) Class 1 with PI

Figure 7.14: The total number of packets received at class 1 nodes in the network configuration given in Figure 7.8.

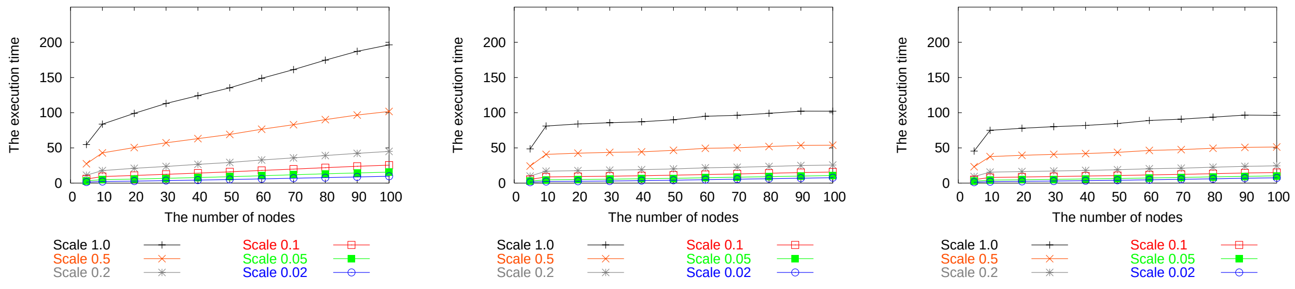
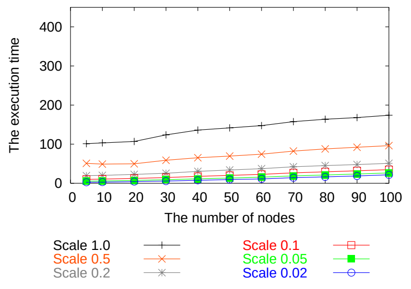

(a) PI (b) Virtual Queue (c) Drop Tail

Figure 7.15: Execution time (sec.) required to carry out a 1000-second simulation run in the network configuration given in Figure 7.6.

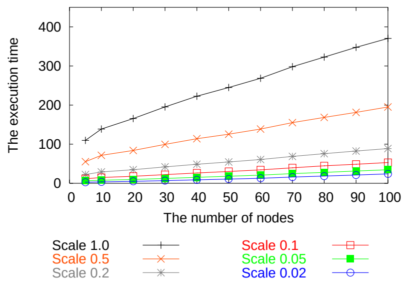
Figure 7.17 depicts the execution time versus the number of nodes in a 1000-second simulation run in the network configuration given in Figure 7.8. The speed-up in execution time as a result of employing RSM can be as large as 50 times.

Performance in the presence of long- and short-lived TCP connections

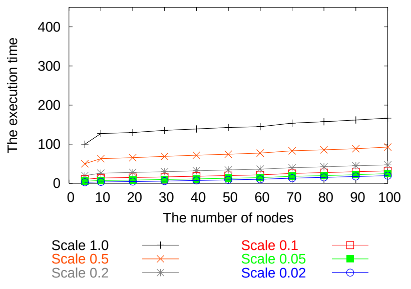
In this section, we explore more dynamic scenarios in which long-lived and short-lived TCP connections co-exist and interfere with each other. Each short-lived connection is generated by a Pareto traffic generator in *ns-2*. Packets are sent at a fixed rate of 200 Kbps during the on periods and no packets are sent during the off periods. The length of each on/off period follows a Pareto distribution with the Pareto shape parameter of 1.5 and the mean value of 100 *ms*. The application frame size is set to 210 bytes. Note that the Pareto traffic distributor is positioned on top of TCP, and that most of short-lived TCP flows only last in their slow start phase.



(a) RED

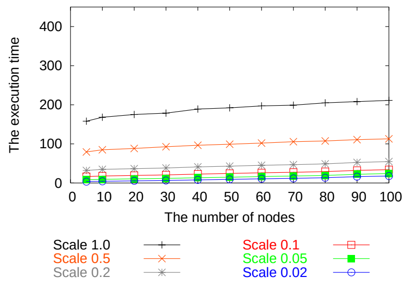


(b) PI

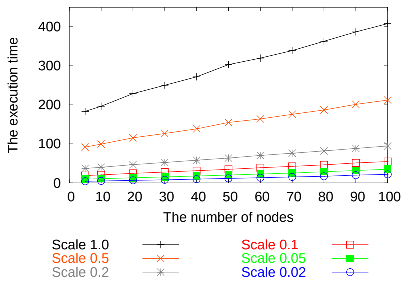


(c) Virtual Queue

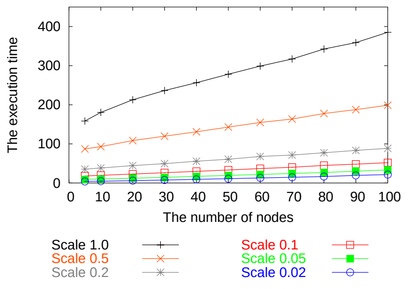
Figure 7.16: Execution time (s) required to carry out a 1000-second simulation run in the network configuration given in Figure 7.7.



(a) Drop Tail



(b) REM

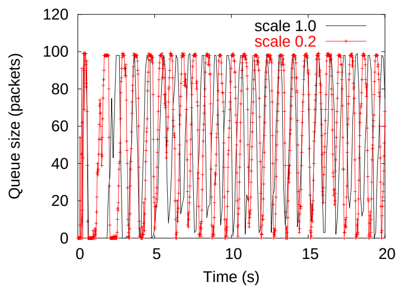


(c) PI

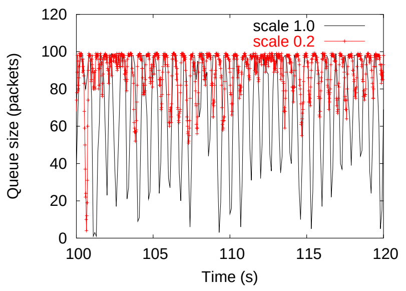
Figure 7.17: Execution time (s) required to carry out a 1000-second simulation run in the network configuration given in Figure 7.8.

Performance of RSM based simulation w.r.t. network dynamics Figures 7.18–7.19 depicts the instantaneous queue length versus time in the network configuration given in Figure 7.7, except that the capacities of all the link, excluding the bottleneck link, are increased to 100 Mb/s. Each class has 50 nodes. Each simulation run lasts for 300 seconds. Short-lived connections are only active in the interval [100, 200] seconds, and hence the entire simulation time is separated into three periods: [0, 100], [100, 200], and [200, 300]. Figure 7.18 (7.19) show the dynamic changes of the queue length during the 20 % of each period when AQM is **Drop Tail** (**PI**). We can see that the changes in the down-scaled network agree very well with the original behavior.

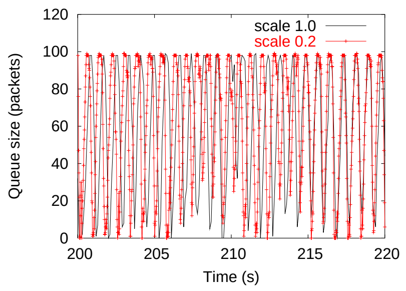
Performance of RSM based simulation w.r.t. error discrepancy We again quantitatively evaluate the discrepancy between results obtained in RSM based simulation and those in packet mode simulation. Figure 7.20 gives the total number of packets received at class 0 nodes in the network configuration given in Figure 7.7. Each simulation run lasts for 900 seconds, and short-lived connections are only active in the interval [300, 600] seconds. The error discrepancy observed



(a) From 0 to 20 (s)

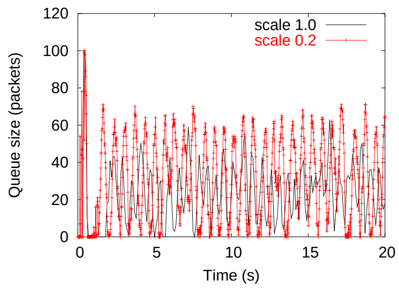


(b) From 100 to 120 (s)

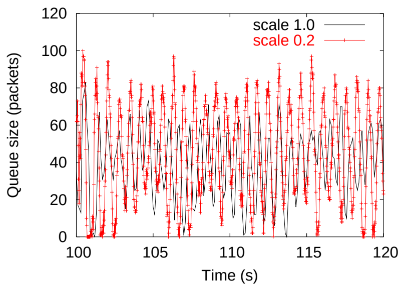


(c) From 200 to 220 (s)

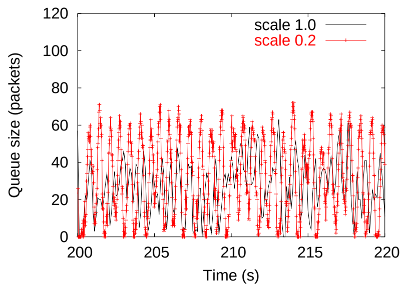
Figure 7.18: Instantaneous queue length versus time in the presence of both long-lived and short-lived TCP connections in the network configuration given in Figure 7.7. The capacities of all the links, excluding the bottleneck link, are increased to 100 Mb/s. Drop Tail is used as the AQM strategy.



(a) From 0 to 20 (s)



(b) From 100 to 120 (s)

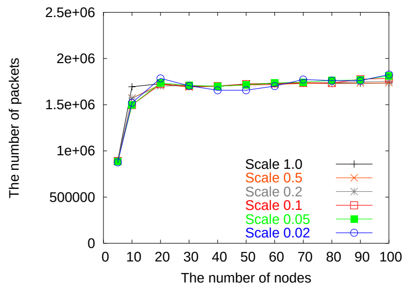


(c) From 200 to 220 (s)

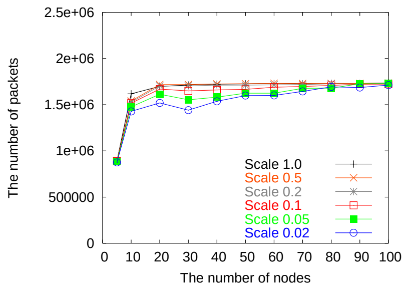
Figure 7.19: Instantaneous queue length versus time in the presence of both long-lived and short-lived TCP connections in the same network configuration used in Figure 7.18, except that PI is employed as the AQM strategy.

between two simulation modes is still confined by 10 % of the capacity of the bottleneck link. (Note that the error discrepancy per flow is much smaller than 10% of its attained throughput then.)

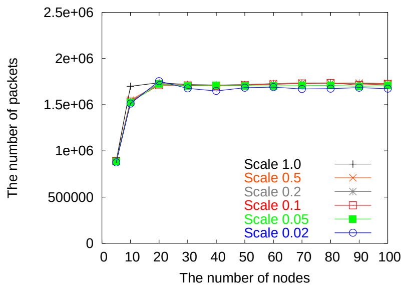
Performance of RSM based simulation w.r.t. execution time We evaluate again the performance gain of RSM (as compared to packet level simulation) in terms of the execution time required to carry out the simulation. Figure 7.21 depicts the execution time versus the number of nodes in a 900-second simulation run in the network configuration given in Figure 7.7. The same conclusion made in the case of presence of only long-lived TCP connections can be applied here: the performance improvement can be as large as 50 times.



(a) Drop Tail

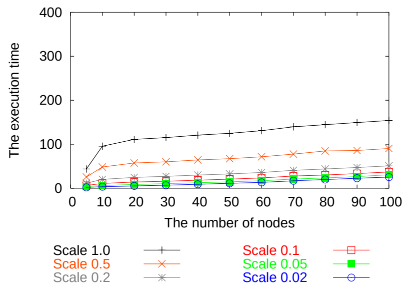


(b) PI

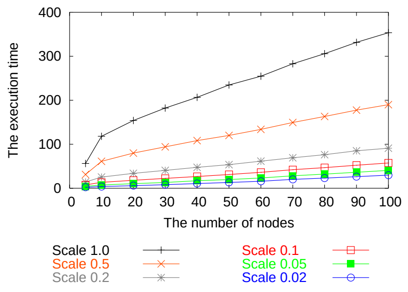


(c) REM

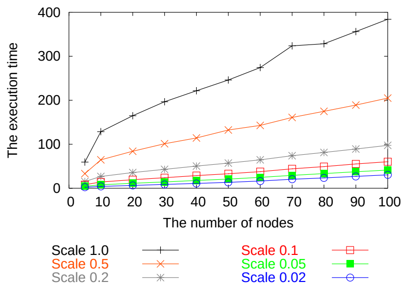
Figure 7.20: The total number of packets received at class 0 nodes in the presence of both long-lived and short-lived TCP connections in a 900-second simulation run in the same network configuration used in Figure 7.18.



(a) Drop Tail



(b) PI



(c) REM

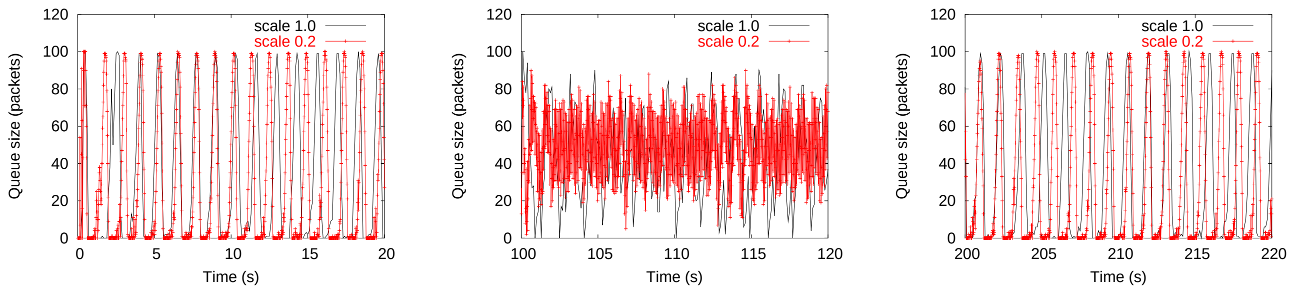
Figure 7.21: Execution time (s) required to carry out a 900-second simulation run in the presence of both short-lived and long-lived TCP connections in the same network configuration given in Figure 7.18.

Performance in the presence of long-lived TCP and unresponsive UDP connections

In this section, we investigate RSM in scenarios in which long-lived TCP connections co-exist with unresponsive UDP connections. Each UDP connection delivers traffic generated by a CBR traffic generator with the rate set to *750 Kbps*.

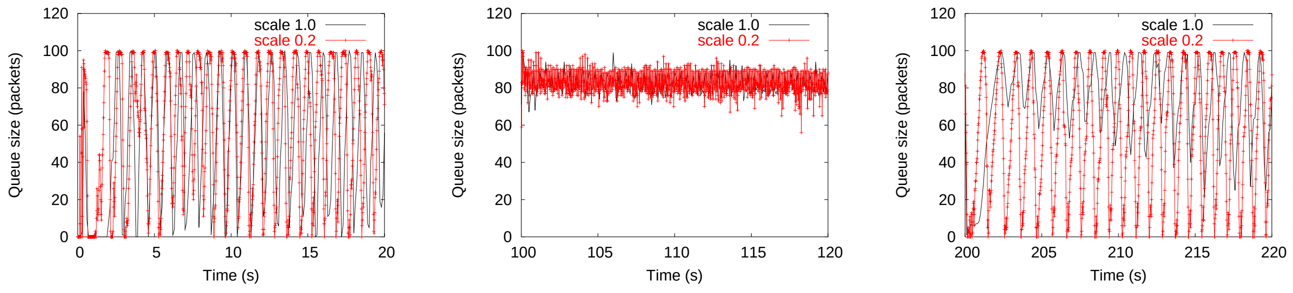
Performance of RSM based simulation w.r.t. network dynamics Figures 7.22–7.23 depict the instantaneous queue length versus time in the network configuration given in Figure 7.7, except that the capacities of all the link, excluding the bottleneck link, are increased to 100 Mb/s. Each class has 50 nodes. Each simulation run lasts for 300 seconds. UDP connections are only active in the interval [100, 200] seconds, and hence there are three periods in the entire simulation time: [0, 100], [100, 200], and [200, 300].

Figure 7.22 (7.23) show the dynamics changes of the queue length during the 20 % of each



(a) From 0 to 20 (s) (b) From 100 to 120 (s) (c) From 200 to 220 (s)

Figure 7.22: Instantaneous queue length versus time in the presence of both long-lived TCP and UDP connections in the network configuration in Figure 7.7, except that the capacities of all the links, excluding the bottleneck link are increased to 100 Mb/s. RED is employed as the AQM strategy.



(a) From 0 to 20 (s) (b) From 100 to 120 (s) (c) From 200 to 220 (s)

Figure 7.23: Instantaneous queue length versus time in the presence of both long-lived TCP and UDP connections in the same network configuration used in Figure 7.22, except that Virtual Queue is used as the AQM strategy.

period, when AQM is **RED** (**Virtual Queue**). The dynamic changes of the queue length in the down-scaled network agree very well with the original behavior.

Performance of RSM based simulation w.r.t. error discrepancy Figure 7.24 gives the total number of packets received at class 0 nodes in the network configuration given in Figure 7.7. Each simulation run lasts for 900 seconds, and UDP connections are only active in the interval [300, 600] seconds. The error discrepancy observed between two simulation modes is still within 10 % of the capacity of the bottleneck link.

Performance of RSM based simulation w.r.t. execution time Figure 7.25 depicts the execution time versus the number of nodes in a 900-second simulation run in the network configuration given in Figure 7.7. The same conclusion made in the previous two cases can be applied

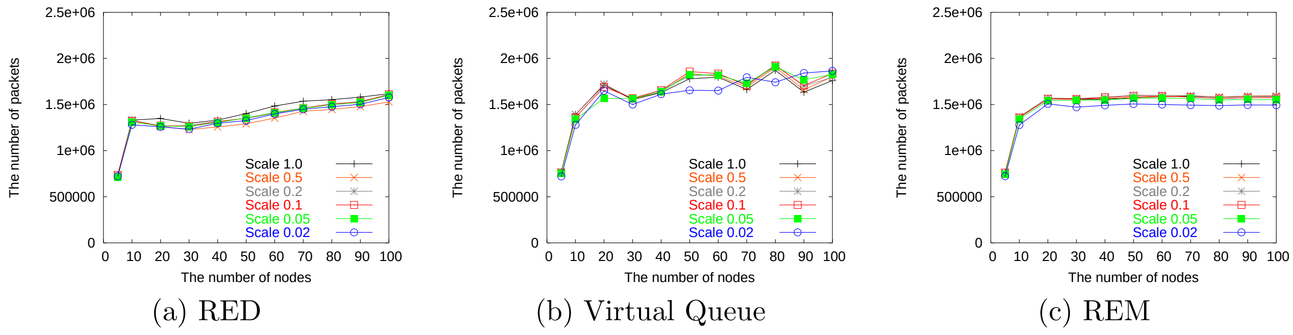


Figure 7.24: The total number of packets received at class 0 nodes in the presence of both long-lived TCP and UDP connections in a 900-second simulation run in the same network configuration used in Figure 7.22.

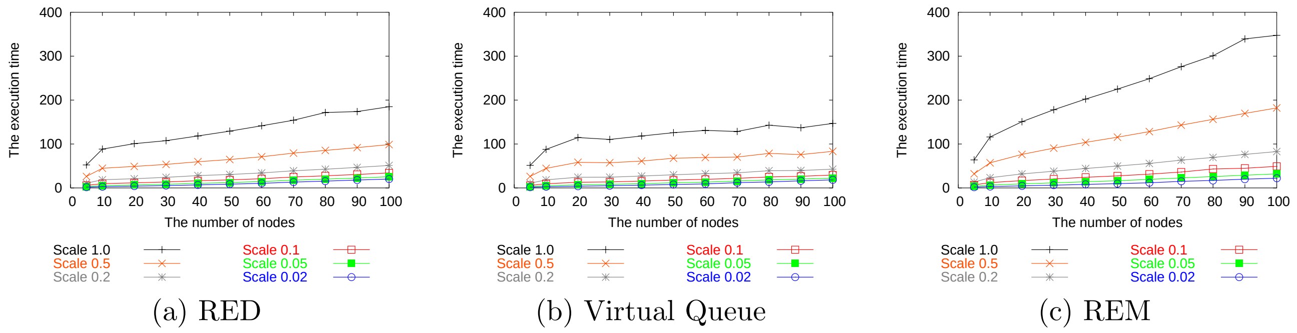


Figure 7.25: Execution time (s) required to carry out a 900-second simulation run in the presence of both short-lived TCP and UDP connections in the same network configuration used in Figure 7.22.

here: the speed-up in execution time as a result of employing RSM can be as large as 50 times.

Chapter 8

Conclusion

In this chapter, we specify the main contributions of the thesis in detail, and we then identify several research directions for future work.

8.1 Summary and Contributions

The fundamental contribution of this thesis is that we successfully enabled theoretical model based simulation techniques for simulating large scale networks, and conducted a comprehensive set of simulation experiments to validate their effectiveness in both correctness and performance gain. In particular,

- We took advantages of fluid model based approach, which is mainly employed to analyze TCP congestion control together with AQM strategy, to lay a fast simulation framework for IEEE 802.11-operated wireless LANs for the first time in Chapter 3. The framework abstracts a large number of packets with fluid chunks and characterizes the behaviors of them with the analytical model that we developed. Simulation results indicate that the proposed fluid model based simulation is effective in simulating IEEE 802.11-operated WLANs. It achieves in the best case more than two orders of magnitude improvement in terms of execution time as compared to the packet mode simulation, irrespective of the type of traffic (UDP or TCP). The performance improvement is more pronounced when the number of nodes, the number of applications running on each node, or the number of WLANs in a network to be simulated increases. The relative error, on the other hand, falls within 2 % in all cases as long as the value of the time step is appropriately determined;
- We contrived network calculus based simulation to speed up network simulations for TCP-

operated networks by approximating TCP traffics with the network calculus theory in Chapter 4. We first derive two analytical models to characterize how TCP operates in the *congestion avoidance* phase as well as in the *slow start* phase in perspective of traffic amount: (i) the throughput model that determines both the minimum and the maximum throughput for a TCP flow when the number of losses in an interval is given, and (ii) the network calculus model which, give based on the derived throughput model, the upper and lower bounds on the attainable TCP throughput in each interval. Network calculus based simulation can give an order of magnitude or more (maximally 30 times) improvement in execution time, while yielding an error discrepancy of minimally 1-2 % and maximally 8-12 %, of the bottleneck link capacity. In the case of error discrepancy, we observe that the error comes to be neglected when we conduct the study on a per flow basis since each network calculus based flow is allowed to produce its throughput in the range of the maximum and the minimum throughput measured in the corresponding packet mode simulation. Based on the simulation results, we conclude that in addition to their existing applications in the traffic regulation area, network calculus theory can also be used for network simulation, and that the network calculus based simulation is quite effective in simulating TCP traffic.

- Based on the recognition that we are usually interested in one flow or one point of networking, we proposed mixed mode simulation for IEEE 802.11-operated wireless LANs in Chapter 5. In the mixed mode simulation, the packet mode simulation co-exists with the fluid model based simulation and they affect one another at the point of interaction, the wireless medium. In order to specify the interaction, we developed an analytic model at the wireless channel among one foreground flow and multiple background flows. Simulation results indicate that mixed mode simulation speeds up simulation performance and produces the accurate results for both foreground flow and background flows in a WLAN. The mixed mode simulation achieves two orders of magnitude improvement in terms of execution time as compared to packet mode simulation. The achievement is almost equal to the achievement of the previous fluid model based simulation. The relative error, on the contrary, falls within 2 % in all the cases as far as the time step value is appropriate. Conspicuously, the mixed-mode simulation still shows packet level details for the foreground traffic.

- In the extended line of Chapter 5, we contrived mixed mode simulation for TCP-operated networks in Chapter 6, in which packet mode simulation runs one foreground, packet mode flow while network calculus based simulation runs the other background, network calculus based flows. In order to describe the interaction between two modes of simulation techniques at each point of bottleneck link, we developed an analytical model in terms of queue length, dropping, and achievable throughput. The simulation results indicate an order of magnitude improvement (maximally 20 times) in execution time and the performance improvement becomes more pronounced as the network size increases. The error discrepancy of the achieved throughput of one foreground flow in the mixed mode simulation, against the throughput range in the corresponding packet mode simulation (within which all the packet mode flows produce their throughput) falls within 1-2 % of the bottleneck link capacity in a wide spectrum of network topologies and traffic loads. Additionally, the average throughput of the other network calculus based flows in the mixed mode simulation also has the same error discrepancy. More promisingly, the mixed mode simulation still generate the behavior for one foreground flow in packet level details.

- We contrived the rescaling simulation methodology (RSM) based simulation for large scale IP networks in Chapter 7 for the cases that all the details for all the flows cannot be approximated as done in the previous simulation techniques. We scale down the original network by reducing the link capacity by a fraction, increasing the link delay with the same factor in the way that the bandwidth-delay product is preserved. By preserving the product invariant during the down/up scaling operations, the network capacity perceived by each TCP flow does not change in down-scaled network, which is analytically proved by presenting that the queue dynamics, the RTT dynamics, the window size dynamics, and even the effect of TCP-friendly and unresponsive flows remain unchanged in both up/down-scaling operation. The simulation results indicate an order of magnitude or more improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases in terms of number of nodes and network capacity or as the scaling parameter decreases. The error discrepancy between RSM based simulation and packet mode simulation, on the contrary, is minimally 1-2 % as long as the scaling parameter is appropriately determined and maximally 10 % in a wide variety of network topologies with various AQM strategies and traffic loads. The most noticeable point in RSM-based simulation is that it can handle any AQM strategy available, and also deal with UDP traffic in addition to TCP traffic.

8.2 Future Work

We have two tiers of future work: the one is short term work and the other is long term work. We identify the specific plans for each fast simulation method.

- In the case of fluid model based simulation for wireless LANs, we would like to extend the analytic models presented in Chapter 3 to accommodate hidden nodes, timing violations, external interference, self interference and overlapping channels. Such the components are necessary to extend the proposed fluid model based simulation to simulate multi-hop wireless LAN networks.

- In the case of network calculus based simulation, we plan to derive the packet dropping rules at each link for other active queue management (AQM) schemes. We have to seek better methods to solve the optimization problem. Furthermore, we would like to apply network calculus based simulation to other network architectures.

- In the case of mixed mode simulation for wireless LANs, we would like to elaborate the interaction model in Chapter 5 to address the cases in which we want to see packet level details of two or more number of foreground flows. We also plan to keep this method in the pace with the improvement which will be done to the analytical model for wireless LANs.

- In the case of mixed mode simulation for TCP-operated networks, we plan to accommodate all the improvement which will be made in network calculus based simulation. We also plan to carry out a more thorough analysis for the throughput difference presented in Chapter 6.

- In the case of rescaling simulation methodologies, we need to theoretically analyze the relationship between the error discrepancy and the scaling parameter. We also would like to include in our study wireless LANs in one of major part in large scale networks.

Together with the specific short term plans, we also identify long term plans, some of which pertain to how to integrate the developed simulation techniques into one framework and others focus on how to use the resulting simulation framework.

We plan to incorporate all the simulation models that we propose in this thesis into one integrated simulation framework. As a first step, we plan to improve a hybrid simulation framework in Chapter 3, so that TCP flows across over both wireless and wireline networks can be more correctly handled. Then, we will develop several interface modules capable of translating all the simulation units, such as fluid model based packet, network calculus based packet, and a collection of packets, in order to incorporate all the simulation techniques into one framework. The final simulation framework thus realized allows us to selectively use an appropriate simulation technique or to employ several techniques of simulation at the same time for a given network according to the degree of interest, the network type, and traffic type. Figure 8.1 demonstrates one example network configuration in simulation study, which consists of home, edge, access, and core networks. As shown in the figure, we can employ a different simulation technique for each area of concern and interest by using the prospective integrated simulation framework.

The fundamental goal of the integrated simulation framework inclusive of all the current fast simulation techniques is to speed up network simulation while the accuracy is acceptably guaranteed in simulation results. With the framework that rapidly estimates the upcoming status of networks, we can provide status information for any application which needs on-demand estimate for future status, such as delay or loss. Also, the framework can assist the network operators who usually need to know the effect of new network policy, resource management details, protocol deployment, and additional network load before they allow them permanently or for a while. Furthermore, frameworks can help QoS provisioning in the aspects of both admission control and resource allocation because it can empirically estimate the effect of newly arrived service requests on the current network traffic.

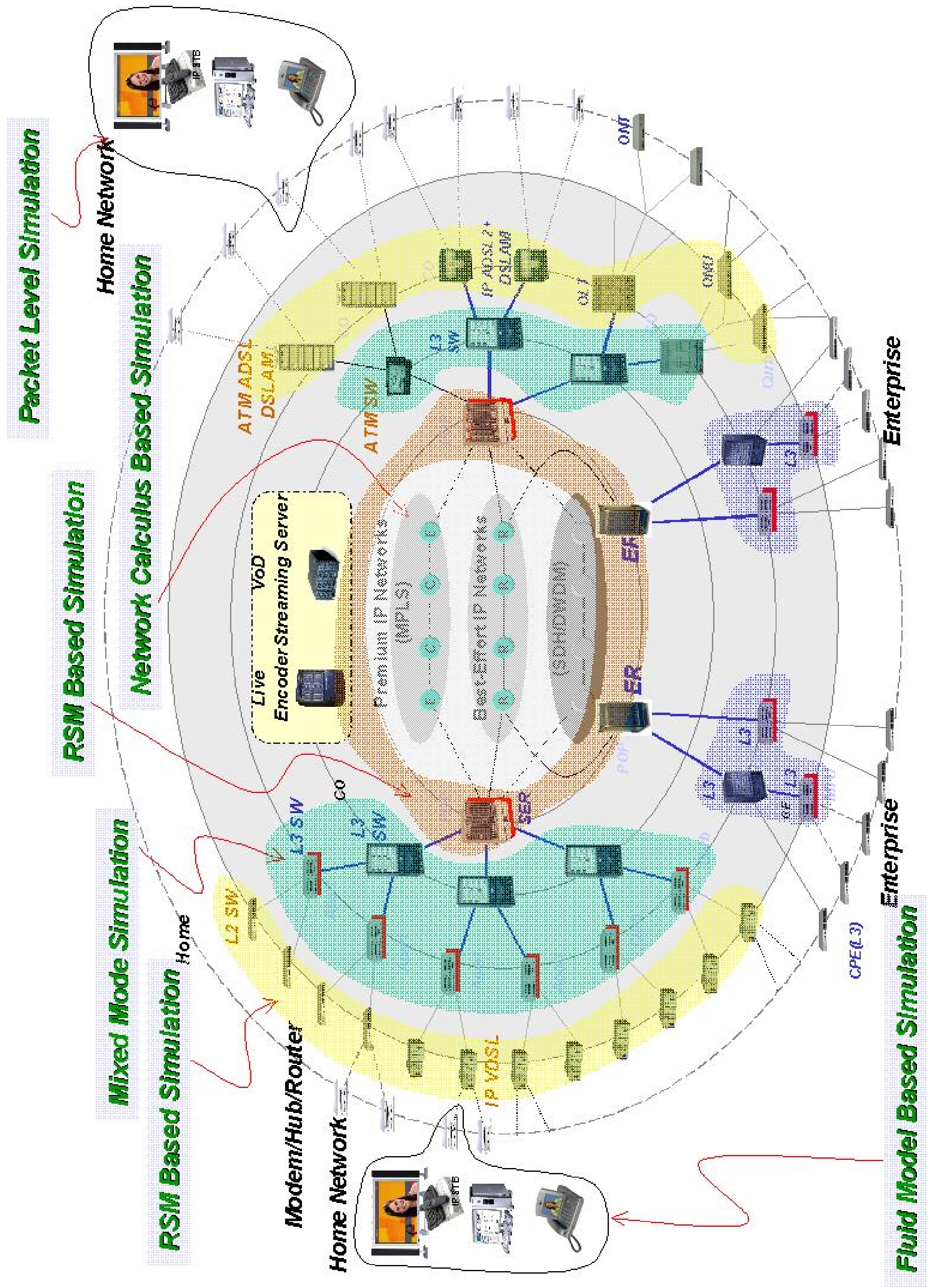


Figure 8.1: The prospective usage of the integrated simulation framework

References

[1] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity.* John Wiley & Sons, 1992.

[2] F. Baccelli and D. Hong. TCP is Max-Plus Linear: and what it tells us on its throughput. In *Proceedings of ACM SIGCOMM, (Stockholm, Sweden)*, August 2000.

[3] F. Baccelli and D. Hong. Flow Level Simulation of Large IP Networks. In *Proceedings of IEEE INFOCOM (San Francisco, California)*, April 2003.

[4] U. Berkeley, LBL, USC/ISI, and X. PARC. *The ns Manual.* http://www-mash.cs.berkeley.edu/ns/, April 2002.

[5] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

[6] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3), March 2000.

[7] G. Bianchi, L. Fratta, and M. Oliveri. Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs. In *Proceedings of IEEE PIMRC 1996*, volume 2, pages 392–396, October 1996.

[8] J.-Y. L. Boudec. Some Properties of Variable Length Packet Shaper. In *Proceedings of ACM SIGMETRIC, (Cambridge, Massachusetts)*, June 2001.

[9] J.-Y. L. Boudec and P. Thiran. *Network Calculus.* Springer-Verlag, 2002.

[10] C. R. Wylie Jr. *Advanced Engineering Mathematics (second edition).* McGraw-Hill Book Company, Inc., 1960.

[11] F. Calí, M. Conti, and E. Gregori. Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit. *IEEE/ACM Transactions on Networking*, 8(6), December 2000.

[12] M. M. Carvalho and J. J. Gaicia-Luna-Aceves. Delay Analysis of IEEE 802.11 in Single-Hop Networks. In *Proceedings of IEEE ICNP 2003, (Atlanta, Georgia)*, November 2003.

[13] C.-S. Chang. *Performance Guarantees in Communication Networks.* Springer-Verlag, 2000.

[14] C.-S. Chang, R. L. Cruz, J. Y. Boudec, and P. Thiran. A Min-Plus System Theory for Constrained Traffic Regulation and Dynamic Service Guarantees. In *Technical Report SSc/1994/024 EPFL*, July 1999.

[15] R. L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1), 1991.

[16] R. L. Cruz. A Calculus for Network Delay, Part II: Network Analysis. *IEEE Transactions on Information Theory*, 37(1), 1991.

[17] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4), 1993.

[18] C. H. Foh and M. Zukerman. Performance Analysis of the IEEE 802.11 MAC Protocol. In *Proceedings of the EW 2002 Conference, (Italy)*, February 2002.

[19] Y. Ge and J. C. Hou. An Analytical Model for Service Differentiation in IEEE 802.11. In *IEEE International Conf. on Communications (ICC'03)*, May 2003.

[20] J. Goodman and A. G. Greenberg. Stability of Binary Exponential Backoff. *Journal of the ACM*, 35(3), March 1998.

[21] Y. Gu, Y. Liu, and D. Towsley. On Integrating Fluid Models with Packet Simulation. In *Proceedings of IEEE INFOCOM 2004, (Hong Kong, China)*, March 2004.

[22] Y. Guo, W. Gong, and D. Towsley. Time-stepped Hybrid Simulation (TSHS) for Large Scale Networks. In *Proceedings of IEEE INFOCOM 2000, (Tel-Aviv, Israel)*, March 2000.

[23] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proceedings of IEEE INFOCOM 2001, (Anchorage, Alaska)*, 2001.

[24] IEEE. *IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification.* IEEE, 1990.

[25] F. P. Kelly. Charging and Rate Control for Elastic Traffic. *European transactions on Telecommunications*, 8, 1997.

[26] F. P. Kelly. Mathematical Modeling of the Internet. *Mathematics Unlimited - 2001 and Beyond (Springer-Verlag)*, 2001.

[27] F. P. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 49, 1998.

[28] H. Kim and J. C. Hou. A Fast Simulation Framework for IEEE 802.11-Operated WLANs. In *Proceedings of ACM SIGMETRICS 2004, (New York, New York)*, June 2004.

[29] H. Kim and J. C. Hou. Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation. In *Proceedings of IEEE INFOCOM 2004, (Hong Kong, China)*, March 2004.

[30] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ). In *Proceedings of ACM SIGCOMM 2001, (San Diego, California)*, August 2001.

[31] B. Liu, D. R. Figueiredo, Y. Guo, J. Kurose, and D. Towsley. A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-level Simulation. In *Proceedings of IEEE INFOCOM 2001, (Anchorage, Alaska)*, April 2001.

[32] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu. Fluid Models and Solutions for Large-scale IP networks. In *Proceedings of ACM SIGMETRICS 2003, (San Diego, California)*, June 2003.

[33] S. H. Low. A Duality Model of TCP Congestion. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management (Monterey, California)*, September 2000.

[34] S. H. Low and D. E. Lapsley. Optimization Flow Control I: Basic Algorithm and Convergence. IEEE/ACM Transactions on Networking, 7(6), 1999.

[35] N. Milidrag, G. Kesidis, and M. Devetsikiotis. An Overview of Fluid-Based Quick Simulation Techniques for Large Packet switched Communication Networks. In *Proceedings of SPIE ITCom 2001, (Denver, Colorado)*, August 2001.

[36] V. Misra, M. Gong, and D. Towsley. A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proceedings of ACM SIGCOMM 2000, (Stockholm, Sweden)*, September 2000.

[37] V. Misra, W. Gong, and D. Towsley. Stochastic Differential Equation Modeling and Analysis of TCP Window Size Behavior. In *Proceedings of Performance 1999, (Istanbul, Turkey)*, October 1999.

[38] D. Nicole, M. Goldsby, and M. Johnson. Fluid-based Simulation of Communication Networks using SSF. In *Proceedings of European Simulation Symposium (Erlangen-Nuremberg, Germany)*, October 1999.

[39] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM SIGCOMM 1998, (Vancouver, Canada)*, September 1999.

[40] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik. SHRiNK: A Method for Scalable Performance Prediction and Efficient Network Simulation. In *Proceedings of IEEE INFOCOM 2003, (San Francisco, California)*, March 2003.

[41] L. L. Perterson and B. S. Davie. *Computer Networks: A Systems Approach.* Morgan Kaufman, 1996.

[42] S. Shakkottai and R. Srikant. How Good are Deterministic Fluid Models of Internet Congestion Control? In *Proceedings of IEEE INFOCOM 2002, (New York, New York)*, June 2002.

[43] The Mathworks Inc. *MATLAB: The Language of Technical Computing.* The Mathworks Inc., 1999.

[44] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma. Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement. In *Proceedings of IEEE INFOCOM 2002, (New York, New York)*, June 2002.

[45] Y. Wu and W. Gong. Time Stepped Simulation of Queuing Systems. In *Technical Report, Department of Electrical and Computer Engineering, University of Massachusetts*, 2001.

[46] Y. Xiao. An Analysis for Differentiated Service in IEEE 802.11 and IEEE 802.11e Wireless LANs. In *Proceedings of IEEE ICDCS 2004, (Tokyo, Japan)*, Match 2004.

[47] A. Yan and W. Gong. Time-Driven Fluid Simulation for High-Speed Networks. *IEEE Transactions on Information Theory*, 45(5), 1999.

Vita

Hwangnam Kim received the M.S degree from the Department of Computer Engineering, Seoul National University, Seoul, Korea, in Feb. 1994, and received the B.E. degree from the Department of Computer Engineering, Pusan National University, Pusan, Korea, in Feb. 1992. After he had worked for LG Electronics, Ltd., Seoul, Korea from Jan. 1994 to June 1999, he joined Department of Computer Science, University of Illinois at Urbana-Champaign in 1999. In his Ph.D. research, he worked on how to enable theoretical model based techniques to simulate large scale networks. He contrived fluid model based simulation for IEEE 802.11-compliant networks, network calculus based simulation for TCP-operated networks, mixed mode simulation for both IEEE 802.11-compliant and TCP-operated networks, and rescaling model based simulation for IP networks.

Abstract

Modern data communication networks are extremely complex and do not lend well to theoretical analysis. It is common that network analysis can be rigorously made after leaving out subtle details that cannot be easily captured in the analysis. Instead, packet mode, event driven simulation studies are usually resorted to better study the network performance. The major obstacle in packet mode simulation is, however, the vast number of packets that have to be simulated for producing accurate results. What seems to be a reasonable solution is to incorporate theoretical modeling into packet mode simulation.

We developed a fluid model of describing the data transmission activities in IEEE 802.11-operated WLANs, and used it to explore whether or not fluid model-based simulation is effective in simulating WLANs. Fluid model based simulation is not well-suited for studying the network behavior under light and/or sporadic traffic, as it assumes a large number of flows in networks. To address the issue, we introduced network calculus based simulation; we characterize the interaction between TCP and AQM, determine necessary scheduling rules to regulate TCP traffic, and incorporate the rules into a simulation engine. Although both fluid model based and network calculus based simulation give encouraging results in terms of performance improvement, they cannot provide the packet level details, such as the instantaneous queue length and packet dropping probability, due to the use of larger simulation units. In order to examine the packet level dynamics, we proposed mixed mode simulation. In the mixed mode framework, packet mode simulation co-exists with theoretical model-based simulation within one simulation framework. We also proposed a new rescaling simulation methodology (RSM) to simulate IP networks with TCP and/or UDP traffic for the cases that all the network behaviors should be inspected. The underlying idea of RSM based simulation is to reduce the computation by scaling down the network to tractable one that can be simulated for a short time to produce sufficient results, and then to extrapolate the results for the original network with the obtained results.