Heterogeneity-Aware Approaches to Optimizing Performance of Computing and Communication Tasks

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Jun Huang

Certificate of Approval:

Heterogeneity-Aware Approaches to Optimizing Performance of Computing and Communication Tasks

Jun Huang

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
December 16, 2005

Heterogeneity-Aware Approaches to Optimizing Performance of Computing and Communication Tasks

Jun Huang

Signature of Author

Date of Graduation

Dissertation Abstract

Heterogeneity-Aware Approaches to Optimizing Performance of Computing and Communication Tasks

Jun Huang

Doctor of Philosophy, December 16, 2005
(M.S., University of Science and Technology of China, 2000)
(B.S., University of Science and Technology of China, 1997)

132 Typed Pages

Directed by Soo-Young Lee

As the domain of computing and communication systems grows, heterogeneity among computers and subnetworks employed for a task also increases. It is important to understand how heterogeneity affects performance of computing and communication tasks in order to optimally utilize heterogeneous resources. However, the effects of heterogeneity in heterogeneous computing and communication systems were either not taken into account explicitly or not thoroughly analyzed in the previous research work in this field.

In this dissertation, effects of heterogeneity are analyzed, and heterogeneity-aware approaches are proposed for both computing and communication systems.

In the computing system context, temporal heterogeneity refers to variation, along the time dimension, of computing power available for a task on a computer, and spatial heterogeneity represents the variation among computers. Effects of heterogeneity on the performance of a target task have been analyzed in terms of the mean and standard deviation of parallel execution time. The results reveal that, in minimizing the average parallel execution time of a target task on a spatially heterogeneous computing system, it is not optimal to distribute the target task linearly proportional to the average computing powers available on computers. Based on the analysis results, an approach to load balancing for minimizing the average parallel execution time of a target task is described. The proposed approach, of which validity has been verified through simulation, considers temporal and spatial heterogeneities in addition to the average computing power of each computer.

In the communication system context, the concept of temporal and spatial heterogeneity in the available communication resource is applicable to various levels of a communication network. Effects of heterogeneity on the performance of individual messages in a heterogeneous communication systems are analyzed. A heterogeneity-aware approach to source rate control is proposed, which utilizes the heterogeneity information on the available bandwidth in a UDP-based protocol, to improve throughput, dropping rate, end-to-end delay, and real-time performance. Two main components of the heterogeneity aware source rate control scheme are a receiver side feature extractor and a sender-side adaptive rate controller. The feature extractor captures the dynamic features in the bandwidth heterogeneity, and the source rate controller utilizes the extracted features in the rate control. Performance of the proposed source rate control scheme has been analyzed in detail through an extensive simulation for the single and multiple path media streaming, and multiple HAA and/or TCP flows.

Acknowledgments

First and foremost, I want to express my thanks and gratitude to my advisor, Dr. Soo-Young Lee, for his guidance, encouragement, and support in every step of my doctoral program at Auburn University. His comprehensive knowledge and insight on the research were of great help to the completion of this dissertation. His understanding and patience are sincerely appreciated, and his teaching on the academic research and many other aspects will be an everlasting treasure in my life.

My thanks also go to the members of my dissertation committee, Dr. Victor Nelson, Dr. Alvin Lim, and my outside reader Dr. Kai Chang. In spite of their very busy schedules, they read the drafts of this dissertation and provided many valuable comments. This dissertation owes a great deal to the time and attention they have given to it.

Last, but not least, I want to thank my family for their understanding and love along the path of my academic pursuits, and many of my friends for their generous help when I was in Auburn. Without their continuous support and encouragement, it would never have been possible to finish this dissertation.

Style manual or journal used Transactions of the American Mathematical Society (together with the style known as "auphd"). Bibliography follows van Leunen's *A Handbook for Scholars.*

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `auphd.sty`.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Formulation

1.1.1 Computing Systems

A cluster of computers is commonly used for high performance computing these days [1][2]. Such a system is often shared by multiple users and computers in the system may not be "identical." One of the most distinct characteristics of such an environment compared to a dedicated multiprocessor system is its heterogeneity [3]. Heterogeneity of such shared systems has been addressed from various viewpoints of high performance computing, but much of the research has been devoted to scheduling a stream of tasks (i.e. from the system's viewpoint) [4][5][6].

A measure of available computing power may be used to quantify the CPU share allocated for a given task, which represents the percentage of CPU cycles (or time slots) per unit time that the task can utilize at a given moment. This measure usually varies with both time and space. Therefore, the heterogeneity in computing systems can be classified into two types: *temporal heterogeneity* and *spatial heterogeneity*. The former refers to the variation of available resources with time on a computer, and the latter represents the variation of available resources among computers.

1.1.2 Communication Systems

A communication network, whether wired or wireless, is normally shared by multiple users who submit their requests at any time. Heterogeneity in communication systems can also be classified into two types: (1) the communication resources available on a channel for a request (or traffic, messages) varies with time, i.e., a channel is temporally heterogeneous, and (2) different channels would have different characteristics in terms of the communication resources available to them, i.e., channels in a network are spatially heterogeneous.

The concept of temporal and spatial heterogeneity in the available communication resources is applicable to various levels of a communication network. For example, temporal heterogeneity may be considered within individual sub-networks and spatial heterogeneity among sub-networks. These two types of heterogeneity can have a significant effect on the Quality of Service (QoS) of an individual request, such as delay and transfer time, and also network-wide performance measures such as utilization, system throughput, and queue length.

It is much more difficult to quantify the available resources in communication systems than in computing systems. There are various sources of heterogeneity on the Internet, and they can be divided into the following three main categories [7]:

- The topology of the Internet is constantly changing and the types of link (slow modems, fiber optic links, copper or glass wires, radio, infrared, etc.) span a wide range and are generally unknown to an end-user. Also, multi-path and dynamic routing introduce extra complexities.

- There are many protocols coexisting in the Internet, and each protocol has been implemented by multiple communities. For example, the widely used Transmission Control Protocol (TCP) has many different implementations, for example, Tahoe, Reno, NewReno, Sack, etc. Also, different applications ask for different features.

- Background competing traffic varies constantly in an unpredictable way.

Each individual heterogeneity is either not directly observable or cannot be measured accurately. For the protocol heterogeneity, a session is only able to know its own protocol, and it is almost impossible to have knowledge of other sessions' protocols. The heterogeneity due to background competing traffic is also hard to model in detail. Since it is either impossible or unnecessary to describe different components of heterogeneity individually, "available bandwidth" is used as a measure to quantify the "net" or aggregated effect of all the sources of heterogeneity in this dissertation.

1.2 Related Work

1.2.1 Heterogeneous Computing Systems

One of the important issues is how to utilize a heterogeneous computing system to optimize performance of a target task. This issue involves modelling the heterogeneity of the system [8][9], performance estimation [10][11][12][13], reliability [14], scheduling and load balancing [15], etc.

The concepts of "machine heterogeneity" and "task heterogeneity" are employed in an effort to model task execution time on heterogeneous computing systems [8][9]. The machine heterogeneity refers to the degree to which the machine execution times vary for a given task and the task heterogeneity refers to the degree to which the task execution times vary for a given machine. The main objective was to simulate different heterogeneous computing environments in evaluating the behavior of the task mapping heuristics.

A performance estimation model for heterogeneous parallel applications is proposed in [10]. They addressed the heterogeneity among the varying platforms and operating environments used and the possibility of having multiple tasks of the parallel application on each machine.

In [11], a model that calculates the slowdown imposed on applications in time-shared multi-user clusters is defined. Three kinds of "slowdown" are identified and the effects of slowdown on application performance are verified with emulated loads.

In [12], the issue of predicting parallel execution time of non-deterministic bulk synchronous computations, where computation time on each computer is modelled as a random variable, is considered. A tighter bound of the average parallel execution time has been derived. Though the randomness of computation time does not originate from heterogeneity of the computing system in their model, the results may be applicable to estimating parallel execution time on a heterogeneous computing system with minor modification.

In [13], the issue of modelling and characterizing parallel computing performance of heterogeneous Network Of Workstations (NOW) is considered, and the effects of heterogeneity on efficiency and scalability are investigated. The computing power (speed) "weight" is used to quantify the spatial heterogeneity among workstations, but the temporal heterogeneity of computing power on each workstation is not considered.

In [14], the probability of application failure is defined to quantify the reliability of a heterogeneous computing system. Minimizing execution time usually conflicts with improving reliability, and algorithms capable of trading off execution time for reliability are proposed.

In [15], a “stochastic scheduling” strategy, where the performance variability is considered in partitioning a data parallel application to be executed on a heterogeneous computing platform, is described. The mean and standard deviation of the predicted completion time of a task are used in scheduling, along with a “tuning factor” which specifies how “conservative” the scheduling is to be.

1.2.2 Heterogeneous Communication Systems

Realization of an overlay network involves mapping its topology onto the base (physical) network [16][17][18][19]. This mapping requires finding a “path” in the base network for each “edge” in the overlay network. Here, a path may consist of one or more channels or links. The path is to be selected such that it satisfies requirements (for example, bandwidth) of the corresponding edge. Also, multiple paths may be employed for an edge when a single path cannot meet the requirements of the edge. Heterogeneity must be considered in mapping of an overlay network and scheduling of messages in the network, in order to optimize its performance.

In an effort to minimize communication overhead in high performance computing, multiple communication paths between computing nodes were employed [20]. The main issues are how to choose one path over the others and how to partition a large size of data (message) among multiple paths. They considered “characteristics” of paths, such as latency, in path selection and aggregation. However, only spatial heterogeneity in terms of the average characteristics has been considered. That is, temporal variation of the characteristics was not taken into account.

The idea of utilizing multiple paths was also employed for video streaming in the "path diversity" framework [21]. Here, the objectives are to minimize packet loss caused by congestion due to time-varying bandwidth and to provide sufficient bandwidth for media streaming, by using more than one path simultaneously from a source to a destination. A quantitative measure of temporal heterogeneity in the bandwidth was not considered in their algorithms.

In order to quantify (temporal) heterogeneity in the channel bandwidth, one may consider a second order measure of the bandwidth in addition to its mean. Earlier, heterogeneity in the available computing powers of computing nodes on a cluster or Grid was studied in terms of load balancing [22][23], and it has been shown that average parallel execution time of a task can be substantially reduced by taking the standard deviation of available computing power on each node into account when partitioning the task over multiple nodes.

Most of the previous work on communication bandwidth management utilized either the average or instant bandwidth. Nevertheless, there have been some research efforts on scheduling and allocation problems under varying channel conditions [24][25][26]. Basically, heuristic approaches are taken to compensate for variations in the channel conditions in order to satisfy certain QoS requirements. Another load-aware routing protocol [27] also considers only the average load at intermediate nodes. However, heterogeneity was not explicitly taken into account. More specifically, the second order moment of available channel bandwidth was not quantitatively utilized to optimize network performance.

1.2.3 Network Measurement

Measurement of the bottleneck bandwidth over a network path may be performed on either the sender (source) or receiver (sink) side. Sender side measurement avoids the need for modification of software on the receiver end, and therefore a new protocol is easier to deploy. However, the measurement delay for a sender side scheme is about twice that of a receiver side scheme. Depending on the requirement of a specific application, where to perform the measurement needs to be determined.

Several schemes for probing the link states based on "packet pairs" were proposed in [28][29][30]. The sender transmits probing packets in the form of back-to-back pairs, and estimates the bottleneck service rate from the spacing of the acknowledgements (ACK). However, this approach relies on several unrealistic assumptions which may lead to inaccurate measurements. In order to improve the accuracy, an intersection filtering method with a multi-phase variable packet size was used in estimating the bandwidth [31]. The main problem with this scheme is that it is difficult to properly set the bin width and boundary of the histogram without knowing the shape of the distribution in advance. To overcome this difficulty, a packet pair scheme combined with bandwidth filtering using a *kernel density estimator* algorithm was proposed [32][33]. Note that the post-measurement processing/filtering steps in [31][32][33] are performed off-line, therefore, they would not be appropriate for applications where the real-time requirement is critical.

1.2.4 Source Rate Control

TCP uses explicit rate/congestion control. All implementations of TCP employ the so-called *Additive Increase and Multiplicative Decrease* (AIMD) algorithm first proposed by Jacobson in 1986 [34][35]. The main problems of this algorithm are its slow convergence speed and large rate oscillation [36]. Also, its multiplicative decrease scheme, which cuts its sending rate in half in response to each indication of link congestion, may unnecessarily waste the available network bandwidth. For a smoother and more gradual rate adaptation, many TCP-friendly congestion/rate control schemes have been proposed [37]. For example, a Loss-Delay based Adaption algorithm (LDA+) [38] is a variant of AIMD congestion control scheme, which adjusts the source rate dynamically based on the current network situation (using an estimate of the bottleneck bandwidth obtained by a packet pair approach). The bandwidth share of a flow is determined using the TCP model during loss situations, and can be increased by a value that does not exceed the increase of the bandwidth share of a TCP connection for the case of no loss. Because AIMD-like control mechanisms mostly focus on fairness among competing TCP traffic flows and usually lead to bursty traffic, they are not preferred by many real-time applications, when one is more interested in per-flow performances such as throughput, end-to-end delay, and inter-packet arrival time (less jitter).

A TCP-friendly solution has been proposed to evenly space TCP packets injected into the network over an entire round-trip time, so that data is not sent in a burst [39]. However, this approach often leads to significantly worse throughput than the regular TCP because it is susceptible to synchronization losses and delays congestion signals [40].

A feedback mechanism has been introduced for packet-spacing techniques [41][42], which enables UDP-based applications to perform well while being adaptive and self-regulating. This damped Packet-Spacing Protocol (PSP) transmits data at a near optimal rate by using a simple feedback mechanism that reports packet loss every round-trip time. However, the only congestion indicator used in PSP is packet loss, which may not be sufficient for optimizing QoS on a network with a high degree of heterogeneity in bandwidth.

1.3 Motivation

In most of the previous work, (*i*) not all types of heterogeneity were taken into account, (*ii*) heterogeneity was often considered only implicitly, and (*iii*) effects of heterogeneity were not thoroughly analyzed.

The heterogeneity in both computing and communication systems needs to be quantitatively represented. It's also necessary to have a clear understanding of how heterogeneity affects the performance of a given task, and show how the proposed approach can be applied to various application scenarios.

1.4 Research Objectives

The effects of various types of heterogeneity are to be analyzed to show the possibility of improving performance of tasks by considering the information of heterogeneity. Heterogeneity-aware approaches (HAA) are proposed to improve performance of both computing and communication systems.

1.4.1 Computing Systems

Effects of spatial and temporal heterogeneity on the performance of a target task are first analyzed in detail. It is shown that, it is not optimal to partition a task over computers linearly proportional to their average computing powers in order to minimize the average parallel execution time.

A heterogeneity-aware approach (HAA) to load balancing, designed to minimize the average parallel execution time of a target task in a spatially heterogeneous computing environment, is then proposed in order to provide a theoretical foundation for developing a practical load balancing scheme. The proposed approach explicitly takes into account the standard deviation of available computing power in addition to the average computing power available on each computer. It has been shown that the average parallel execution time of a target task can be significantly reduced by the approach.

1.4.2 Communication Systems

The TCP/AIMD-like source rate control schemes usually lead to bursty traffic and therefore are not preferred by many real-time applications. The use of feedback based packet-spacing approaches [41][42] improves real-time QoS in some situations. However, "packet loss" by itself may not be sufficient for describing the dynamic features of rapidly varying bandwidth. A bandwidth measurement technique could be used to provide some additional information to the source rate controller (as in [38]), but previous researchers have only used the first order feature of bandwidth samples. In order to extract useful features from the measurement, some types of post-measurement technique needs to be used. In a real network, however, a rate control scheme cannot afford a time-consuming filtering algorithm [31][32][33].

In this dissertation, a heterogeneity-aware approach to source rate control is proposed, which is adaptive to time-varying available (bottleneck) bandwidth. The proposed scheme extracts the second-order moment (standard deviation) as well as the first-order moment (mean) of the available bandwidth and explicitly takes them into account in determining the source rate dynamically. A simple feature extractor implemented at the receiving end measures the *features*, including the mean and standard deviation, using a packet-pair approach and sends them to the source. For the feature measurement, data packets are used instead of inserting control packets and, therefore, no extra traffic is generated.

Applications of the HAA rate control scheme to the path selection problems on an overlay network, and a multi-path media streaming framework are also considered.

Chapter 2

Effects of Heterogeneity on A Target Computing Task

2.1 System Model

The high performance computing system to be employed in this study consists of N heterogeneous computers interconnected via a communication network. The speed of processor, speed and capacity of memory (including cache), and bandwidth of I/O system, etc. may not be the same on all computers. Different software, including the OS, may be available on each of the computers.

All these hardware and software components collectively affect the effective computing power available for applications on each computer. It is the available computing power for a target task that eventually determines execution time of the task. Hence, in this dissertation, "availability" of computing power and communication bandwidth is employed in defining heterogeneity.

2.1.1 Availability

Let the maximum computing power of computer C_i be denoted by α_i for a target task for $i = 1, \ldots, N$, where N is the number of computers. The computing power available for the task at time t may be expressed by $\alpha_i A_i(t)$ where $A_i(t)$ is the "availability" of C_i for the task at time t and $0 \leq A_i(t) \leq 1$. The mean and standard deviation of $A_i(t)$ are denoted by a_i and σ_{A_i}, respectively.

In the steady state, a_i and σ_{A_i} are assumed to be fixed and do not vary with time while $A_i(t)$ varies with time. In the time-varying state, not only $A_i(t)$ but also a_i and/or σ_{A_i} vary with time, i.e., one needs to use the notations $a_i(t)$ and $\sigma_{A_i}(t)$. The time-varying state is not considered in this dissertation.

Availability of communication bandwidth can be defined similarly with notations β_i, $B_i(t)$, b_i, and σ_{B_i} for the maximum, instantaneous, mean and standard deviation of bandwidth, respectively. To avoid redundancy, they are not defined separately. One difference is that $B_i(t)$ tends to decrease as N increases due to the contention on the shared media, while $A_i(t)$ is independent of N, especially for data parallel applications.

2.1.2 Heterogeneity

When a set of heterogeneous computers is shared by multiple independent users, workload on each computer would vary with time and computer. Therefore, workload coupled with the heterogeneous hardware and software components makes availability (computing power) vary spatially and temporally. *Temporal heterogeneity* refers to variation of availability along the time dimension on a computer while *spatial heterogeneity* refers to that among computers as illustrated in Figure 2.1.

With the notations given in the definition of availability above, a computer (C_i) exhibits temporal heterogeneity when $A_i(t)$ is a non-uniform function of time. A system consisting of multiple computers shows spatial heterogeneity when $a_i \neq a_j$ and/or $\sigma_{A_i} \neq \sigma_{A_j}$ for some i, j where $i \neq j$. These two types of heterogeneity will be quantified in Section 2.3.5.

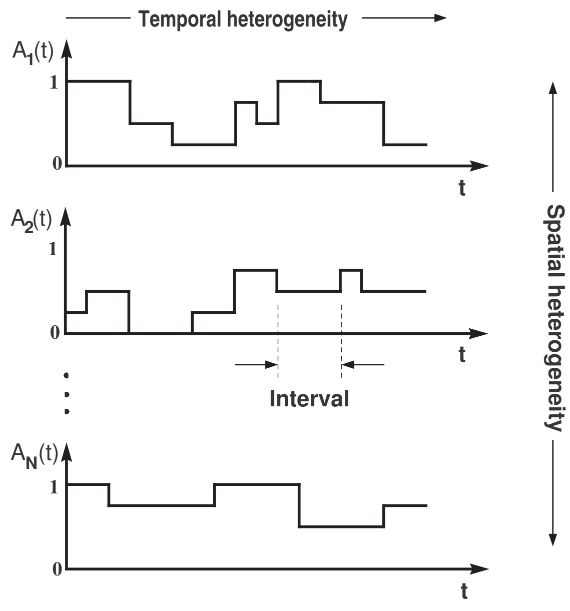


Figure 2.1: Temporal and spatial heterogeneity. $A_i(t)$ is the availability of computing power on C_i for a target task, where $i = 1, \ldots, N$.

The temporal heterogeneity is due to the fact that users submit their tasks at any time and the size of a task is variable, i.e., the task arrival and size are random. There are two sources of the spatial heterogeneity. One is system-related: hardware and software components are not the same for all computers. Also, the interconnection network may not be "symmetric." The other is task-related: the task (workload) characteristics (arrival time and size) may vary with computer.

An interconnection network or networks are usually shared by computers. As a result, the spatial heterogeneity of communication bandwidth tends to be lower than that of computing power.

2.2 Task Model

Two task models are employed in this dissertation: the *base* and *augmented* task models. In the *base* task model, a target task is assumed to be linearly partitionable over N computers, i.e., the sum of computational loads of subtasks is equal to that of the target task. In order to focus on effects of heterogeneous computing power on the performance of a target task, communication is not considered in the base model. Many data-parallel applications such as low-level image processing, (Monte Carlo) simulations, matrix computation, etc. would fit to this model well.

However, in many other applications, as a task is partitioned, communication among subtasks is unavoidable for sharing data and results, collecting local results, etc. The *augmented* task model is derived by incorporating "periodic" communication phases into the base task model. That is, execution of a target task consists of a sequence of alternating computation and communication phases. Hence, a communication phase can be considered as a synchronization point at which all N computers are required to be synchronized for data communication before proceeding to the next (computation) phase. There are many applications which can be represented by this model, e.g., iterative optimization, simulation of a physical or chemical phenomenon, medical image reconstruction, etc. The number of synchronization points is denoted by N_s.

2.3 Performance Measures

Performance of a target task on a heterogeneous computing system may be measured in a few different ways. One popular metric is to use the execution time of the target task, of which minimization is the main objective of high performance computing. In a heterogeneous computing environment, execution time varies significantly due to the temporal and spatial heterogeneity and, therefore, the *average execution time* may be employed as a performance measure. From the stability point of view, one may want to minimize variation of the execution time, in which case the standard deviation of execution time can be used. Another way to quantify stability is to specify the probability that the execution time longer than a certain threshold is obtained. In this section, the performance measures to be employed in this dissertation are derived.

Let's denote the size of a target task by X and the portion of the target task assigned to the computer C_i by X_i. The execution time of X_i on C_i is referred to as T_i, which is a random variable, and its mean and standard deviation are denoted by t_i and σ_{T_i}, respectively. In this dissertation, the same notation is used for a random variable and its observation in order to minimize the number of variables.

2.3.1 Mean and Standard Deviation of Execution Time

Referring to the definition of availability, the relationship between X_i and T_i may be expressed as follows:

$$\int_0^{T_i} \alpha_i A_i(t)\, dt \quad = \quad X_i$$

By taking the expectation on both sides,

$$t_i \quad = \quad \frac{X_i}{\alpha_i a_i} \tag{2.1}$$

Assuming the uncorrelatedness between $A_i(t)$ and $A_i(t')$, the standard deviation, σ_{X_i}, of the work completed during t_i can be shown to be $\sigma_{X_i} = \sqrt{t_i}\alpha_i\sigma_{A_i}$. Then, the standard deviation of T_i may be given by

$$\sigma_{T_i} \;=\; \sqrt{t_i}\frac{\sigma_{A_i}}{a_i} \tag{2.2}$$

The parallel execution time of X in a run, denoted by T, which is also a random variable, is given by $T \;=\; max_i\ \{\ T_i\ \}$ where the notation $\{\ \}$ refers to a set. The mean (τ) and standard deviation (σ_T) of parallel execution time of X are computed as follows:

$$\begin{aligned} \tau &= E[\ T\ ] \\ \sigma_T &= \sqrt{E[\ (T\ -\ \tau)^2\ ]} \end{aligned}$$

where $E[\ ]$ is the expectation operator.

2.3.2 Spatially Homogeneous Environment

When all N computers have the same distribution of execution time (T') and the corresponding *cdf* (cumulative distribution function) is a monotonically increasing function, τ may be expressed as follows (note that the subscript i is not used in this subsection since all computers are identical and that $T_i = T'$ and $X_i = X'$ for all i):

$$\tau \;=\; \int NT'F(T')^{N-1}f(T')\ dT' \tag{2.3}$$

where $F(T')$ and $f(T')$ are the *cdf* and *pdf* (probability density function) of T', respectively.

It is possible to derive τ in the following form [12][43], referring to Equations 2.1 and 2.2:

$$\begin{aligned} \tau &= t + K(N)\,\sigma_{T'} \\ &= t + K(N)\,\sqrt{t}\,\frac{\sigma_A}{a} \\ &= \frac{X'}{\alpha a} + K(N)\,\sqrt{\frac{X'}{\alpha a}}\,\frac{\sigma_A}{a} \end{aligned} \tag{2.4}$$

where $K(N)$ is an increasing function of N and $K(1) = 0$.

Notice that the average parallel execution time of a target task consists of two components, one depending on the mean of availability and the other on the standard deviation of availability, i.e., temporal heterogeneity. It is to be noted that temporal heterogeneity makes the average parallel execution time increase beyond the average (sequential) execution time on each computer. The increase is larger when the number of computers employed is greater. Also, as will be shown later, a higher spatial heterogeneity leads to a longer average parallel execution time.

2.3.3 Synchronization

When the communication phase in the augmented task model does not include data communication, it can be considered as a synchronization point. That is, at each synchronization point, all computers need to wait for the "slowest" computer before they proceed to next computation phase. Suppose that there are N_s synchronization points in a target task, such that the amount of work to be done between two successive synchronization points is $\frac{X'}{N_s}$. In order to extract effects of synchronization only, let's assume that no data communication is actually carried out at each synchronization point. Referring to Equation 2.4, the mean parallel execution time in the jth phase, $\tau^{(j)}$, can be expressed as $\frac{X'}{\alpha a N_s} + K(N)\sqrt{\frac{X'}{\alpha a N_s}\frac{\sigma_A}{a}}$ for $j = 1, \ldots, N_s$. Then, the average parallel execution time τ is $N_s \tau^{(j)}$. Hence,

$$\tau \;=\; \frac{X'}{\alpha a} \;+\; K(N)\;\sqrt{\frac{X' N_s}{\alpha a}\;\frac{\sigma_A}{a}} \tag{2.5}$$

It can be seen that τ increases as N_s increases even though no synchronization overhead (e.g., communication overhead for synchronization) is taken into account.

2.3.4 Stability of Performance

Variation of T may be quantified by its standard deviation, σ_T, which is to be minimized when one wants to achieve a stable performance. Also, one may be interested in knowing the probability, to be referred to as "risk factor," that T is longer than a certain desirable threshold, τ_d. The risk factor, denoted by RF, is given by $Prob[\ T > \tau_d\ ]$.

2.3.5 Temporal and Spatial Heterogeneity

In this section, temporal heterogeneity and spatial heterogeneity are quantified for computing power only, since the definitions would be identical for communication bandwidth. A difference is that spatial heterogeneity would generally be lower for communication bandwidth than for computing power. This is mainly because the communication paths are usually shared by computers.

2.3.5.1 Temporal Heterogeneity

Temporal heterogeneity is defined on an individual computer, which indicates variability of computing power available for a task along the time dimension. Therefore, the standard deviation of the availability may be used to quantify temporal heterogeneity on each computer. Noting that the average availability may vary with computer (and also time) and that τ depends on the ratio of σ_{A_i} to a_i (refer to Equation 2.4), temporal heterogeneity, to be denoted by TH_i, on C_i is defined to be the normalized standard deviation of availability.

$$TH_i \ \triangleq\ \bar{\sigma}_{A_i} \ =\ \frac{\sigma_{A_i}}{a_i} \tag{2.6}$$

The notation TH will be used when TH_i is the same for all i (computers), i.e., spatially homogeneous, or, when the mean of TH_i among all computers is to be referred to.

2.3.5.2 Spatial Heterogeneity

Spatial heterogeneity is defined for a group of computers to be employed to execute a target task. It represents variability of computing power among the computers.

Let's denote the mean and maximum of $\bar{\sigma}_{A_i}$ among C_i by $\bar{\sigma}_A^{mean}$ and $\bar{\sigma}_A^{max}$, respectively, i.e., $\bar{\sigma}_A^{mean} = \frac{1}{N}\sum_{i=1}^{N}\bar{\sigma}_{A_i} = TH^{mean}$ and $\bar{\sigma}_A^{max} = max_i\{\bar{\sigma}_{A_i}\}$. Spatial heterogeneity denoted by SH for a set of computers $\{C_i\}$ is defined as

$$SH \ \triangleq\ \bar{\sigma}_A^{max} \ -\ \bar{\sigma}_A^{mean}. \tag{2.7}$$

That is, SH quantifies variation of temporal heterogeneity among computers.

2.4 Simulation Results and Discussion

2.4.1 Simulation Setup

Availability is assumed to have a uniform distribution (it was observed that other distributions such as a "truncated" Gaussian distribution resulted in similar trends). Then, the distribution of execution time on each computer looks similar to a Gaussian or Gamma distribution which was adopted also in another study [8]. The average availability a_i (or average computing power $\alpha_i a_i$) may vary with computer (C_i). However, a_i (or $\alpha_i a_i$) can be effectively normalized in task partitioning, i.e., load balancing for the average availability (or average computing power) varying with computer can be done easily by assigning X_i, proportional to a_i (or $\alpha_i a_i$), to C_i. Hence, a_i is set to 0.5 with $\alpha_i = 1.0$ for all i in the simulation in order to focus on the effects of heterogeneity in the availability (instead of the average availability or computing power) and to maximize the range of variation of σ_{A_i} (note that $0 \leq a_i \leq 1.0$). Then, the maximum σ_{A_i} ($\bar{\sigma}_{A_i}$) is $\frac{1}{2\sqrt{3}}$ ($\frac{1}{\sqrt{3}}$). The computing power α_i of C_i is set to 1.0 also for normalization purpose.

The notion of "interval" is adopted to quantify the time duration in which availability ($A_i(t)$) remains constant. The interval is mostly affected by other tasks (distributions of their arrival time and size) and the local scheduler on C_i and is to be modelled by a random variable. Note that decreasing the interval given a fixed task size is equivalent to increasing the task size with the interval fixed, and vice versa. A larger interval (for a given task size) leads to a higher chance for load imbalance among computers. The interval is generated from a uniform distribution of which range is [0,20] except when it is varied.

Simulation was repeated 1000 times for each case and results were averaged for the graphs in the following sections.

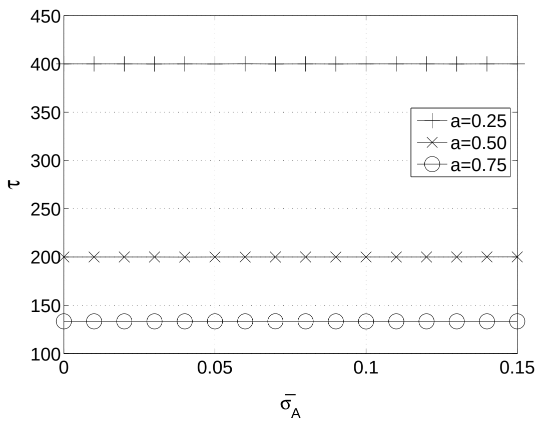
2.4.2 Effects of Heterogeneity on a Target Task

In this section, some of the simulation results are presented to discuss effects of temporal and spatial heterogeneity on the performance of a target computing task. In the graphs where variation of the parallel execution time (T) is analyzed, the standard deviation of T normalized by τ is used, i.e., $\frac{\sigma_T}{\tau} \overset{\triangle}{=} \bar{\sigma}_T$. The results without communication overhead (but with synchronization in some cases) are provided in Figures 2.2 - 2.11, and those with communication overhead are in Figures 2.12 and 2.13.

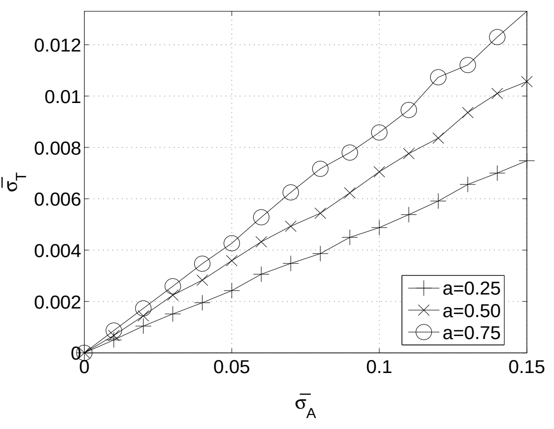
2.4.2.1 Spatially Homogeneous and Temporally Heterogeneous

In Figure 2.2, τ and $\bar{\sigma}_T$ are plotted when $N = 1$. As can be seen in the figure, it is clear that the average sequential execution time is not affected by $\bar{\sigma}_A$, but its variation increases proportional to $\bar{\sigma}_A$, as shown in Equations 2.1 and 2.2.

In Figure 2.3, effects of $\bar{\sigma}_A$ on τ and $\bar{\sigma}_T$ are shown for different values of N. As predicted in Equation 2.4, τ and $\bar{\sigma}_T$ increase almost linearly proportional to $\bar{\sigma}_A$ (which is TH) when multiple computers are employed. When more computers are employed (a larger N), the effect of $\bar{\sigma}_A$ on τ is larger, as shown in Figure 2.3-(a). The effect of $\bar{\sigma}_A$ on σ_T is also larger. However, since τ increases faster than σ_T does, the effect of $\bar{\sigma}_A$ on $\bar{\sigma}_T$ is smaller for a larger N, as shown in Figure 2.3-(b).

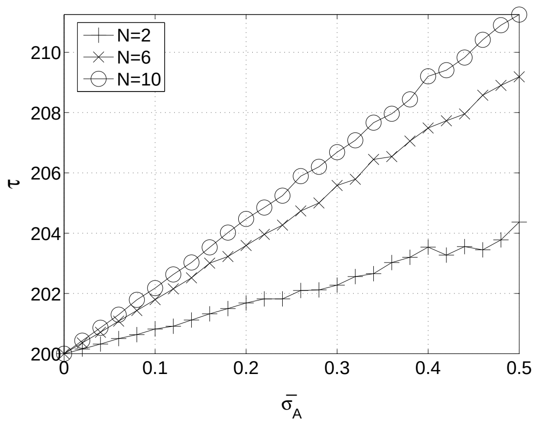


(a)

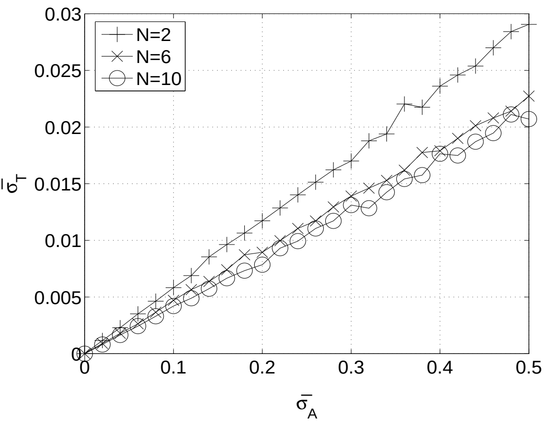


(b)

Figure 2.2: (a) Average execution time and (b) normalized standard deviation of execution time on a single computer where $X = 100$.



(a)



(b)

Figure 2.3: (a) Average parallel execution time and (b) normalized standard deviation of parallel execution time on N computers where $X_i = 100$ for $i = 1, \ldots, N$.

2.4.2.2 Spatially and Temporally Heterogeneous

In Figure 2.4, dependency of τ and $\bar{\sigma}_T$ on SH and TH is shown when N is fixed to 8. In this graph, when SH is zero (i.e., on the TH axis), $\bar{\sigma}_A$, which is TH_i, is the same for all computers. When SH is greater than zero, distribution of $\bar{\sigma}_{A_i}$ among computers is *linear* such that $\bar{\sigma}_{A_i} = \bar{\sigma}_A^{mean} + 2(\frac{i-1}{N-1} - 0.5)(\bar{\sigma}_A^{max} - \bar{\sigma}_A^{mean})$ for $i = 1, \ldots, N$. That is, $TH = \bar{\sigma}_A^{mean}$ in these graphs. As SH increases, τ increases significantly, especially when $\bar{\sigma}_A^{mean}$ also increases (going diagonally from the origin on the $SH - TH$ plane).

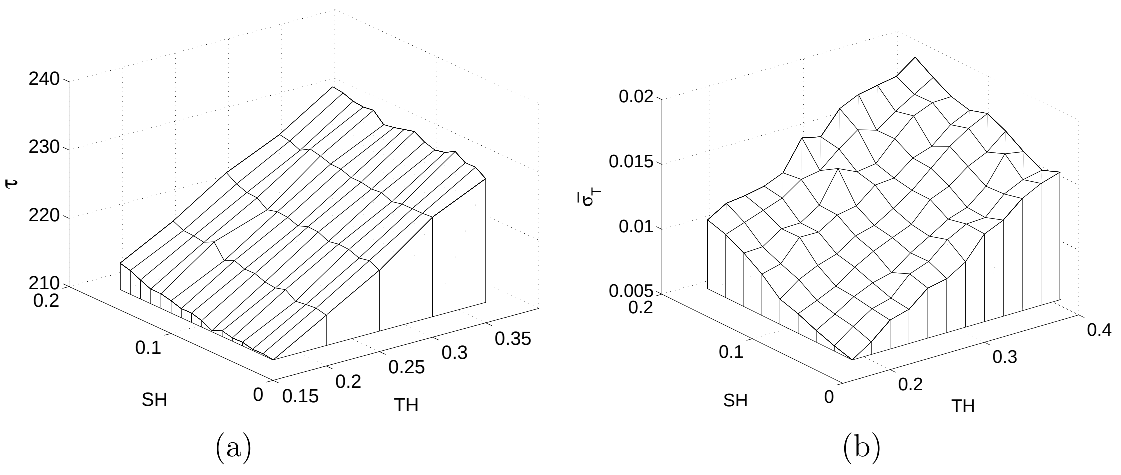
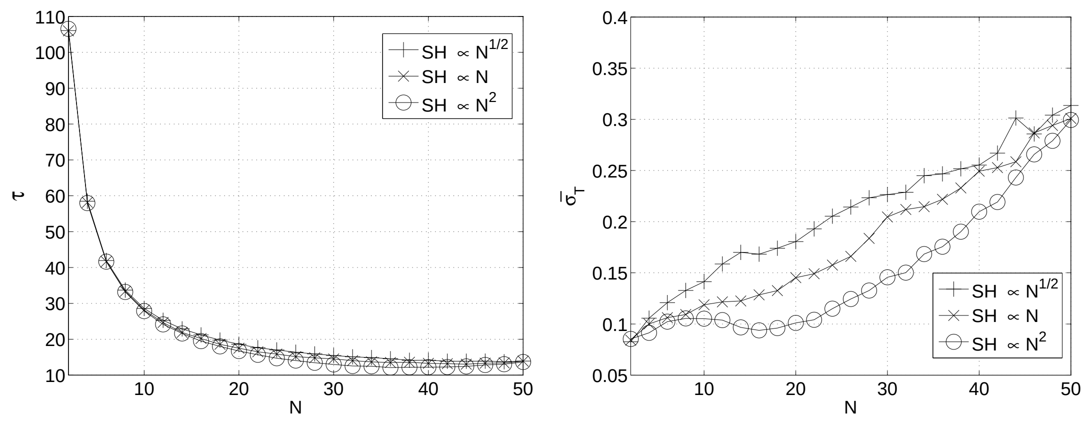


Figure 2.4: (a) Average execution time and (b) normalized standard deviation of execution time when SH and TH are varied with N fixed at 8 where $X_i = 100$ for $i = 1, \ldots, 8$.

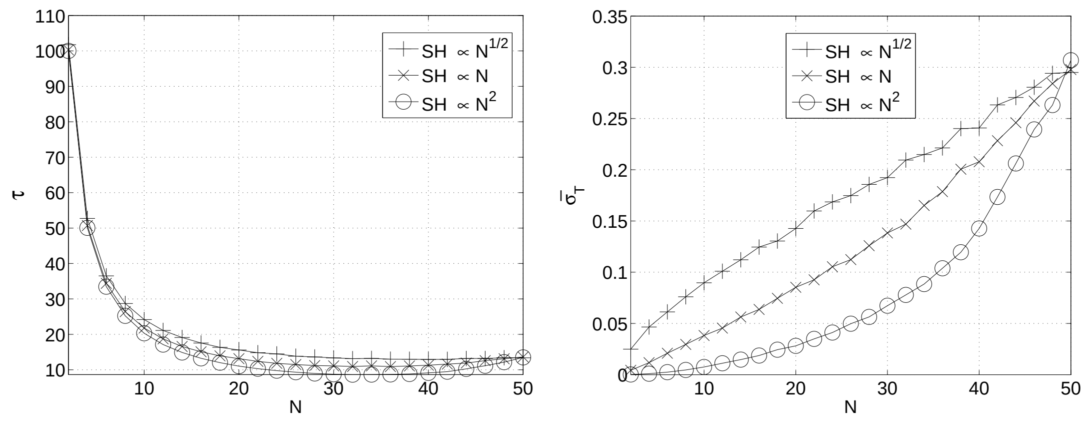
Now, suppose that the size of a target task, X, is fixed independent of N and it is uniformly distributed over N computers. The larger the set of computers employed for the target task is, the larger heterogeneity (SH) among computers becomes in general. Two cases are considered: (i) when $\bar{\sigma}_A^{mean}$ is fixed while SH increases and (ii) when both of SH and $\bar{\sigma}_A^{mean}$ increase. In both cases, three different situations, in terms of how $\bar{\sigma}_A^{max}$ is increased, are considered: proportional to $\sqrt{N}$, N, and N^2. Simulation results for cases (i) and (ii) are provided in Figures 2.5 and 2.6, respectively.

As N increases, τ decreases almost inversely proportional to N and then starts to increase beyond a certain N especially when SH increases rapidly. In contrast, $\bar{\sigma}_T$ monotonically increases as N increases. It increases sharply after a certain value of N in the case that SH increases proportional to N^2.



(a) (b)

Figure 2.5: (a) Average execution time and (b) normalized standard deviation of execution time. $X_i = \frac{X}{N}$ for $i = 1, \ldots, N$ and $X = 100$. As N increases, $\bar{\sigma}_A^{mean}$ remains fixed.



(a) (b)

Figure 2.6: (a) Average execution time and (b) normalized standard deviation of execution time. $X_i = \frac{X}{N}$ for $i = 1, \ldots, N$ and $X = 100$. As N increases, $\bar{\sigma}_A^{mean}$ increases.

Speed-up is shown for the cases of (i) and (ii) in Figure 2.7-(a) and Figure 2.7-(b), respectively. The reason why the three curves meet when $N = 50$ is that the simulation was set up such that the distribution of $\bar{\sigma}_A$ among computers becomes identical in all three situations when N is increased to 50. Hence, what is to be observed in these graphs is the "shape" (trend) of each curve. It is clear that spatial heterogeneity SH alone keeps one from employing more than a certain number of computers even for an embarrassingly parallel task. Also, it can be induced that the complexity of the function $K(N)$ in Equation 2.4 is at least $O(\sqrt{N})$.

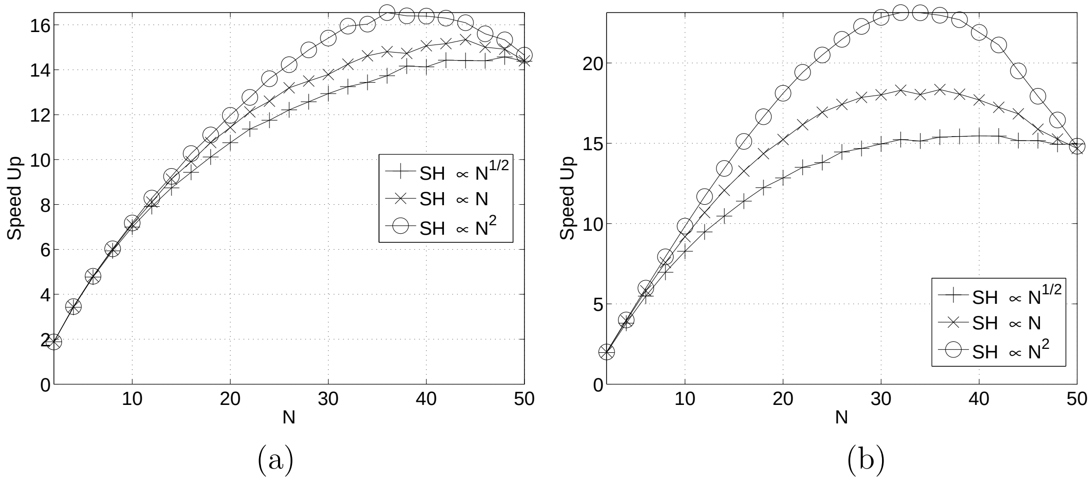


Figure 2.7: Speed-up when (a) $\bar{\sigma}_A^{mean}$ remains fixed and (b) $\bar{\sigma}_A^{mean}$ increases as N increases. $X_i = \frac{X}{N}$ for $i = 1, \ldots, N$. $X = 100$.

2.4.2.3 Synchronization

The number of synchronization points, N_s, is varied for different $\bar{\sigma}_{A_i}$, to observe its effects on τ and $\bar{\sigma}_T$ when $SH = 0$ in Figure 2.8 and when $SH > 0$ in Figure 2.9. In Figure 2.8, it is seen that as N_s increases τ increases since the idle time due to synchronization increases. However, $\bar{\sigma}_T$ decreases. This can be explained as follows. The increased T_i (or t_i) due to a larger N_s leads to the increased σ_T (refer to Equation 2.2). However, T_i increases faster than σ_T, causing $\bar{\sigma}_T$ to decrease. The increase rate in τ is larger for a larger $\bar{\sigma}_A$ (TH) and a higher SH, as shown in Figure 2.9.

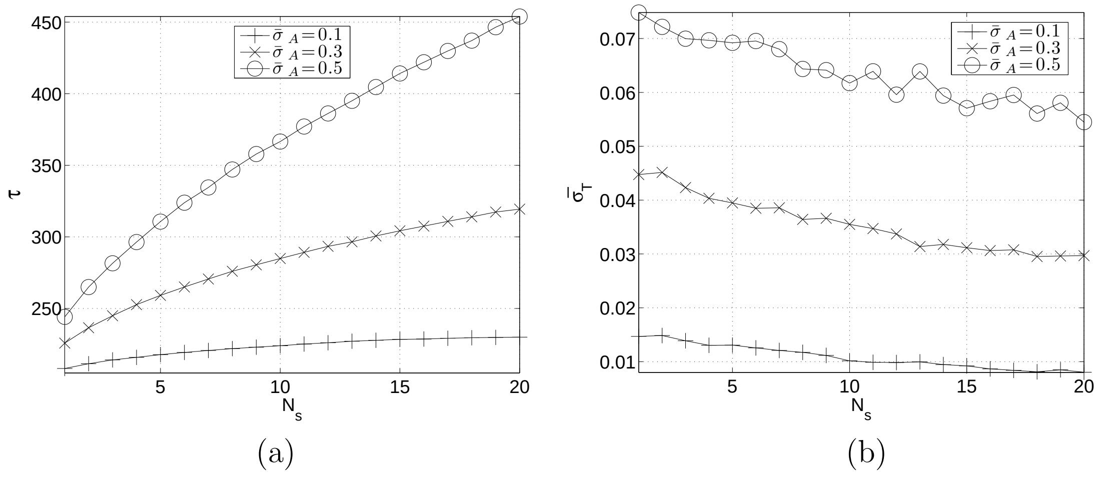


Figure 2.8: (a) Average execution time and (b) normalized standard deviation of execution time as functions of N_s. $X = 1000$, $N = 10$, and $X_i = \frac{X}{N} = 100$.

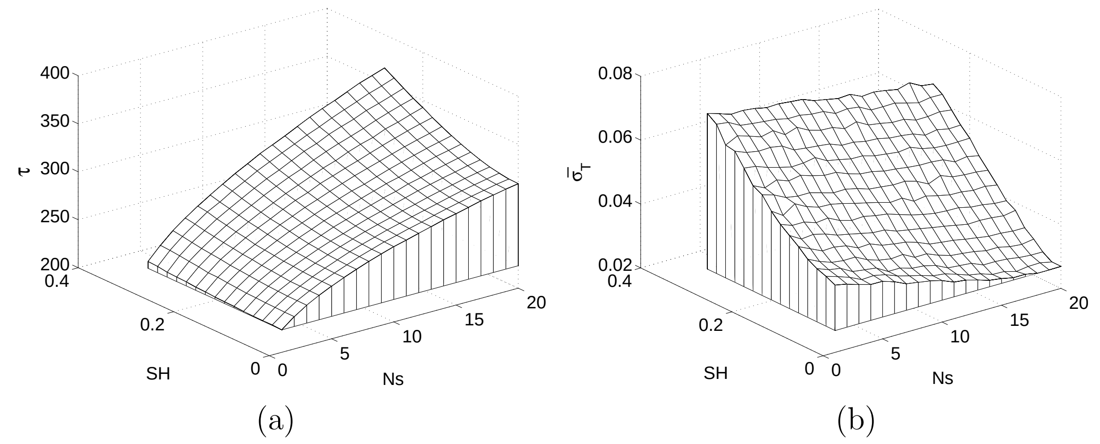


Figure 2.9: (a) Average parallel execution time and (b) normalized standard deviation of parallel execution time as functions of SH and N_s. $X = 1000$, $N = 10$, $X_i = \frac{X}{N} = 100$, and $\bar{\sigma}_A^{mean} = 0.28$.

2.4.2.4 Granularity of Availability

When the size of a target task (in terms of its execution time) is much larger than an *interval* (the duration when availability remains constant, refer to Section 2.4.1) (equivalently, the *interval* is much smaller than the target task size), effects of heterogeneity (temporal or spatial) are reduced since the probability of load imbalance decreases. Dependency on interval is shown when $SH = 0$ in Figure 2.10 and when $SH > 0$ in Figure 2.11. As the *interval* increases (i.e., availability varies more slowly with time) for a given target task, effects of TH and SH become larger making τ and $\bar{\sigma}_T$ larger. Equivalently, a smaller task would be more sensitive to heterogeneity.

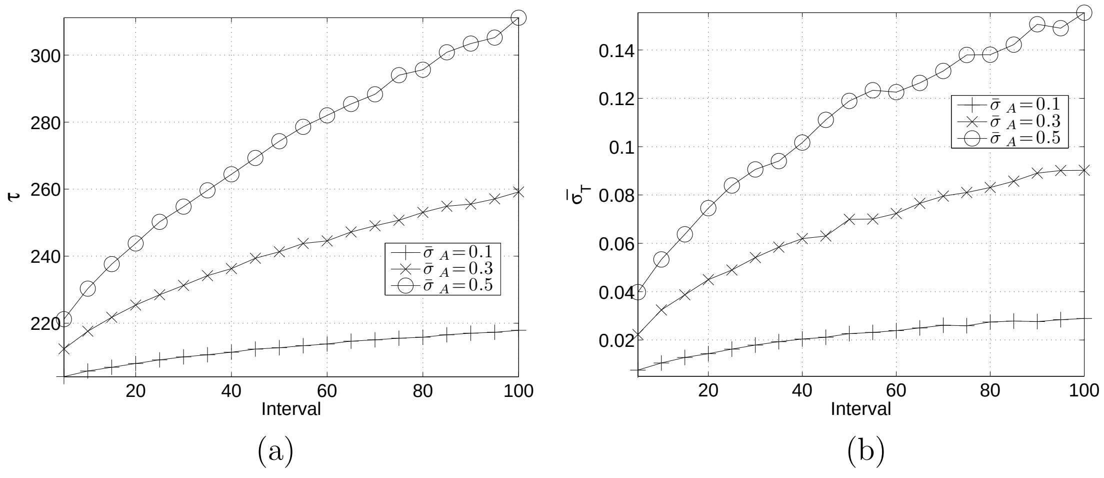


Figure 2.10: (a) Average execution time and (b) normalized standard deviation of execution time as functions of interval. $X = 1000$, $N = 10$, and $X_i = \frac{X}{N} = 100$.

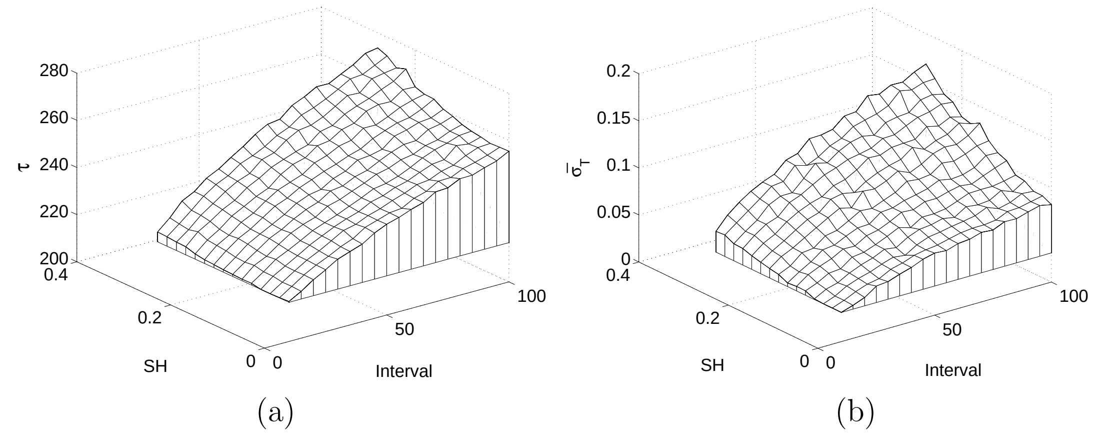
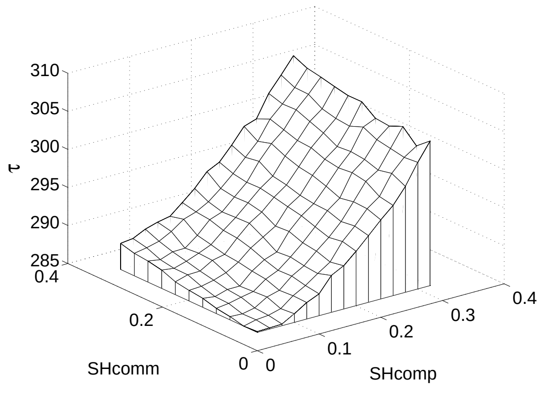


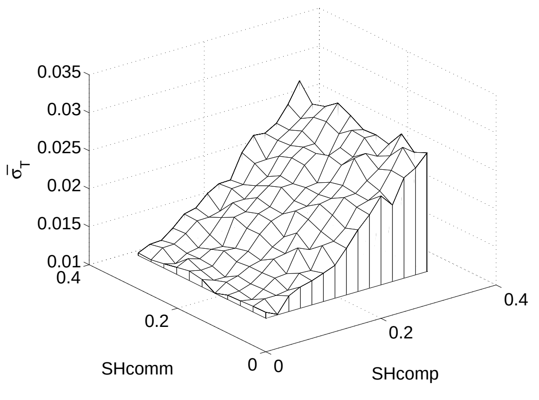
Figure 2.11: (a) Average parallel execution time and (b) normalized standard deviation of parallel execution time as functions of SH and interval. $X = 1000$, $N = 10$, $X_i = \frac{X}{N} = 100$, and $\bar{\sigma}_A^{mean} = 0.28$.

2.4.2.5 Communication Overhead

Simulation results when the augmented task model is adopted are provided in Figures 2.12 and 2.13. In these graphs, τ includes both computation and communication times. In Figure 2.12, SH_{comp} and SH_{comm} represent the spatial heterogeneity of availability in computing power and communication bandwidth, respectively. As expected for the simulation model, SH_{comp} and SH_{comm} have the similar effect on τ and $\bar{\sigma}_T$ (refer to Section 2.4.2.2). In this simulation, the ratio of computation time to communication overhead is set to 6. That is why τ increases more slowly along SH_{comm} than SH_{comp}.



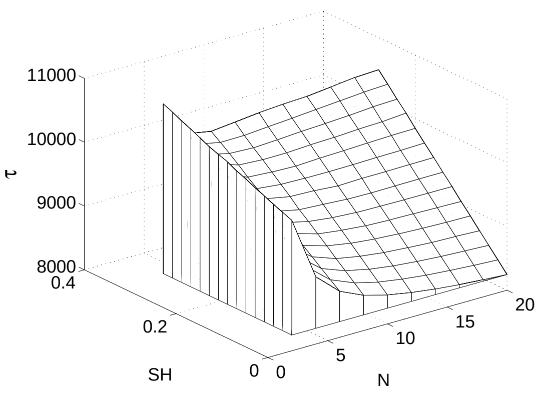
(a)



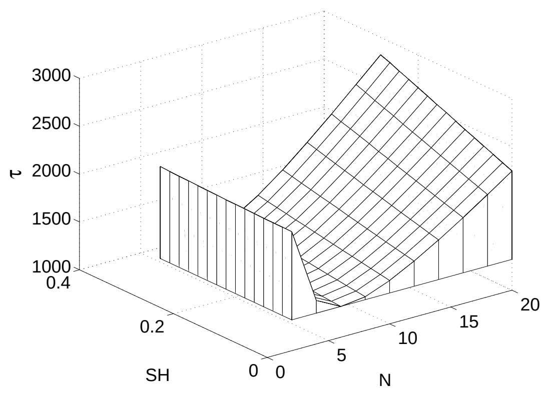
(b)

Figure 2.12: (a) Average execution time (computation and communication times) and (b) normalized standard deviation of execution time as functions of SH_{comm} and SH_{comp}. $N = 20$, $X = 2000$, $Ns = 10$, $X_i = \frac{X}{N} = 100$, and $\bar{\sigma}_A^{mean} = \bar{\sigma}_B^{mean} = 0.28$.

In Figure 2.13, SH is the same for both the computing power availability and bandwidth availability. As N increases, τ decreases initially and then turns around to increase after a certain value of N. This is mainly because communication overhead becomes dominant as N increases. It is also observed that the trend is more visible when the bandwidth availability (and computing power availability) shows a higher



(a)



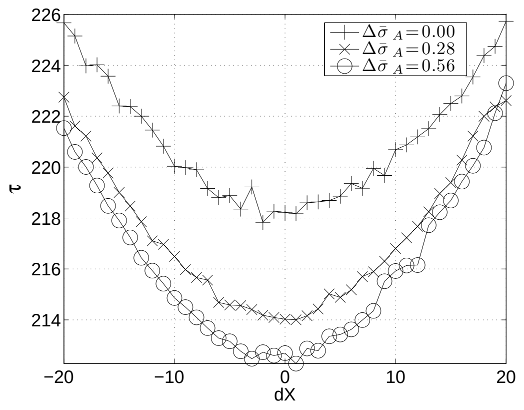
(b)

Figure 2.13: Average execution time: (a) when communication overhead is independent of N and $X = 400$, and (b) when communication overhead increases linearly proportional to N and $X = 2000$. $Ns = 10$ and $\bar{\sigma}_A^{mean} = \bar{\sigma}_B^{mean} = 0.28$.

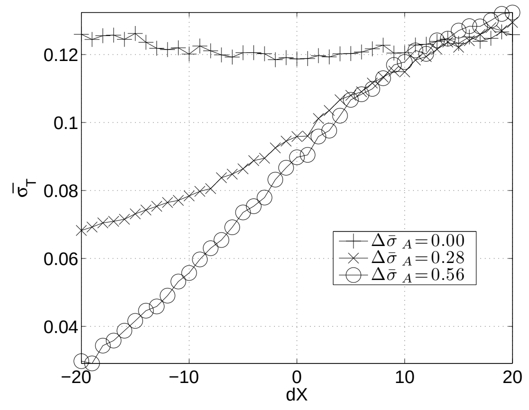
spatial heterogeneity (SH). For example, in Figure 2.13-(a), it is seen that τ turns around to increase when SH is relatively high while it monotonically decreases when $SH = 0$.

2.4.2.6 Load Distribution and Stability (Risk Factor)

So far, only the cases where a target task is uniformly distributed among computers have been discussed. Let's examine the effect of distribution of the target task with SH varied. In Figure 2.14, τ and $\bar{\sigma}_T$ are plotted as functions of dX with $N = 2$ where $dX = \frac{X_1 - X_2}{2}$ and $\Delta\bar{\sigma}_A = \bar{\sigma}_{A_1} - \bar{\sigma}_{A_2}$. In Figure 2.15, the risk factor defined in Section 2.3.4 is shown as a function of dX. The same set of results are provided when $N = 10$ in Figures 2.16 and 2.17. In these simulations, $\bar{\sigma}_{A_i}$ is increased *linearly* with i among 10 computers as before (refer to Figure 2.4). X_i is also varied linearly with respect to i, i.e., $X_i = X_0 + Slope_{X_i}(i-1)$ for $i = 1, \ldots, N$. Therefore, a negative slope means that a computer with a smaller $\bar{\sigma}_{A_i}$ is assigned a larger fraction of X (a larger X_i).



(a)



(b)

Figure 2.14: (a) Average execution time and (b) normalized standard deviation of execution time on two computers where $X_1 = \frac{X}{2} - dX$, $X_2 = \frac{X}{2} + dX$, and $X = 100$.

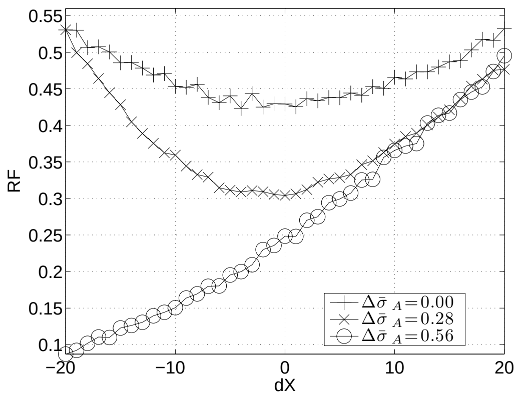
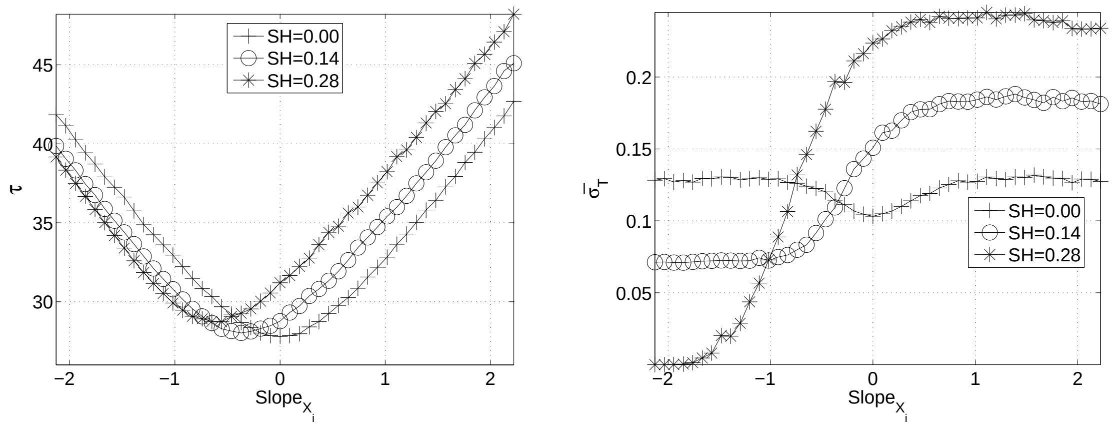


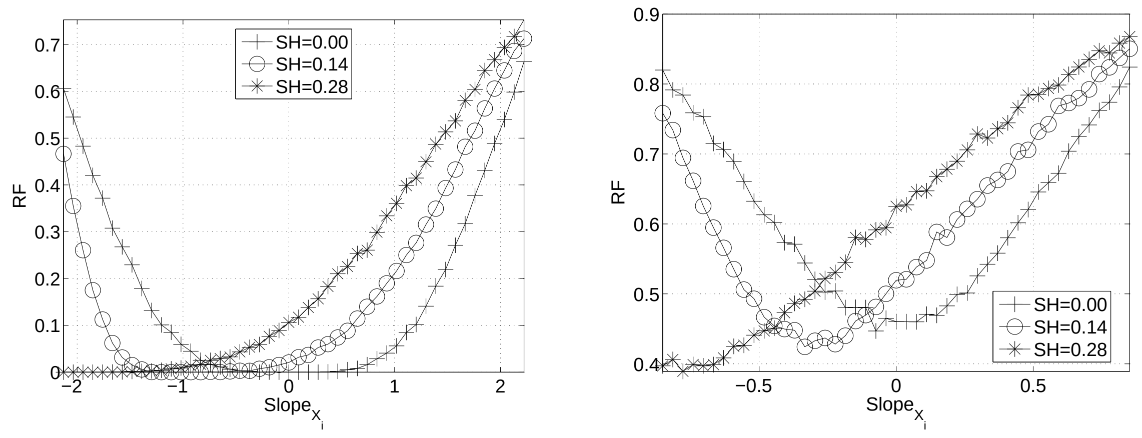
Figure 2.15: Risk factor on two computers where $X_1 = \frac{X}{2} - dX$, $X_2 = \frac{X}{2} + dX$, and $X = 200$. $\tau_d = 221$. $\Delta\bar{\sigma}_A = \bar{\sigma}_{A_2} - \bar{\sigma}_{A_1}$.

It may be observed from these results that, by assigning more work (a larger fraction of target task) to a computer with a smaller variation of availability, (a) a shorter (minimum) τ is obtained, (b) $\bar{\sigma}_T$ is sharply decreased, and (c) the risk factor is quickly reduced. These trends are more explicit for a larger N and a higher SH. It is also seen that the amount of work to be assigned to each computer to minimize τ, $\bar{\sigma}_T$, and RF depends on SH (and the distribution of $\bar{\sigma}_{A_i}$). That is, for achieving the minimum execution time on a spatially heterogeneous computing system, it is not



(a) (b)

Figure 2.16: (a) Average execution time and (b) normalized standard deviation of execution time on 10 computers. $X = 100$ and $X_i = X_0 + Slope_{X_i}(i-1)$ for $i = 1, \ldots, 10$.



(a) (b)

Figure 2.17: Risk factor on 10 computers (a) $\tau_d = 40$ and (b) $\tau_d = 28$. $X = 100$ and $X_i = X_0 + Slope_{X_i}(i-1)$ for $i = 1, \ldots, 10$.

optimal to distribute a target task linearly proportional to the *mean* availabilities of computers.

Chapter 3

Load Balancing on a Heterogeneous Computing System

The results in Chapter 2 reveal that, for achieving the minimum average parallel execution time on a spatially heterogeneous computing system, it is not optimal to distribute a target task linearly proportional to the *average* availabilities (or *average* computing powers) of computers. This observation has motivated the development of an efficient load balancing scheme for heterogeneous cluster and grid computing environments, which is described in this chapter.

3.1 Two-Step Heuristic Approach

Let's denote the computing power (*speed*) of C_i at t by $S_i(t)$ which has the mean s_i and the standard deviation σ_{S_i}. Then, noting that $S_i(t) = \alpha_i A_i(t)$, $s_i = \alpha_i a_i$ and $\sigma_{S_i} = \alpha_i \sigma_{A_i}$. When $\sigma_{S_i} = 0$ for all i, a target task is to be partitioned proportional to s_i, in order to achieve the minimum τ. However, such partitioning does not lead to the minimum τ in general when $\sigma_{S_i} \neq 0$ for some i.

Let's consider cases where $\sigma_{S_i} \neq 0$ for some i. Suppose that X is partitioned such that X_i is linearly proportional to s_i. Then, t_i (the average execution time on C_i) would be the same for all i, but σ_{T_i} (the standard deviation of execution time on C_i) would not be. It is to be pointed out that σ_{T_i} is linearly proportional to $\bar{\sigma}_{A_i}$ (or TH_i). Noting that τ is given by $E[max_i\{T_i\}]$ rather than $max_i\{T_i\}$ or $max_i\{t_i\}$, it is possible to further reduce τ by taking $\{\sigma_{T_i}\}$ into account. Therefore, load balancing may be carried out in two steps as follows:

(1) X is partitioned over $\{C_i\}$ such that X_i is proportional to s_i,

(2) $\{X_i\}$ is further adjusted considering $\{\sigma_{T_i}\}$ and $\{s_i\}$.

In the following, this two-step load balancing approach is elaborated for different cases.

3.1.1 When $s_i = s_j$ for all i, j:

Consider the case of $N = 2$ (two computers). In Step (1) of the two-step load balancing approach, X is divided equally between C_1 and C_2 since $s_1 = s_2$. That is, after Step (1), $X_1 = X_2 = \frac{X}{2}$ and $t_1 = t_2$. Suppose that $\sigma_{T_1} > \sigma_{T_2}$ after Step (1). Then, it is possible to reduce τ further by transferring a certain amount of work ($\Delta X \geq 0$) from C_1 to C_2 in Step (2), i.e., $X_1' = X_1 - \Delta X$ and $X_2' = X_2 + \Delta X$, where X_i' is X_i after Step (2). Then,

$$t_1' = t_1 - \frac{\Delta X}{s_1} \triangleq t_1 - \Delta t_1 \quad \text{and} \quad t_2' = t_2 + \frac{\Delta X}{s_2} \triangleq t_2 + \Delta t_2 \tag{3.1}$$

where t_i' is t_i after Step (2).

Note that $\Delta t_1 = \Delta t_2$ since $s_1 = s_2$. Also, from Equation 2.2, it can be shown that

$$\sigma_{T_1}' = \sigma_{T_1}\sqrt{1 - \frac{\Delta X}{s_1 t_1}} \quad \text{and} \quad \sigma_{T_2}' = \sigma_{T_2}\sqrt{1 + \frac{\Delta X}{s_2 t_2}} \tag{3.2}$$

where σ_{T_i}' is σ_{T_i} after Step (2).

A heuristic scheme, to be referred to as *equalization scheme*, that can be employed in Step (2) equalizes the sum of the mean and standard deviation of execution time between two computers. That is, it determines ΔX such that

$$t'_1 + \sigma'_{T_1} = t'_2 + \sigma'_{T_2} \tag{3.3}$$

where t_i and σ'_{T_i} are given by Equations 3.1 and 3.2, respectively.

The equalization scheme is illustrated in Figure 3.1. The scheme attempts to reduce the probability that the parallel execution time of a target task, $T = max_i\{T_i\}$, is large, in order to minimize its average parallel execution time. This heuristic can be generalized for cases where $N > 2$, i.e., X_i is determined such that $t'_i + \sigma'_{T_i}$ is the same for all i.

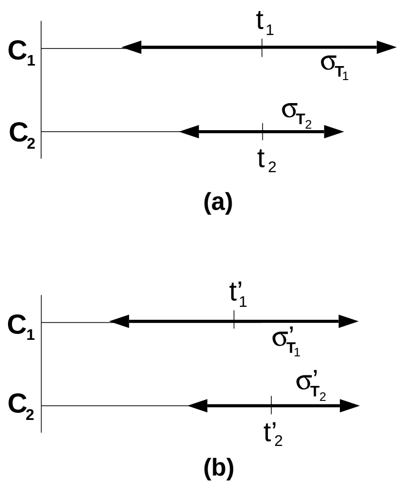


Figure 3.1: Equalization scheme: (a) after Step (1) where a target task is partitioned such that the average execution time is the same on all computers and (b) after Step (2) where the partitioning is adjusted such that $t'_i + \sigma'_{T_i}$ is equalized for all computers.

3.1.2 When $s_i \neq s_j$ for some i, j:

Again, consider the case of $N = 2$, and suppose that $s_1 < s_2$. After Step (1), $X_1 = \frac{s_1 X}{s_1+s_2}$ and $X_2 = \frac{s_2 X}{s_1+s_2}$. What is to be done in Step (2) to further reduce τ depends on $\{\sigma_{T_i}\}$ and $\{s_i\}$, as discussed below, and can be generalized for cases where $N > 2$.

(a) $\sigma_{T_1} > \sigma_{T_2}$: In this case, ΔX is to be moved from C_1 to C_2 and the equalization scheme may be employed in determining ΔX. One difference is that $\Delta t_1 > \Delta t_2$. In other words, the same increase in t_2 (by moving ΔX from C_1 to C_2) results in a larger decrease in t_1, leading to a larger reduction in τ, compared to the case where $s_1 = s_2$.

(b) $\sigma_{T_1} < \sigma_{T_2}$ and $s_1 \ll s_2$: It is still possible to reduce τ by transferring ΔX from C_1 to C_2 though the equalization scheme may not be used since the condition, $t'_1 + \sigma'_{T_1} = t'_2 + \sigma'_{T_2}$, cannot be satisfied. Reduction in τ would be smaller than that in (a).

(c) $\sigma_{T_1} < \sigma_{T_2}$ and $s_1 \simeq s_2$: In this case, ΔX is to be moved from C_2 to C_1 and the equalization scheme can be used. Reduction in τ would be smaller than that in (a) or (b).

It should be clear that transferring ΔX in the other direction than specified above would result in a longer τ. The above discussion on reduction in τ for different cases is summarized in Table 3.1.

	$s_1 = s_2$	$s_1 < s_2$		
	$\sigma_{T_1} > \sigma_{T_2}$	$\sigma_{T_1} > \sigma_{T_2}$	$\sigma_{T_1} < \sigma_{T_2}$ $(s_1 << s_2)$	$\sigma_{T_1} < \sigma_{T_2}$ $(s_1 \simeq s_2)$
Direction of ΔX	$C_1 \rightarrow C_2$	$C_1 \rightarrow C_2$	$C_1 \rightarrow C_2$	$C_1 \leftarrow C_2$
Reduction in τ	Smallest	Largest	Small	Small

Table 3.1: Possible reduction in τ

3.2 Simulation Results and Discussion

Let's first consider a case where the average computing speed is the same for all computers, i.e., $s_i = s_j$ for all i, j. First, $X_i = X_j$ for all i, j in Step (1) of the two-step load balancing approach described in Section 3.1. It is assumed that $t_i = 100$ and $\sigma_{T_i} = \frac{i-1}{N-1}\sigma_{max}$, for $i = 1, \ldots, N$, after Step (1). In Step (2), X_i is adjusted such that $t'_i = t_i + Slope_T(i - \frac{N-1}{2})$ for $i = 1, \ldots, N$. $Slope_T = 0$ corresponds to the case where X'_i is proportional to s_i (the *average* computing speed or power), i.e., Step (1) only. A negative $Slope_T$ indicates that a computer with a larger σ_{T_i} is assigned a larger ΔX_i (refer to Section 3.1) in Step (2) in addition to X_i allocated in Step (1).

In Figure 3.2, τ and *percentage standard deviation* of $(t'_i + \sigma'_{T_i})$ are shown as functions of $Slope_T$ which specifies how a target task is distributed over N computers. The percentage standard deviation of $(t'_i + \sigma'_{T_i})$ is a measure which reflects the degree of equalization, and is defined as:

$$\% \text{ standard deviation of } (t'_i + \sigma'_{T_i}) \quad = \quad \frac{\sigma_{(t'+\sigma'_T)}}{(t' + \sigma'_T)} \times 100\% \tag{3.4}$$

where $\overline{(t' + \sigma'_T)}$ and $\sigma_{(t'+\sigma'_T)}$ are average and standard deviation of $(t'_i + \sigma'_{T_i})$ over N computers:

$$\overline{(t' + \sigma'_T)} = \frac{\sum_{i=1}^{N} (t'_i + \sigma'_{T_i})}{N} \tag{3.5}$$

$$\sigma_{(t'+\sigma'_T)} = \sqrt{\frac{\sum_{i=1}^{N}(t'_i + \sigma'_{T_i} - \overline{t' + \sigma'_T})^2}{N}} \tag{3.6}$$

First of all, it is to be noted from Figure 3.2 that it is not optimal to distribute a target task linearly proportional to the *average* computing power of computers. In Figure 3.2-(a), it is clear that the performance improvement (reduction in τ) achieved by Step (2) (more precisely, Step (1) + Step (2)) over Step (1) is significant, and is larger for a larger $\Delta\sigma_T$ (equivalently, a higher spatial heterogeneity). Comparing Figures 3.2-(a) and (b), it can be seen that the equalization scheme employed in Step (2) works well, i.e., the distribution of X minimizing τ closely matches with that minimizing variation in $t_i + \sigma_{T_i}$.

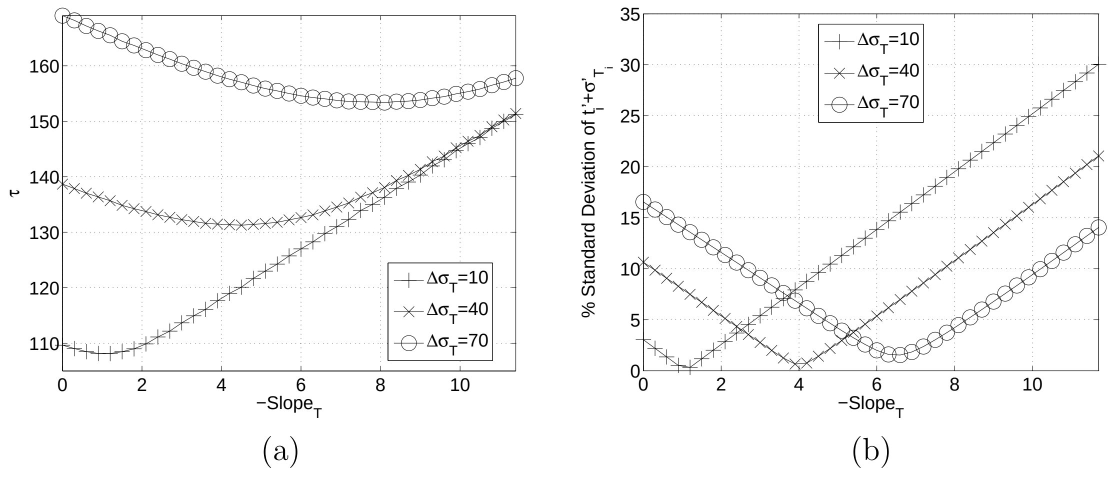


Figure 3.2: (a) Average parallel execution time and (b) Percentage standard deviation of $t'_i + \sigma'_{T_i}$ after Step (2) in the load balancing, when the number of computers, N, is 10. After Step (1), $t_i = 100$ and $\Delta\sigma_T = \sigma_{T_N} - \sigma_{T_1}$ where $\sigma_{T_i} = \frac{i-1}{N-1}\sigma_{Tmax}$, where $i = 1, \ldots, 10$.

In Figure 3.3, a system of two computers where $s_1 < s_2$ is considered. In these graphs, τ is plotted as a function of ΔX. Again, $\Delta X = 0$ corresponds to the cases where X is divided between two computers proportional to their average computing powers (s_1 and s_2). It can be observed that reduction in τ, which can be achieved by Step (2), is larger for a larger s or a larger $\Delta\sigma_T$. Let's define the percentage error (ε) of the equalization scheme as $\frac{\Delta\tau_{max}-\Delta\tau}{\Delta\tau_{max}} \times 100$ where $\Delta\tau_{max}$ and $\Delta\tau$ are the maximum possible and achieved (by the heuristic) reductions in τ, respectively. In this set of results, the average ε was 5.1%.

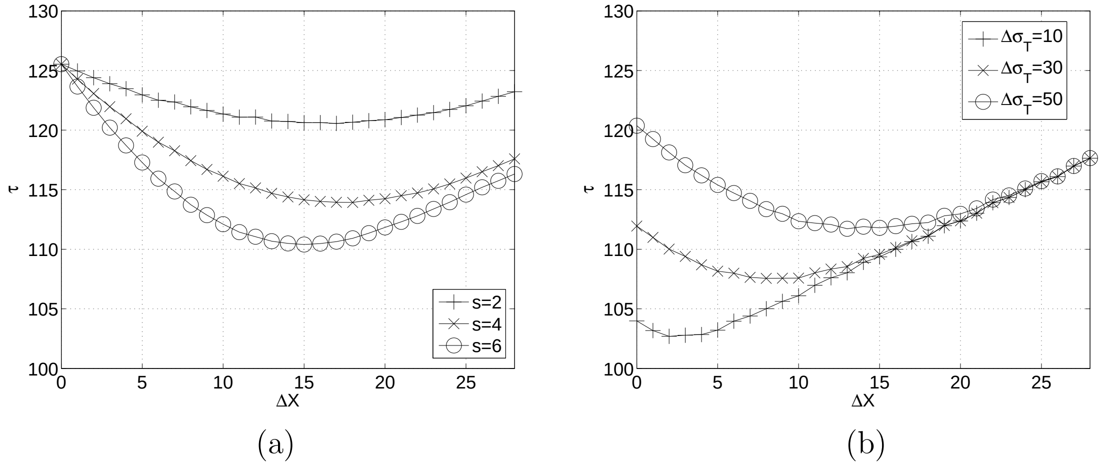
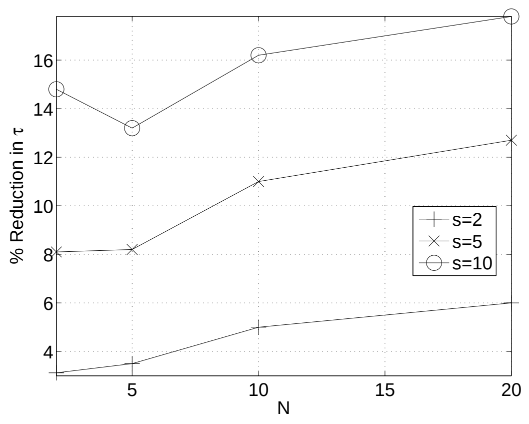
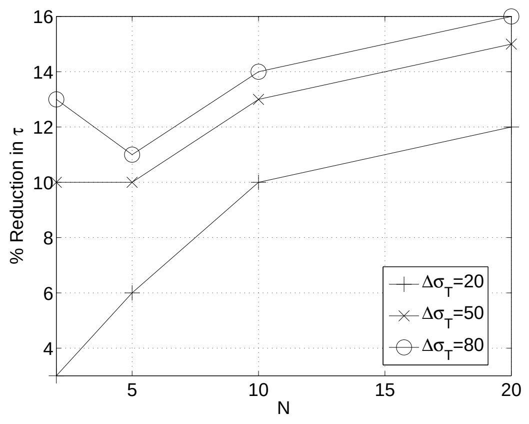


Figure 3.3: Average parallel execution time on two computers (a) with a fixed $\Delta\sigma_T = \sigma_{T_1} - \sigma_{T_2} = 50$ ($\sigma_{T_2} = 10$) and (b) with a fixed $s = 4$, where $s = \frac{s_2}{s_1}$ ($s_1 = 1$) and $\Delta\sigma_T = \sigma_{T_1} - \sigma_{T_2}$ ($\sigma_{T_2} = 0$).

In Figure 3.4, percentage reduction in τ, which is achieved by Step (2), is analyzed with the number of computers (N) varied. In this simulation, s_i increases and σ_{T_i} decreases linearly proportional to i, i.e., a faster computer has a smaller variation of execution time. The *percentage reduction in* τ is defined to be $\frac{\tau_{(1)}-\tau_{(2)}}{\tau_{(1)}} \times 100$ where $\tau_{(1)}$ and $\tau_{(2)}$ are τ achieved by Steps (1) and (2), respectively. It can be seen that the percentage reduction increases as N increases. That is, the performance improvement



(a)



(b)

Figure 3.4: Percentage reduction in average parallel execution time as a function of the number of computers (a) with a fixed $\Delta\sigma_T = \sigma_{T_1} - \sigma_{T_N} = 60$ ($\sigma_{T_N} = 0$) and (b) with a fixed $s = 5$, where $s = \frac{s_N}{s_1}$ ($s_1 = 1$) and $\Delta\sigma_T = \sigma_{T_1} - \sigma_{T_N}$ ($\sigma_{T_N} = 0$). After Step (1), $t_i = 100$ for all i.

by Step (2) becomes greater when a larger number of computers are employed. In Figure 3.4-(a), $\Delta\sigma_T$ is fixed independent of N. However, in reality, $\Delta\sigma_T$ would usually increase as N increases. Therefore, in such a case, one may expect a larger reduction in τ. Note that a larger $\Delta\sigma_T$ leads to a greater reduction in τ as shown in Figure 3.4-(b).

Chapter 4

Effects of Heterogeneity on an Individual Network Traffic

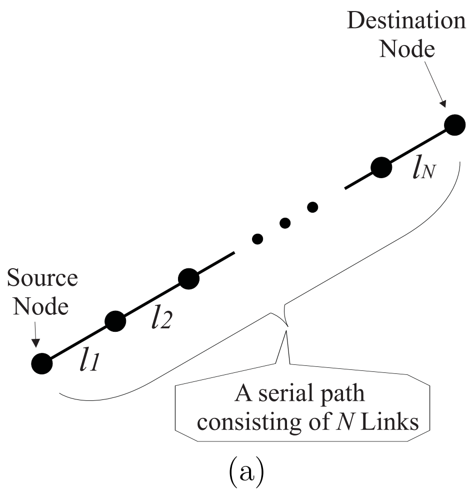
In this chapter, how heterogeneity in channel bandwidth affects the performance of an individual message is analyzed. Also, application of the analysis results to path selection and multi-path data transfer are considered.

4.1 Network Model

4.1.1 Topology

In this dissertation, a communication network is represented by a graph where a node may be a user, a switch, a router, etc. and an edge is a wired or wireless channel or link. Bandwidth of a channel refers to the bandwidth that is available for or can be allocated to an individual message (request) on the channel. A *path* is a set of consecutive edges (links) from one node to another. Two types of paths are of interest, *serial* path and *parallel* path.

A serial path consists of N serially connected links from a source node to a destination node through $N-1$ intermediate nodes (routers) as illustrated in Figure 4.1-(a). The bandwidth $B_i(t)$ of link i (l_i in the figure), specified in terms of packets-per-second (pps), is assumed to be randomly distributed between B_i^{min} and B_i^{max} with the mean of b_i and the standard deviation of σ_{B_i}, where $1 \leq i \leq N$. $B_i(t)$ remains



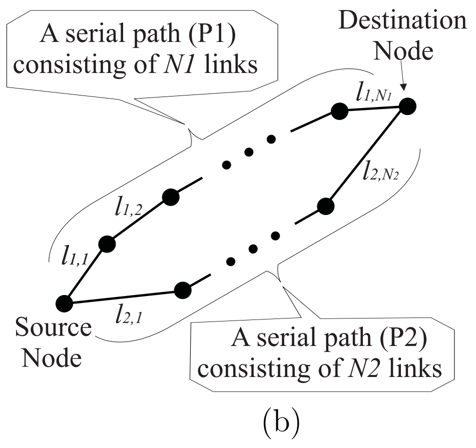

A serial path (P1) consisting of $N1$ links
Destination Node
l_{1,N_1}
l_{2,N_2}
$l_{1,2}$
$l_{1,1}$
Source Node
$l_{2,1}$
A serial path (P2) consisting of $N2$ links
(b)

Figure 4.1: Illustration of (a) a serial path and (b) a parallel path where $N_p = 2$.

constant over a period of time, referred to as an *interval*, $T_{interval}$, as illustrated in Figure 4.2. Also, the normalized standard deviation, $\frac{\sigma_{B_i}}{b_i}$, is denoted by $\bar{\sigma}_{B_i}$.

A parallel path is composed of N_p serial paths which are independent of each other, as illustrated in Figure 4.1-(b). Path j of a parallel path consists of N_j links where $1 \le j \le N_p$. Link i on path j may be denoted by l_{ij} along with the associated bandwidth parameters $B_{ij}(t)$, B_{ij}^{max}, B_{ij}^{min}, b_{ij}, and $\sigma_{B_{ij}}$. For a parallel path where all links on each path are identical, b_j, σ_{B_j}, and $\bar{\sigma}_{B_j}$ will be used to denote the mean, standard deviation, and normalized standard deviation of bandwidth of all links on path j, respectively.

Transmission rate at the source node follows the bandwidth of the bottleneck link (the link with the minimum bandwidth) in the path. Note that a higher transmission rate would lead to a higher throughput, but at the expense of a longer (queueing) delay. The source node is informed of any change in $B_i(t)$ with a delay of $T_{feedback}$, and updates its transmission rate at an interval of T_{update}. Practically, the value of

T_{update} should not be shorter than the average one-way-trip-time from the destination node to the source node.

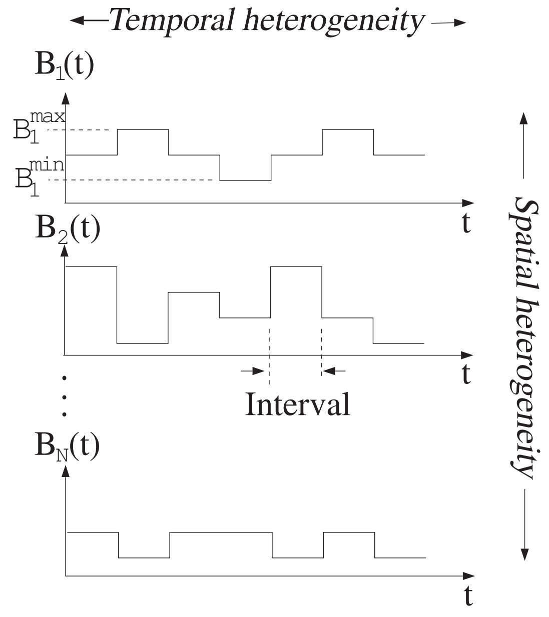


Figure 4.2: Temporal bandwidth heterogeneity on each channel and spatial bandwidth heterogeneity among channels.

Temporal heterogeneity of the bandwidth is defined for each channel or link i and is quantified by $\bar{\sigma}_{B_i}$. This indicates the temporal fluctuation of bandwidth (available for a message) about its mean. Spatial heterogeneity is defined for a group of channels or links and may be quantified by a measure of variation among the members in the set $\{\bar{\sigma}_{B_i}\}$, such as the difference between the maximum and minimum.

There are other factors in addition to channel bandwidth, which can be considered in a network model. However, the main focus of this chapter is on analyzing effects of temporal and spatial variations in the bandwidth allocated to a message on

QoS's. Therefore, those factors are not explicitly considered in the model. Equivalently, the model takes only their combined "net" effect on the allocated bandwidth into account.

4.1.2 Performance Measures

Some communication performance measures used in this dissertation are defined below.

Throughput (THR) is defined to be the size of data that a path (or multi-path) is able to transfer in a unit time, in terms of packets-per-second (pps) when the protocol has a uniform packet size, or bits-per-second (bps). This measure reflects the effective bandwidth of the network path. This is a long-term measure, which is evaluated on the destination node.

End-to-End-Delay (T_{ee}) is defined to be the time it takes to transmit a packet from the source to the destination. This can be decomposed into 4 components, i.e., propagation delay (T_p), transmission delay (T_{tr}), queueing delay (T_q), and other overhead (T_o). T_o represents the overhead including system processing time, packet assembling or disassembling time; T_p is related to the physical property of the network which is determined by the speed of the electro-magnetic wave on a certain media (copper, fiber, etc.); T_q is the waiting time that a packet spends in queueing into and dequeuing from the buffers of routers on the path, and reflects the mismatching of bandwidth between two successive links; and T_{tr} can be expressed in terms of the transmission rate ($r(t)$) as follows:

$$T_{tr}(t) = 1/r(t) \tag{4.1}$$

Note that there is a trade-off between T_{tr} and T_q: when increasing the source rate $r(t)$, T_{tr} decreases since it is inversely proportional to $r(t)$ and T_q tends to increase when $r(t)$ gets faster, if there are some packets buffered. This dissertation mainly focuses on T_{tr} and T_q because they are the two main components of T_{ee} while the other two components are also included in the simulation model.

Dropping Rate (DR) is defined to be the ratio of the number of dropped packets to the total number of packets sent. When the available bandwidth drops below the source rate, packets are queued. Packet dropping occurs when packets are sent with a rate higher than the current available bandwidth of an outgoing link over a long enough period of time so that the buffer overflows. Therefore, a good source rate control scheme should at least follow and not go above the average available link (a long term measure) bandwidth for a long period of time.

Transfer Time (T_{trans}) is the time it takes to send a given size of data from a source to a destination.

4.2 Effects of Heterogeneity

In this section, effects of heterogeneity on QoS's of an individual message are analyzed by computer simulations. The bandwidth of each channel, $B_i(t)$, is assumed to be independently and uniformly distributed between B_i^{min} and B_i^{max}. Other distributions such as a truncated Gaussian distribution have been considered, but lead to qualitatively similar results. Therefore, the results for the uniform distribution only are provided in this dissertation.

4.2.1 Feedback Latency

In Figure 4.3, where the notation $T_{interval} = [X, Y]$ indicates that $T_{interval}$ is uniformly distributed between X and Y, effects of $T_{feedback}$ on throughput and end-to-end delay are analyzed for a serial path of 4 links. It is seen that throughput is almost independent of $T_{feedback}$ while it decreases as $\bar{\sigma}_B$ increases (refer to Figures 4.3-(a)). This is because throughput is mainly determined by the average bandwidth of the path as will be discussed later on. However, $T_{feedback}$ has a significant effect on the end-to-end delay as can be seen in Figure 4.3-(b). The longer the feedback latency is, the longer the end-to-end delay results. When the feedback latency is longer, the duration of mismatch between the source rate and the bandwidth of the bottleneck link is longer leading to a longer end-to-end delay per packet due to longer queues along the path.

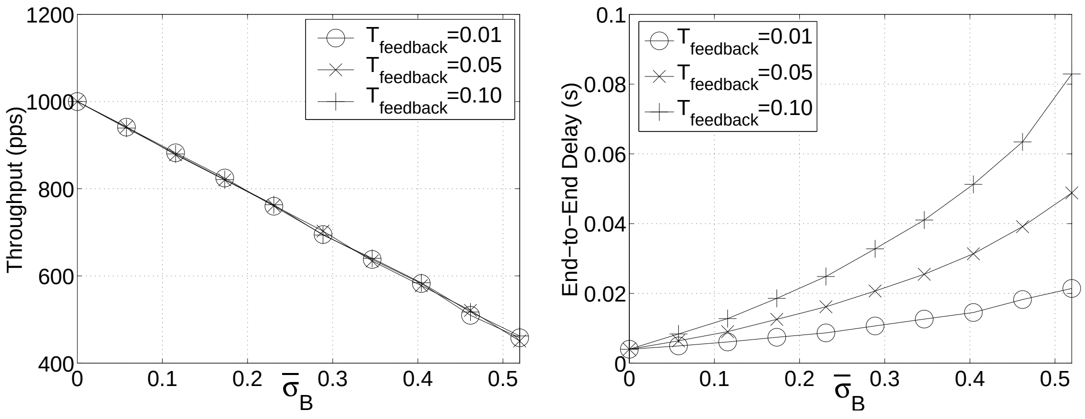


Figure 4.3: Effects of $T_{feedback}$ on (a) throughput and (b) end-to-end delay. $T_{interval}$=[0.01,5.00]s, T_{update}=0.01s, $N = 4$, b_i=1000 pps, and $\bar{\sigma}_{B_i} = \bar{\sigma}_B$ for all i.

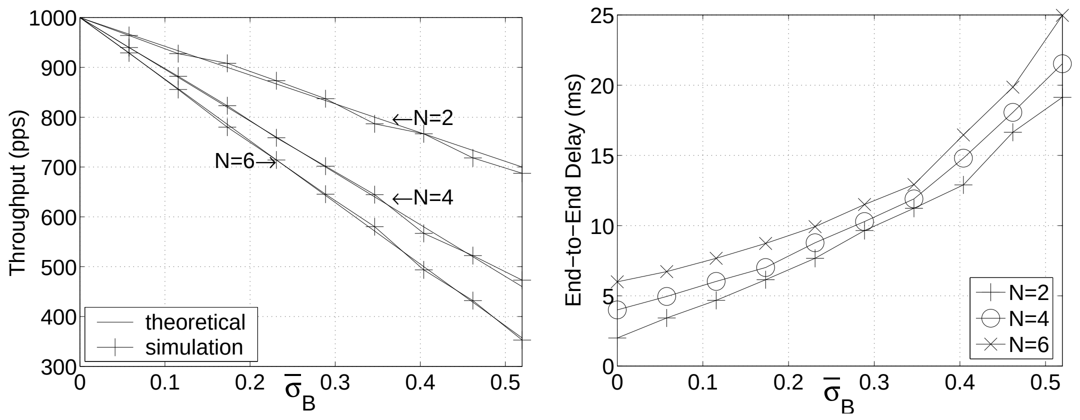
4.2.2 Temporally Heterogeneous and Spatially Homogeneous

4.2.2.1 Serial Path

For a serial path of a single link, it is not difficult to show that the average transfer time, t_{trans}, and the normalized standard deviation of transfer time, $\bar{\sigma}_{t_{trans}}$, can be derived as $t_{trans} = \frac{S}{b}$ and $\bar{\sigma}_{t_{trans}} = \frac{\sigma_{t_{trans}}}{t_{trans}} = \frac{\bar{\sigma}_B \sqrt{b}}{\sqrt{S}}$, respectively, where S is the size of data. Now, consider a serial path of N links where $b_i = b$ and $\sigma_{B_i} = \sigma_B$ for all i. The source rate follows the bandwidth of the bottleneck link along the path in the simulation. Therefore, *effective* bandwidth, $B_{effective}$, of the serial path can be approximated as that of the bottleneck link. The average effective bandwidth, $b_{effective}$, which is equivalent to throughput, can be derived as follows (for the uniform distribution).

$$\begin{aligned} b_{effective} &= E[B_{effective}] \;=\; E[min_i\{B_i\}] \\ &= b - \frac{N-1}{N+1}\left(\frac{B^{max} - B^{min}}{2}\right) \\ &= b\left(1 - \frac{N-1}{N+1}\sqrt{3}\bar{\sigma}_B\right) \qquad (4.2) \end{aligned}$$

From Equation 4.2, one can see that the average effective bandwidth decreases as $\bar{\sigma}_B$ (temporal heterogeneity) increases, and the decrease is larger (as large as $\sqrt{3}\bar{\sigma}_B$) for a longer path (a larger N). The more links are involved in a path, the higher the probability that $min\{B_i\}$ will be smaller. This is verified by the simulation results in Figure 4.4-(a), where it is clear that the simulation results show a good match with the theoretical predictions.



(a) (b)

Figure 4.4: (a) Throughput and (b) end-to-end delay on a temporally heterogeneous serial path of N links. $S = 10^6$ packets. $b_i = 1000$ *pps* and $\bar{\sigma}_{B_i} = \bar{\sigma}_B$ for $i = 1, \ldots, N$.

The end-to-end delay is the sum of delays on N links. The delay on an individual link includes the transmission and queueing delays (refer to Section 4.1.2). The queueing delay increases as $\bar{\sigma}_B$ increases since the probability and degree of bandwidth mismatch between adjacent links increase leading to longer queues. Therefore, the end-to-end delay increases as σ_b and/or N increase as shown in Figure 4.4-(b).

4.2.2.2 Parallel Path

In Figure 4.5, the transfer time and throughput achieved on a parallel path are plotted. In this parallel path, each path is composed of a single link. $S_j = 10^5$ packets for $j = 1, \ldots, N_p$ where S_j is the size of data transferred over path j, i.e., the total size of data is $10^5 N_p$ packets transferred over N_p paths. It can be seen in Figure 4.5-(a) that transfer time increases up to 16% as $\bar{\sigma}_B$ increases when $N_p > 1$. The increase in transfer time is larger for a larger N_p. In Figure 4.5-(b), the aggregated throughput over N_p paths is plotted as a function of N_p. Ideally, it is to increase linearly proportional to N_p. However, throughput is degraded from the ideal one as

N_p increases, with a larger degradation for a larger $\bar{\sigma}_B$ (up to more than 40 % when $N_p = 19$ and $\bar{\sigma}_B$=0.4).

Note that the results in Figure 4.5 are for the cases where each path consists of a single link. By referring to Figure 4.4, it should not be difficult to see that effects of $\bar{\sigma}_B$ and/or N_p are larger on a parallel path when the number of links per path is greater than one as verified in the simulation.

4.2.3 Temporally and Spatially Heterogeneous

In addition to temporal heterogeneity on each link, different links may exhibit different characteristics, i.e., $b_i \neq b_j$ and/or $\bar{\sigma}_{B_i} \neq \bar{\sigma}_{B_j}$ for $i \neq j$. In this section, only the results for cases where $b_i = b$ for all i, but $\bar{\sigma}_{B_i} \neq \bar{\sigma}_{B_j}$ for $i \neq j$, are presented. However, in practice, even when $b_i \neq b_j$, it is often the case that the allocated average bandwidth is the same for all links. For instance, it would not be optimal to allocate different average bandwidths on different links along a serial path since the overall

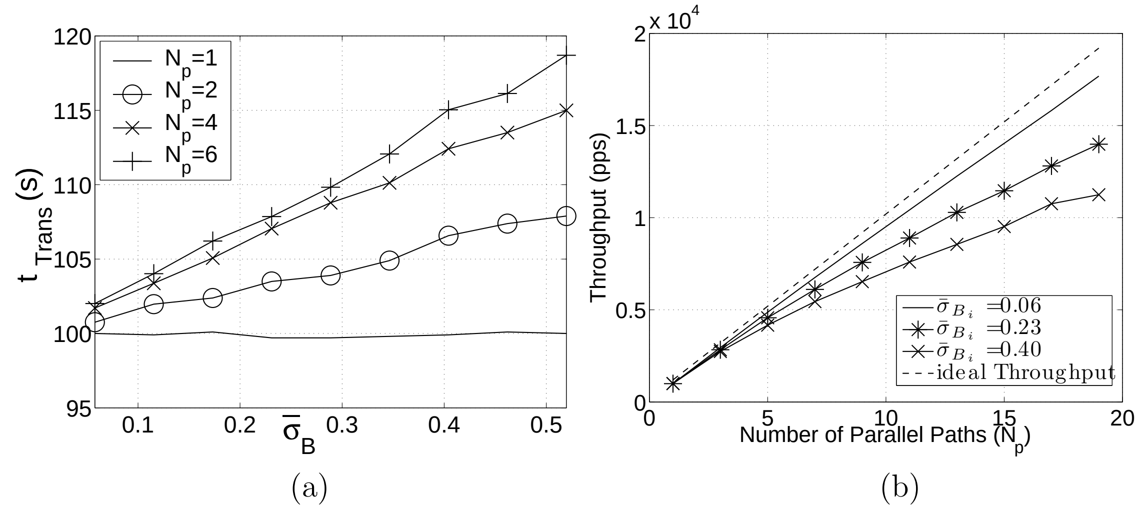


Figure 4.5: (a) Transfer time and (b) throughput on a temporally heterogeneous parallel path consisting of N_p paths with one link per path. S_j=10^5 packets and b_j=1000 *pps* for $j = 1, \ldots, N_p$.

effective bandwidth of the path mainly depends on the bottleneck link, i.e., the link with the lowest (average) bandwidth. Note that a link with a large $\bar{\sigma}_B$ may become a bottleneck link even though its average bandwidth is not the lowest. Nevertheless, cases where $b_i \neq b_j$ are considered for the application examples in Section 4.3.

4.2.3.1 Serial Path

In Figure 4.6, $b_i = 1000$ *pps* for all i and $\bar{\sigma}_{B_1}$ is varied with $\bar{\sigma}_{B_i}$ fixed for $i = 2, \ldots, N$.

In Figures 4.6-(a) and (b), $\bar{\sigma}_{B_i}$ is fixed at 0.1 for $i = 2, \ldots, N$, and $\bar{\sigma}_{B_1}$ is varied from 0 to 0.52. It is clear that throughput is degraded significantly (close to 20%) due to spatial heterogeneity in $\{\bar{\sigma}_{B_i}\}$ as $\bar{\sigma}_{B_1}$ increases. The relative degradation in end-to-end delay is larger as shown in Figure 4.6-(b). As spatial heterogeneity among links increases, the probability and degree of bandwidth mismatch between adjacent links increase, which causes a longer end-to-end delay for each packet as discussed in Section 4.2.2.1. When $\bar{\sigma}_{B_i}$ for $i = 2, \ldots, N$ is fixed at 0.52, the change of spatial heterogeneity in $\{\bar{\sigma}_{B_i}\}$ is smaller as $\bar{\sigma}_{B_1}$ increases from 0 to 0.52. Therefore, the relative degradation in throughput and end-to-end delay, as $\bar{\sigma}_{B_1}$ increases, is less as can be seen in Figures 4.6-(c) and (d).

4.2.3.2 Parallel Path

In Figure 4.7, the average bandwidth is the same for all links, but $\bar{\sigma}_{B_j}$ is distributed linearly among N_p paths such that its mean among the paths is unchanged. In quantifying spatial heterogeneity, $\Delta\bar{\sigma}_B$=$\bar{\sigma}_{B_1}$-$\bar{\sigma}_{B_{N_p}}$ is adopted. Over each of the N_p

paths, the same amount of data ($S_j = 10^5$ packets for all j) is transmitted, and the transfer time is determined by the "slowest" path.

The results show that the transfer time increases substantially as $\Delta\bar{\sigma}_B$ gets larger. The increase in transfer time is larger when each path becomes longer. This is because the temporal (effective) bandwidth heterogeneity of each path increases as the path length increases.

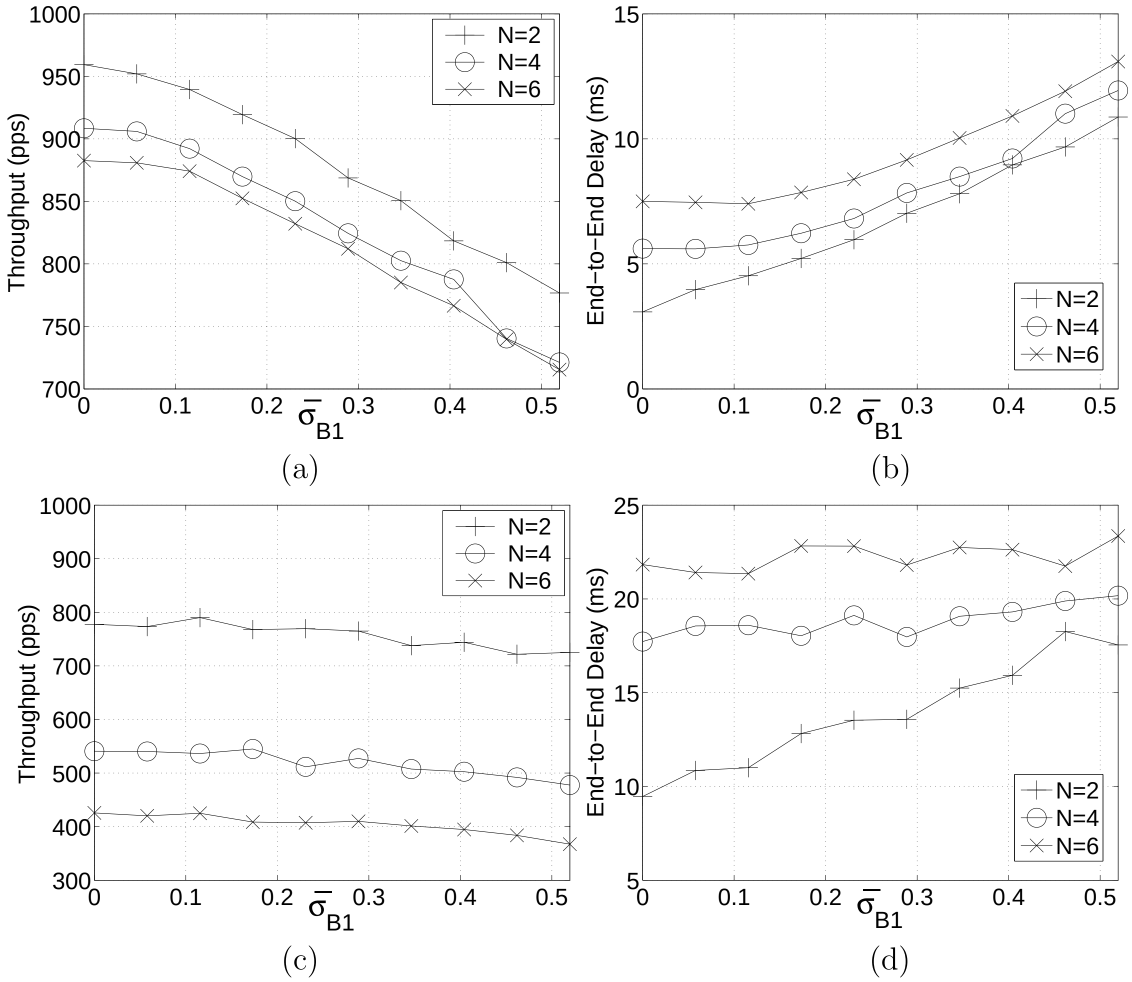
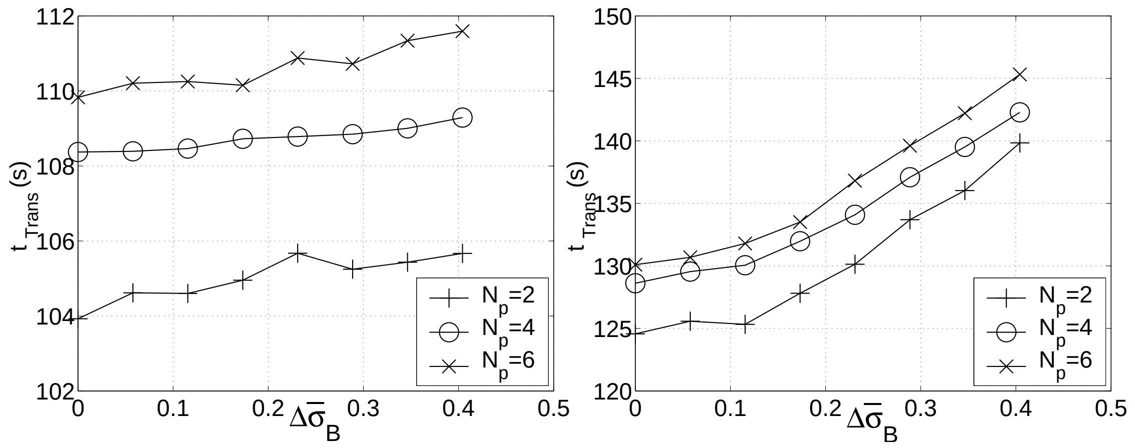


Figure 4.6: Throughput and end-to-end delay on a temporally and spatially heterogeneous serial path with b_i=1000 *pps* for $i = 1, \dots, N$. (a) & (b): $\bar{\sigma}_{B_i} = 0.1$ for $i = 2, \dots, N$; (c) & (d): $\bar{\sigma}_{B_i} = 0.52$ for $i = 2, \dots, N$.



(a) (b)

Figure 4.7: Transfer time on a temporally and spatially heterogeneous parallel path: (a) each path is a single link, and (b) each path consists of 2 links where $\Delta\bar{\sigma}_B = \bar{\sigma}_{B_1}$-$\bar{\sigma}_{B_{N_p}}$. $b_j = 1000$ *pps* and $S_j = 10^5$ packets for $j = 1, \ldots, N_p$. $\bar{\sigma}_{B_j}$ is varied linearly over N_p paths for $j = 1, \ldots, N_p$ such that its mean is fixed at 0.26.

It is also seen that transfer time is longer for a larger N_p. As more paths are involved in a parallel path, the probability that the variation of effective bandwidth among the paths is larger increases, which makes the (average) transfer time longer. Note that transfer time of a message transmitted over multiple paths depends on the "slowest" path, i.e., the path which finishes its transfer last.

4.3 Applications

In this section, two examples where effects of heterogeneity in channel bandwidth may be taken into account in order to improve (optimize) certain QoS's are considered.

4.3.1 Path Selection

In Figure 4.8-(a), a parallel path is shown, where each path consists of 4 links. Note that the average bandwidth of the links on path 3 is higher than those on paths

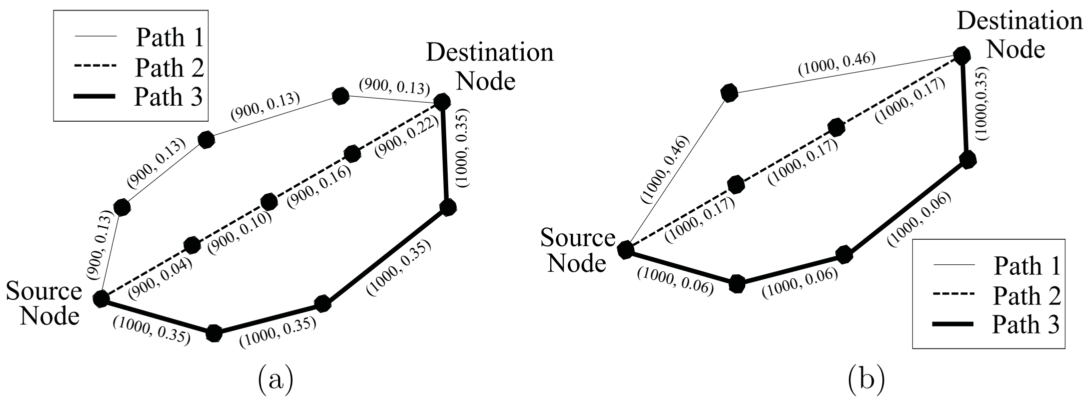
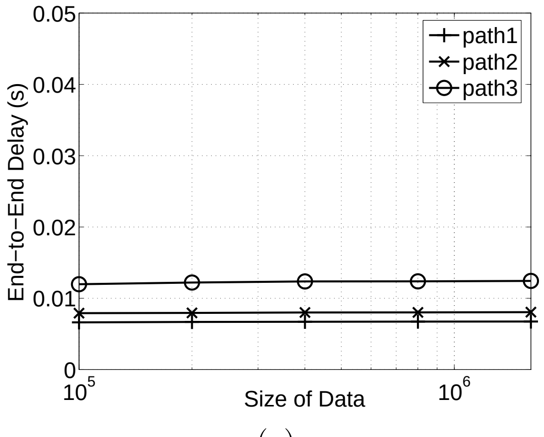


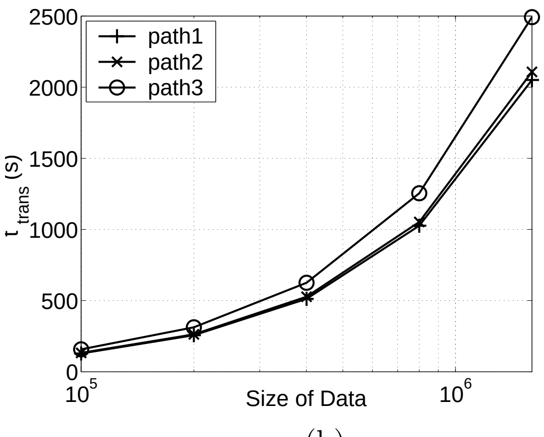
Figure 4.8: (a) A parallel path where the number of links is the same for all paths, and (b) a parallel path where the number of links varies with path. The number pair in brackets over each link represents b_i and $\bar{\sigma}_{B_i}$ of the link.

1 and 2, but $\bar{\sigma}_{B_j}$ (heterogeneity) is also larger for the links on path 3 than those on paths 1 and 2. $\bar{\sigma}_{B_j}$ is constant for all links on path 1 while it is linearly distributed over the links on path 2. Note that the mean of $\bar{\sigma}_{B_i}$ for all links on each path is the same for both paths 1 and 2. In Figure 4.9, the three paths are compared in terms of end-to-end delay and transfer time. First, it is seen that path 3 performs worst among the three in both QoS's though its average bandwidth is higher than that on the other paths. This is due to its high temporal heterogeneity in bandwidth. Comparing paths 1 and 2 which have the identical b_j, one can see that path 1 performs slightly better than path 2 since spatial heterogeneity (among links) is higher on path 2 than on path 1. Therefore, one should consider $\{\bar{\sigma}_{B_i}\}$ in addition to $\{b_i\}$ in selecting a path, in order to optimize the QoS's.

Figure 4.8-(b) shows another example of a parallel path where the number of links varies with path, in this case, 2, 3, and 4 links on paths 1, 2, and 3, respectively, is considered. All links have the same average bandwidth, but $\bar{\sigma}_{B_j}$ is largest on path 1 and smallest on path 3. In Figure 4.10, the same QoS's, end-to-end delay and



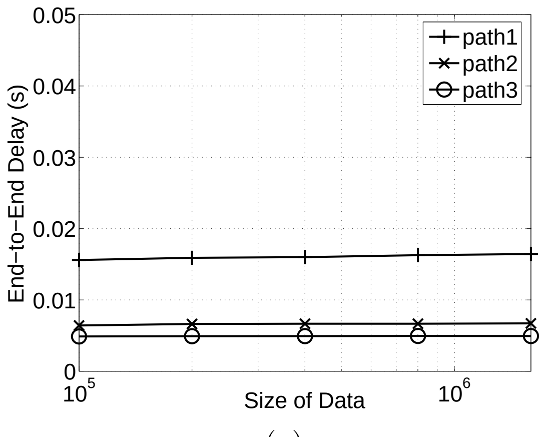
(a)



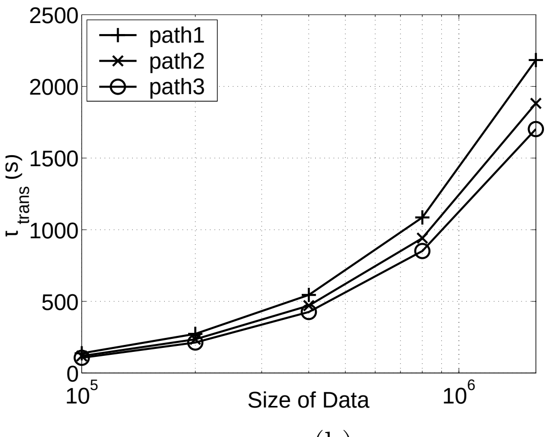
(b)

Figure 4.9: Comparison of the three paths in Figure 4.8-(a) in terms of (a) end-to-end delay and (b) transfer time.

transfer time, are used to compare the three paths. The smallest delay and transfer time are achieved on path 3 which is the "longest" (the largest number of links) among the three paths. Path 1 performs worst due to its highest (temporal) heterogeneity though it is the "shortest." Therefore, one should not simply select the shortest path even when the average bandwidth is the same on all links.



(a)



(b)

Figure 4.10: Comparison of the three paths in Figure 4.8-(b) in terms of (a) end-to-end delay and (b) transfer time.

Implementation

Two cases may be considered: (*i*) the source has the information on the behavior $(b_i, \bar{\sigma}_{B_i})$ of each link, and (*ii*) the source does not have this information. In the case (*i*), an analytic formula or look-up table which relates a given QoS to the set $\{b_i, \bar{\sigma}_{B_i}\}$ characterizing a path is to be derived. If a path is temporally heterogeneous but spatially homogeneous, an analytic formula may be obtained for certain distributions as shown in Section 4.2.2.1 for the QoS of throughput which is proportional to the effective bandwidth of a path. However, for a temporally and spatially heterogeneous path, it is much more challenging. With such an analytic formula or table available, a source refers to it in order to select a path when there are more than one possible path.

In case (*ii*), the source needs to estimate a given QoS by measuring it for each possible path before the selection process.

4.3.2 Multi-Path Data Transfer

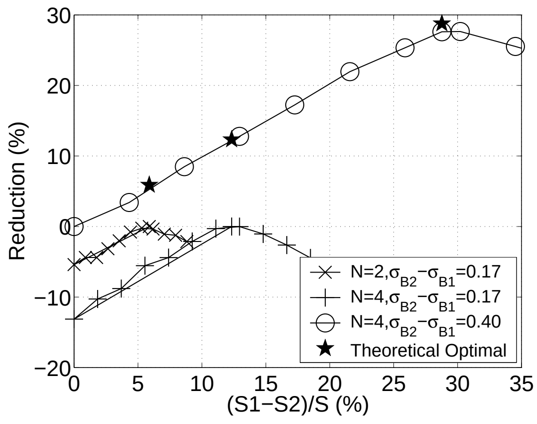
Consider cases where multiple paths (or a parallel path) are available for transferring a large size of data. That is, the data is partitioned for simultaneous transfers over multiple paths. In order to minimize the overall transfer time, it is necessary to determine how much data is to be transferred over each path (or equivalently transmission rate for each path), considering bandwidth heterogeneity on the multiple paths. Let's define percentage *reduction* in transfer time as $\frac{t_{trans} - t'_{trans}}{t_{trans}} \times 100$ where t_{trans} is the transfer time achieved when the partitioning is done considering only the mean of each link's bandwidth, and t'_{trans} is the transfer time obtained by considering the normalized standard deviations (heterogeneity) of link bandwidths also. Similarly, percentage reduction in end-to-end delay may be defined as $\frac{t_{ee}-t'_{ee}}{t_{ee}} \times 100$.

Recall that S denotes the total size of data and S_j the size of data to be transferred over path j, i.e., $S = \sum_j S_j$. In Figure 4.11, cases where 2 paths are employed for data transfer and all links have the same average bandwidth ($b = 1000\ pps$) are considered. Note that one would normally partition S so that $S_1 = S_2$ since $b_1 = b_2$. However, that does not lead to the optimal performance. In Figure 4.11-(a), both paths have the same number of links (N). It can be seen that a significant reduction of up to 28% in transfer time is achieved by assigning more data to the path with lower temporal heterogeneity, path 1 in this example. Also, the reduction becomes larger when there is a larger difference (spatial heterogeneity) between $\bar{\sigma}_{B_1}$ and $\bar{\sigma}_{B_2}$ and when each path is longer (a larger N). In Figure 4.11-(b), cases when each path has different number of links ($N_1 \neq N_2$) are considered. The similar observations can be made in this result. However, the reduction is larger since $N_1 \neq N_2$ leads to higher spatial heterogeneity (between the two paths).

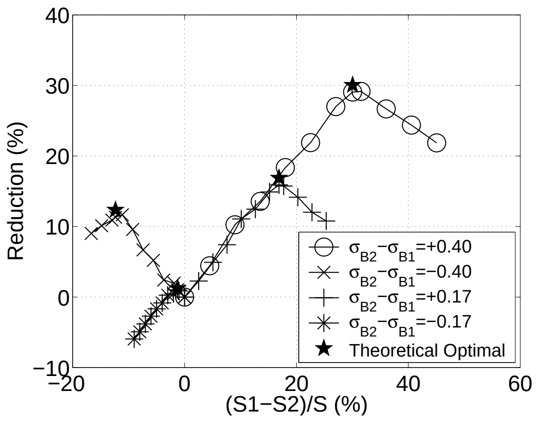
It is also shown in Figure 4.11 that the simulation results provide a good match with the theoretical ones (optimum).

The results for cases where $b_1 \neq b_2$, i.e., two paths have different average link bandwidths, are shown in Figure 4.12. S_j is the size of data to be transferred over path j when only the mean of link bandwidth $\{b_j, j = 1, 2\}$ is taken into account, i.e., S_j is linearly proportional to b_j.

Let S'_j denote the size of data to be transferred over path j when considering $\{\bar{\sigma}_{B_j}, j = 1, 2\}$ in addition to $\{b_j, j = 1, 2\}$. In Figures 4.12-(a) and (b), it can be seen



(a) N_1=N_2 = N



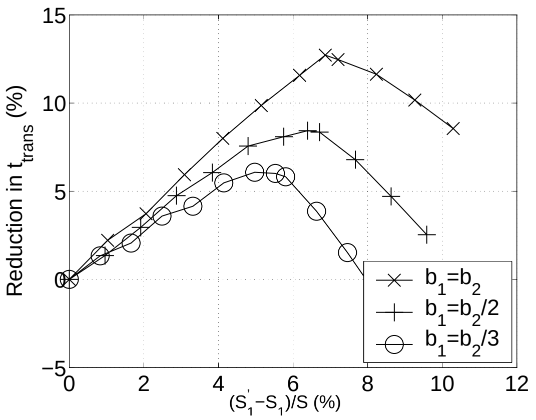
(b) N_1=2, N_2=4

Figure 4.11: Reduction in transfer time on a parallel path ($N_p = 2$). $b_j = 1000$ *pps* for $j = 1, 2$. S=$S_1 + S_2$=2×10^6 packets.

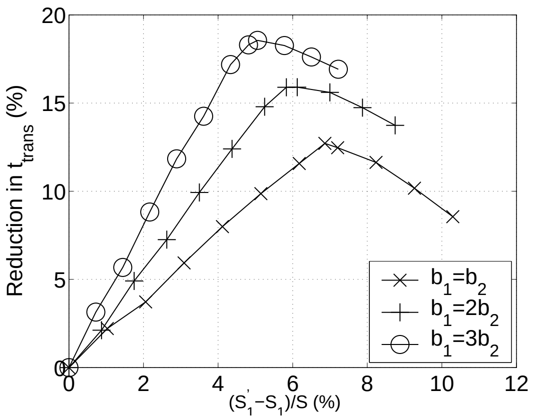
that the minimum transfer time (maximum reduction) is achieved by assigning a size of data larger than S_1 to path 1 on which temporal heterogeneity ($\bar{\sigma}_{B_1}$) is lower than on path 2. The larger the difference in temporal heterogeneity between the two paths (i.e., the larger the spatial heterogeneity), the larger the reduction. Also, a larger reduction is possible when the path with lower temporal heterogeneity (path 1) has a higher average bandwidth. In addition, there exists an optimal point at which the reduction is maximized, and the value of $\frac{S_1'-S_1}{S}$ for the optimal point depends on b_1 and b_2.

In Figures 4.12-(c) and (d), the reduction in the end-to-end delay shows a monotonic behavior, i.e., always increasing as $\frac{S_1'-S_1}{S}$ increases. This is the case since the per-packet delay is reduced as a larger fraction of message is transmitted over the path with lower temporal heterogeneity, path 1 in this case ($\bar{\sigma}_{B_1} < \bar{\sigma}_{B_2}$). Therefore, one is to select a proper value of $\frac{S_1'-S_1}{S}$ depending on which QoS is to be optimized.

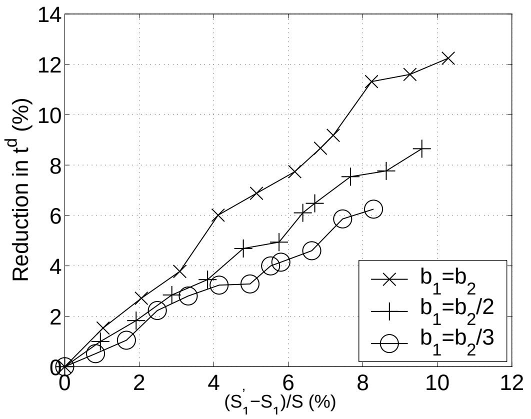
Note that the heterogeneity among paths is likely to increase as N_p increases, and this would lead to an even larger maximum reduction in the transfer time.



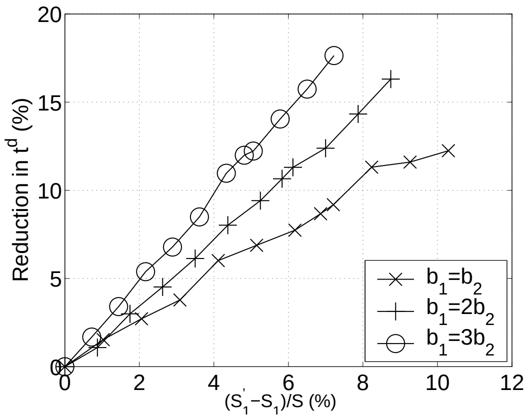
(a) $b_1 \leq b_2$



(b) $b_1 \geq b_2$



(c) $b_1 \leq b_2$



(d) $b_1 \geq b_2$

Figure 4.12: Reduction in transfer time $((a),(b))$ and end-to-end delay $((c),(d))$ on a parallel path with $N_p = 2$. $N_j = 2$ for $j = 1, 2$. $S = S_1 + S_2 = S_1' + S_2' = 2 \times 10^6$ packets, $\bar{\sigma}_{B_1}$=0.06, $\bar{\sigma}_{B_2}$=0.46, and $b_1 + b_2 = 2000$ pps.

Implementation

The main issue in multi-path data transfer is how to divide a given large size of message over multiple paths to be utilized simultaneously for the message, given the effective bandwidth and its variation $(b_j, \bar{\sigma}_{B_j})$ for each path j. Consider the QoS of transfer time of a message. How the message is to be divided to minimize its overall transfer time is equivalent to dividing a computational task over multiple temporally heterogeneous computers for minimizing the parallel execution time. In Chapter 3, an effective load balancing strategy was described along with its performance analysis results. This strategy consists of two steps where a task is partitioned proportional to the average computing power available on the computing nodes in the first step and the partitioning is adjusted by considering the standard deviations of the available computing powers in the second step. A similar scheme can be employed for balancing communication load over temporally heterogeneous paths.

Chapter 5
Source Rate Control for Heterogeneous Communication Systems

Based on the analysis results in Chapter 4, the issue of controlling source rate in order to optimize performance of a communication task is considered, and a heterogeneity aware approach to source rate control is described along with simulation results in this chapter.

5.1 Temporal Heterogeneity under Different Time Scales

Temporal heterogeneity of the available bandwidth on a communication path depends on the activity patterns of different network applications and protocols and can be decomposed into two components, *long-term* and *short-term*, in the time domain. Long-term heterogeneity is caused by the initiation and termination of long sessions, such as large file transfers, real time video and audio streaming, and other bandwidth reservation applications. These types of applications "dominate" the variation of available network resources (bandwidth, buffer, etc), and exhibit slow-varying characteristics.

Short-term heterogeneity is due to short-duration network traffic, such as the overhead in packet processing, buffering, and forwarding, and other sporadic network activities. Examples include Internet Control Messages (ICMP, RSVP), user inputs/outputs of a terminal session (telnet), browsing (small) web pages (http), messenger services (msn/yahoo messenger, icq) most of which are based on UDP, and short email messages (smtp, pop3). Dynamic behaviors of some transport layer protocols, like AIMD in TCP, also cause some short-term variation. This type of heterogeneity is relatively unpredictable.

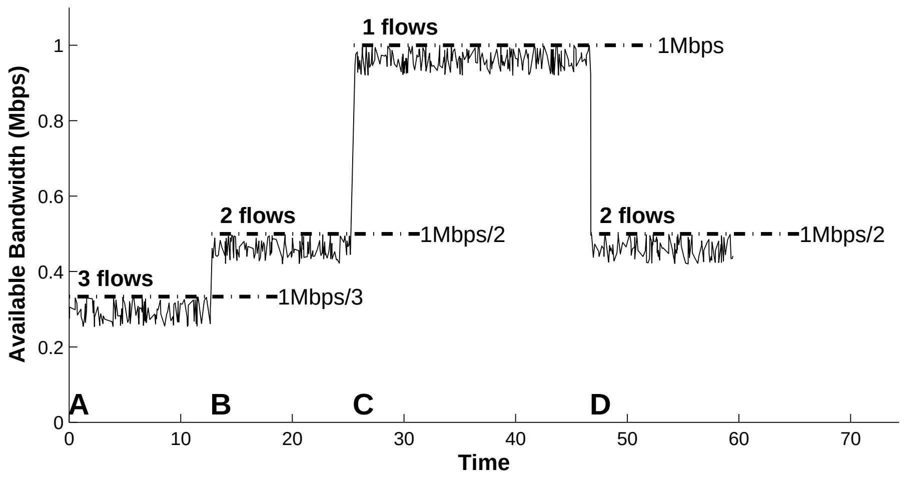


Figure 5.1: Illustration of long-term and short-term temporal heterogeneity of the available bandwidth sensed by one flow ($flow_0$) on a path.

A simplified illustration of long-term and short-term temporal heterogeneity, where multiple flows share a bottleneck link with a capacity of 1 *Mbps*, is given in Figure 5.1. The available bandwidth sensed by a certain long-duration flow (say $flow_0$) over time is plotted in this figure. The queueing scheme is assumed to work in a fairly weighted round-robin fashion, i.e. the ideal effective bandwidth shared by $flow_0$ is $1Mbps/n_f$, when there are n_f co-existing long-duration flows. The "*contour*" of the available bandwidth (marked by the dotted lines) represents the ideal share for $flow_0$, and could be modelled by 1 $Mbps/n_f$. Starting from time A, there are 3 co-existing long-duration flows, so the available bandwidth for $flow_0$ is approximately $1Mbps/3$. At time B, one of the long-duration flows is closed, and the available bandwidth increases to about $1Mbps/2$. When one more flow is dropped at time C, all bandwidth of the link ($1Mbps$) is allocated solely to $flow_0$, but after another flow is introduced at time D, the available bandwidth drops back to about $1Mbps/2$.

Compared to the short term temporal heterogeneity ("spurs" in Figure 5.1), the long-term heterogeneity has a longer time duration and is much easier to track. For example, when the long-term sessions have a duration significantly longer than the synchronization interval of a rate control scheme, the source rate could be adjusted so that it can follow the shape of the available bandwidth contour. However, it is neither possible nor necessary to follow the short term spurs, because they usually vary much faster than the rate control scheme can handle.

5.2 HAA Source Rate Control

5.2.1 Overview of the HAA Source Rate Control Scheme

The main idea of HAA source rate control scheme is to have the source rate follow the long term variation in the available bandwidth, and at the same time to adjust it by means of a "penalty factor" which is derived from the short-term heterogeneity (refer to Section 5.2.5). A finite-size feature extractor is used to collect statistical information on those short term spurs, which is used in the source rate controller.

Figure 5.2 shows the general abstract architecture of the HAA source rate control scheme which consists of four components: *Available Resource Measurement*, *Feature Extractor*, *Source Rate Allocator*, and *Source Rate Controller*.

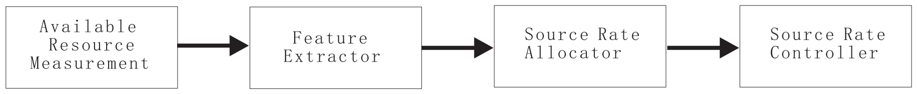


Figure 5.2: The abstract architecture of the HAA source rate control scheme.

The Available Resource Measurement component is responsible for measuring the available resources of interest, which include the *available bottleneck bandwidth* ($ABBW$) of a network path and available buffer size on each router. The proposed scheme only measures $ABBW$. $ABBW(t)$ refers to the minimum bandwidth of all inter-router links at time t, i.e., $ABBW(t) = \min_{i=1}^{N}\{B_i(t)\}$ where i is the link id and $B_i(t)$ is the available bandwidth of link i. Later in this dissertation, the notation $ABBW_l$ will be used to represent $ABBW$ measurement samples, where l is the sample index.

The Feature Extractor extracts quantified heterogeneity information (features) from the measurement samples of the available resources. Typical features are the first order moment (mean), the second order moment (deviation), minimum, maximum, and median of measurement samples for a certain time duration. Currently, only the mean, deviation, minimum, and maximum are used in the proposed scheme.

The Source Rate Allocator is responsible for computing the appropriate source rate for the corresponding communication session based on the extracted features, and the Source Rate Controller is the one which actually controls the source rate.

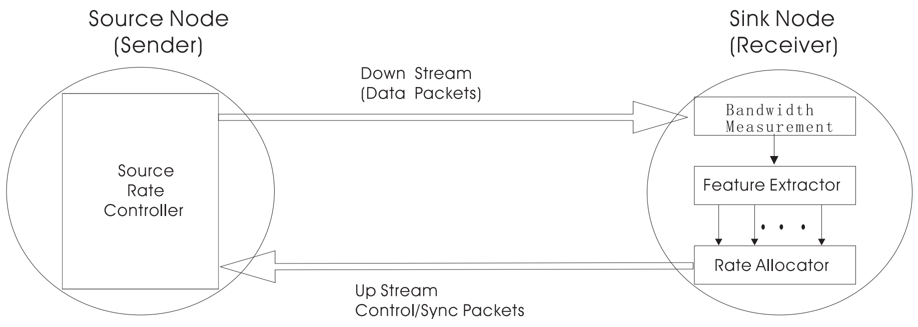


Figure 5.3: Illustration of the proposed HAA communication model.

The schematic diagram of the proposed HAA source rate control communication model is shown in Figure 5.3. The receiver is responsible for measuring the bottleneck bandwidth, and extracting the features from the bandwidth measurement samples. It then periodically (by default one round-trip-time) sends control packets, which contain the information on the allocated rate, to the sender. Upon receiving each control packet, the sender tries to adapt the allocated source rate.

5.2.2 Bottleneck Bandwidth Measurement

The bottleneck bandwidth is measured with a packet pair [28][29][30] based approach in the current implementation. The source sends out data in the form of back-to-back packets, and measurement of $ABBW(t)$ is performed on the receiver side. An up-link (from receiver to sender) carries control packets and a down-link (from sender to receiver) transmits data. The latter has a much heavier load and experiences higher heterogeneity than the former. As a result, the receiver side measurement can get more accurate and timely information on the network load.

Figure 5.4 shows the transmission of packets along a network path. In order to avoid errors due to lost or out of order packets, $ABBW$ is measured only for "valid packet pairs" received. A valid packet pair consists of two packets received successively where the first packet with an even sequence number, $2l$, is followed by the other with the sequence number of $2l+1$.

$ABBW$ measured by the lth valid packet pair ($ABBW_l$) is

$$ABBW_l = \frac{1}{T_{2l+1} - T_{2l}} \tag{5.1}$$

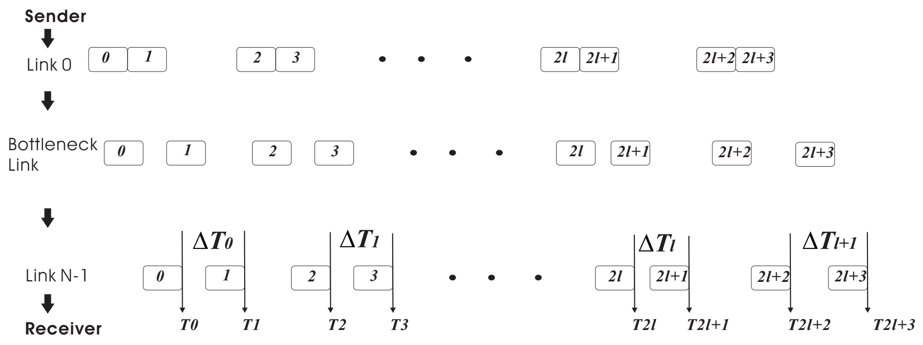


Figure 5.4: Illustration of packet-pair based bandwidth measurement. The horizontal dimension is time, and from top to bottom along the vertical dimension are the sender, intermediate links, and the receiver. Numbers in small boxes are the packet sequence numbers.

where T_{2l} and T_{2l+1} are the receiving times of two packets received consecutively. The time difference $\Delta T_l = T_{2l+1} - T_{2l}$ is dependent on the available bandwidth of the bottleneck link.

5.2.3 Feature Extractor

The features that may be used in the proposed scheme include the mean ($abbw$), standard deviation (σ_{ABBW}), minimum ($ABBW^{min}$), and maximum ($ABBW^{max}$) of available bottleneck bandwidth measurement samples ($ABBW_l$) within an interval, as shown in Figure 5.5. All features are calculated over a sliding window of size L (in terms of the number of packet pairs), which has a longer time duration than the sampling interval. A *feature extractor* is designed to extract the features for the measurement samples of the last L packet-pairs (from $ABBW_{l-L+1}$ to $ABBW_l$), where L is the size of the feature extractor and l is the packet-pair index (or sample index). The *feature interval* is defined to be the time interval over which the L packet pairs are spread.

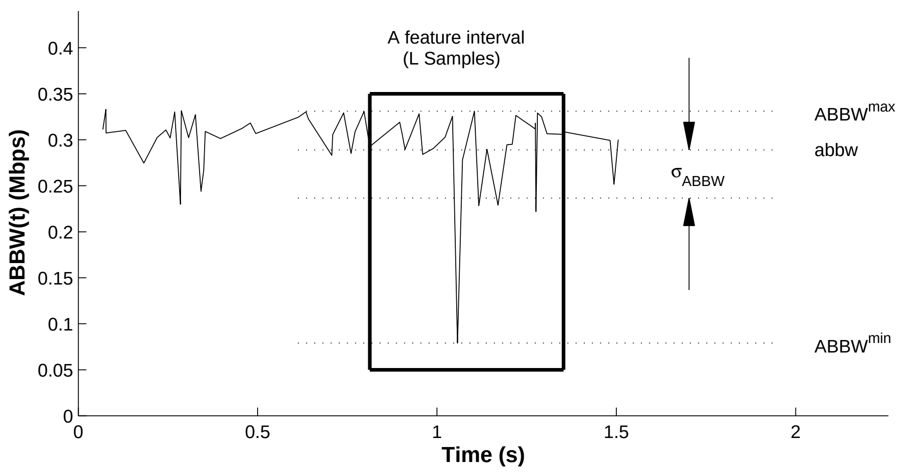


Figure 5.5: Feature extractor.

In order to reduce the computing and storage complexity, two accumulators (A_1, A_2) are employed for the sums of $ABBW_l$ and $ABBW_l^2$, and two other memory variables ($ABBW^{max}$ and $ABBW^{min}$) are used to record the maximum and minimum values for the last L samples. The initialization code is executed immediately after the first valid packet pair is received and the corresponding $ABBW_0$ is obtained (reset A_1 and A_2 to $L \times ABBW_0$ and $L \times ABBW_0^2$, respectively); the iteration code is to calculate the sum of the last L $ABBW_l$ and $ABBW_l^2$ after each valid packet pair l is received; and the extraction code is performed at every *control interval*, which will be described in Section 5.2.5. These codes are shown in Figure 5.6.

5.2.4 Simple Source Rate Allocation Approaches

Three simple source rate control schemes are described for comparison with the proposed scheme (HAA):

Mean-Only Scheme: The source rate may be controlled to follow *abbw*, the mean of measured $ABBW$. This scheme is easy to implement and the throughput is

```
/*initialization code*/
$A_1 = L \times ABBW_0, A_2 = L \times ABBW_0^2$
/*iteration code*/
$A_1 = A_1 + ABBW_l - ABBW_{l-L}$
$A_2 = A_2 + ABBW_l^2 - ABBW_{l-L}^2$
/*extraction code*/
$abbw = A_1/L$
$\sigma_{ABBW} = \sqrt{\frac{1}{L-1}A_2 - \frac{L}{L-1}abbw^2}$
$ABBW^{max} = max(ABBW_j,\ l-L+1 \leq j \leq l)$
$ABBW^{min} = min(ABBW_j,\ l-L+1 \leq j \leq l)$
```

Figure 5.6: Initialization, iteration, and extraction codes for the feature extractor.

close to the bottleneck link's capacity. However, it causes a high dropping rate and a long end-to-end delay due to the limited packet buffer size.

Scaling Scheme: A *scaling factor*, α ($0 < \alpha < 1$), may be employed to make the system work under a lighter load, thus lowering the dropping rate and end-to-end delay [29][30]. The main problem with this scheme is that it sacrifices throughput unnecessarily when the variation of $ABBW$ is small. For example, when α is chosen to be 0.9, there will always be 10 percent of bandwidth being wasted, even if the bandwidth variation is very small (or no variation). The situation is even worse when the bandwidth variation is large and α is not small enough.

Minimum-Bandwidth Scheme: The source rate may be set to the $ABBW^{min}$ at each synchronization interval. However, this is an overly conservative scheme, especially when the $ABBW$ variation is large. For example, when there is a deep valley ($ABBW^{min}$), in a feature interval as shown in Figure 5.5, there would be a lot of bandwidth wasted, since in most of time the available bandwidth is above this value.

5.2.5 Heterogeneity Aware Approach

Synchronization

In order to alleviate the network load, control packets are sent to the sender at every *synchronization interval*. The synchronization interval determines how frequently control packets are sent to the sender, therefore, the shorter the synchronization interval is, the faster the sender can adapt the source rate to the variation of available bandwidth, although more of the up-link bandwidth will be consumed. Usually, it should not be shorter than one round-trip time.

The synchronization interval in the proposed scheme is a function of time, $T_{sync}(t)$, which is set to be smaller when a larger long-term variation is detected, so that the source rate can adapt to the actual $ABBW$ faster. The effect of T_{sync} is discussed in the next section.

HAA Source Rate Function

The main idea of the proposed scheme is to derive the desired source rate r_s at every synchronization interval, based on the set, EF_k, of extracted features including $abbw_k$, $\bar{\sigma}_{ABBW_k}$, $ABBW_k^{min}$, $ABBW_k^{max}$. where k is the synchronization interval index. A general form of HAA *source rate function* $f(ABBW_k,\ EF_k,\ t)$ is shown in Equation 5.2.

$$r_s(k) \;=\; f(ABBW_k,\ EF_k,\ t) \tag{5.2}$$

An implementation of the source rate function is shown in Equation 5.3, which can detect and utilize both long-term and short-term heterogeneity.

$$r_s(k) = \begin{cases} max[ABBW_k^{min}, abbw_k(1 - \beta\bar{\sigma}_{ABBW_k})] & \text{for } \bar{\sigma}_{ABBW_k} \leq \bar{\sigma}_{threshold}; \\ ABBW_k & \text{for } \bar{\sigma}_{ABBW_k} > \bar{\sigma}_{threshold}; \end{cases} \tag{5.3}$$

where $\bar{\sigma}_{threshold}$ is selected such that $\bar{\sigma}_{ABBW_k}$ exceeds it when there is a clear large scale edge (rising or falling) in $ABBW$. A significant transition is most likely due to the release or injection of a long-term competing flow, and the magnitude of the corresponding rise or fall in $ABBW$ is assumed to be significantly larger than the variation due to the short-term heterogeneity.

When $\bar{\sigma}_{ABBW_k}$ is less than $\bar{\sigma}_{threshold}$, $r_s(k)$ is penalized by $\beta \times \bar{\sigma}_{ABBW_k}$, but is set no lower than $ABBW_k^{min}$. β is a *penalty factor* which controls how much the source rate should be penalized for the heterogeneity. The rationale is that the source rate gets a less penalty when $\bar{\sigma}_{ABBW_k}$ (short-term heterogeneity) is small, and a larger penalty when $\bar{\sigma}_{ABBW_k}$ is large.

When $\bar{\sigma}_{ABBW_k}$ becomes greater than $\bar{\sigma}_{threshold}$, i.e., a clear rising or falling edge, $r_s(k)$ is set to the most recent $ABBW_k$ in order to obtain a more accurate estimation of the long-term variation in the current implementation. Also, in order to get a faster response, T_{sync} is made much smaller (thus faster) than the other operation mode.

Due to the delay from the $ABBW$ to $abbw$, the source rate is over-estimated on the falling transitions and under-estimated on the rising transitions. Some more sophisticated predictive scheme could be developed to further improve the performance.

Adaptive Penalty (β)

The main issue in the proposed HAA approach to source rate control is how to select a proper value of the penalty factor (β). In many cases, the users specify QoS requirements that their applications should meet. A typical specification of QoS requirements is given below:

$$\text{QoS requirements} \begin{cases} \text{Throughput :} & thr \geq THR_{min} \\ \text{End-to-End Delay :} & t_{ee} \leq Tee_{max} \\ \text{Dropping Rate :} & dr \leq DR_{max} \end{cases} \tag{5.4}$$

The means (averages) of throughput (*THR*), end-to-end delay (*Tee*), and dropping rate (*DR*) are denoted by thr, t_{ee}, and dr, respectively. THR_{min} is the minimum throughput requirement, and Tee_{max}, and DR_{max} are the upper bounds of the end-to-end delay, and dropping rate, respectively. These are all user-specified parameters, and are given during the session admission control stage.

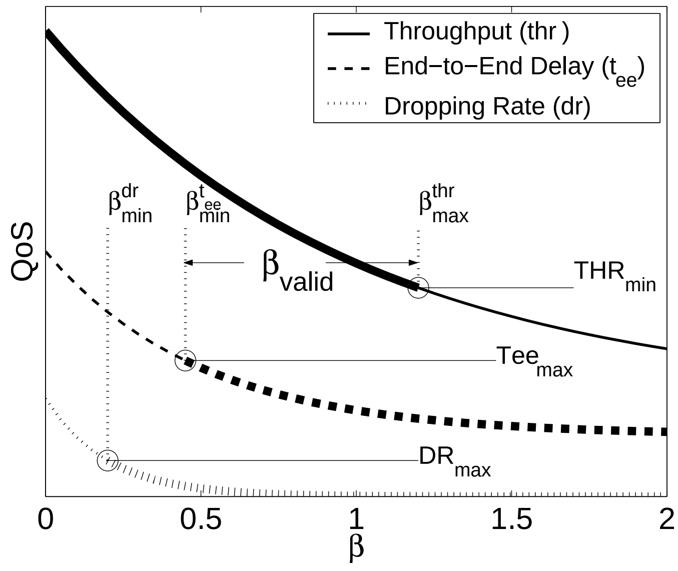


Figure 5.7: QoS requirements and valid range of β.

The proposed scheme determines the valid range of β so that all the QoS requirements can be met by adaptively adjusting β. This is based on the fact that each QoS measure mentioned above is a monotonic function of β, and hence the upper/lower bounds of β for each QoS measure QoS_i can be found via a simple heuristic search.

$$\begin{cases} \text{Throughput :} & \beta^{thr} \leq \beta^{thr}_{max} \\ \text{End-to-End Delay :} & \beta^{t_{ee}} \geq \beta^{t_{ee}}_{min} \\ \text{Dropping Rate :} & \beta^{dr} \geq \beta^{dr}_{min} \end{cases} \tag{5.5}$$

where β^{thr}_{max}, β^{thr}_{min}, and β^{dr}_{min} are the upper/lower bounds of β that satisfy the requirement of throughput, end-to-end delay, and dropping rate, respectively, as shown in Figure 5.7.

And the valid range of β which satisfies all the QoS requirements is

$$max(\beta^{t_{ee}}_{min}, \beta^{dr}_{min}) \leq \beta_{valid} \leq \beta^{thr}_{max} \tag{5.6}$$

If $max(\beta^{t_{ee}}_{min}, \beta^{dr}_{min}) > \beta^{thr}_{max}$, no β can be found to meet all the QoS requirements, and the admission control thus rejects the flow.

A "Slowly-Decreasing-Fast-Increasing" (SDFI) scheme is proposed to adaptively determine a near optimal β. It goes through a 3-stage cycle when a QoS requirement violation is detected. Let $\Delta\beta_{slow}$ denote the "slowly decreasing step" (0.1 by default), $\Delta\beta_{fast}$ the "fast increasing step" (0.6 by default) of β, and β_0 the initial value in the cycle. To reduce the oscillation in the adaptation, a margin is introduced for each

QoS measure (QoS_{margin}), and then the adjusted QoS measures are

$$\begin{cases} THR'_{min} &= THR_{min} + QoS^{thr}_{margin} \\ Tee'_{max} &= Tee_{max} - QoS^{tee}_{margin} \\ DR'_{max} &= DR_{max} - QoS^{dr}_{margin} \end{cases} \tag{5.7}$$

The margin for each QoS measure depends on the tolerance requirements of applications. It is either provided by the user as a part of QoS requirements, or set by the SDFI scheme via experiments. The adjusted QoS measures are "tighter" than those given in the QoS specification, and are used during the Slowly Decreasing stage. The complete pseudo-code of the SDFI scheme is shown in Figure 5.8.

```
while (true)
{
    //Fixed stage
    while ($t_{ee} < Tee_{max}$ && $dr < DR_{max}$ && $thr > THR_{min}$);
    //Fast Increase stage
    $\beta = \beta + \Delta\beta_{fast}$;
    //Slowly Decrease stage
    while ($t_{ee} < Tee'_{max}$ && $dr < DR'_{max}$ && $\beta \geq \Delta\beta_{slow}$)
    {
        $\beta = \beta - \Delta\beta_{slow}$;
    }
    $\beta = \beta + \Delta\beta_{slow}$;
    if($thr < THR'_{min}$) exit(); //can not meet all requirements
}
```

Figure 5.8: Pseudo-code of the proposed SDFI scheme.

(1) Fixed stage: the receiver keeps checking all the QoS measures, and β remains unchanged until any of them violates the requirement.

(2) Fast Increasing stage: β is incremented by $\Delta\beta_{fast}$.

(3) Slowly Decreasing stage: β is decremented by $\Delta\beta_{slow}$ in each step until the QoS measure exceeds QoS'_{max}. β is incremented by $\Delta\beta_{slow}$, i.e., it returns to the last valid value.

The operation of the SDFI scheme is illustrated in the time domain in Figure 5.9.

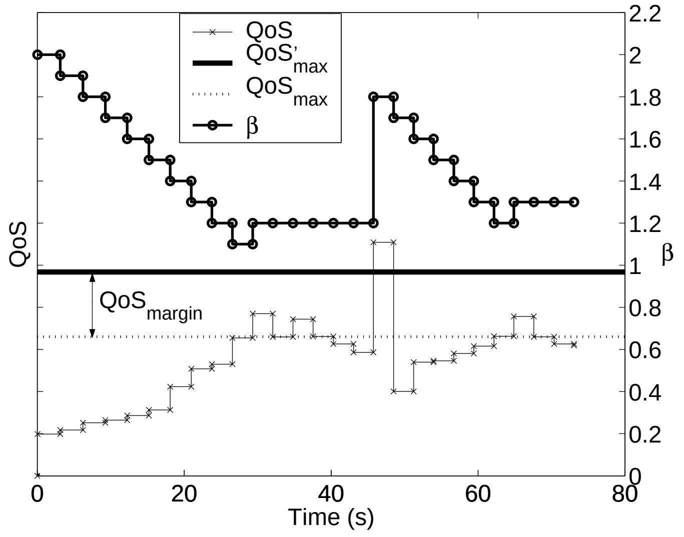


Figure 5.9: Operation of the SDFI scheme.

Chapter 6

Performance of the Source Rate Control Scheme

6.1 Simulation Setup

An extensive simulation has been carried out using a widely-used network simulator (*ns2* [44]). The performance of a single HAA flow is first studied. The packet size in all the results in this chapter is 1000 bytes. The size of buffer on each router is set to be 20 packets by default, and the queuing scheme is the drop-tail round-robin. The default length (L) of the feature extractor is 15 samples. The default synchronization interval is set to be one round-trip-time, except when a significant transition in $ABBW$ is detected.

Topology

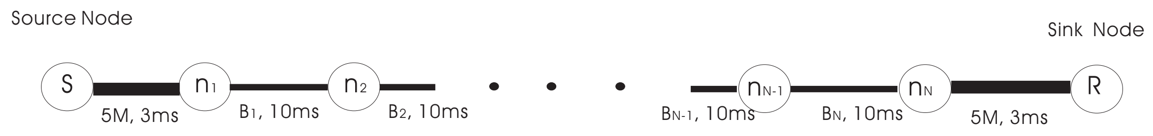


Figure 6.1: Simulation testbench topology for a single path.

The topology of a single path employed in the simulation is shown in Figure 6.1. The link capacity and propagation delay for local connection links (thick lines) are $5Mbps$ and $3ms$, respectively. For each inter-router link i, the propagation delay is $10ms$ and the available bandwidth $B_i(t)$ is a random variable whose the upper limit is $5 Mbps$.

Queue Management

A round-robin like queue management is assumed for the network routers in the simulation since it offers several advantages over the first-in-first-out (FIFO) packet schedulers for bursty data traffic. In particular, the round-robin queue management automatically enforces a min-max fair allocation of resources [45], and ensures that the "well-behaving" users are protected from the "ill-behaving" ones, which is desirable in public data networks [30][46].

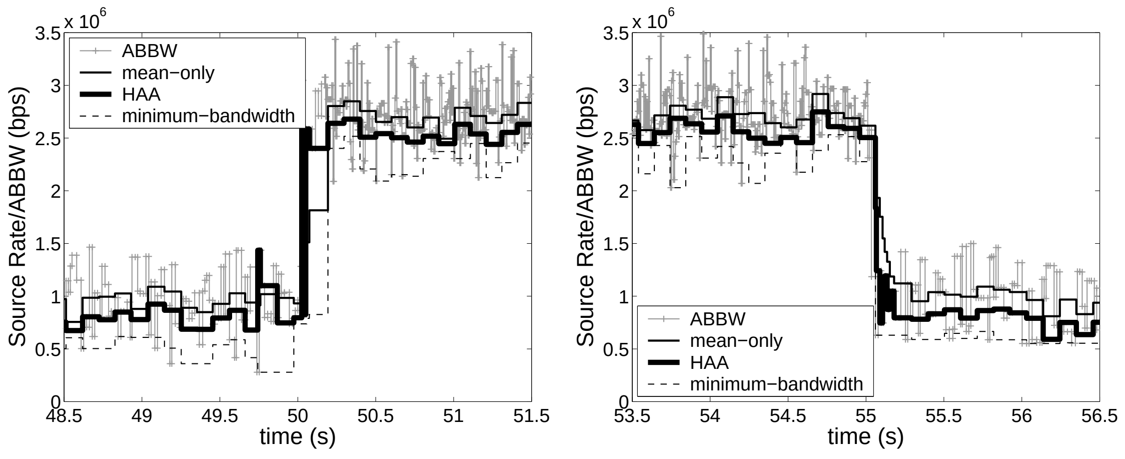
Traffic Generation

Instead of directly simulating the background traffic, the available bandwidth on each link is changed dynamically in the simulation script. This approach enables easy control of the link bandwidth to simulate the heterogeneity under different scenarios, and also saves simulation time.

The varying bandwidth is simulated in two dimensions: magnitude and time. The magnitude variation on a link is independent of those on others and follows a certain distribution. Several different distributions have been tried, including the uniform distribution and truncated normal distribution, which all lead to similar results. Therefore, all simulation results provided in this section assume a uniform distribution. The magnitude of bandwidth on link i follows a uniform distribution with the mean of b_i and normalized standard deviation of $\bar{\sigma}_{B_i}$. In the time dimension, each random magnitude of bandwidth remains unchanged over a random *interval* ($T_{interval}$) which follows an exponential distribution with a mean of $t_{interval}$.

6.2 Single Flow

6.2.1 Adaptation to Long-Term Bandwidth Variation

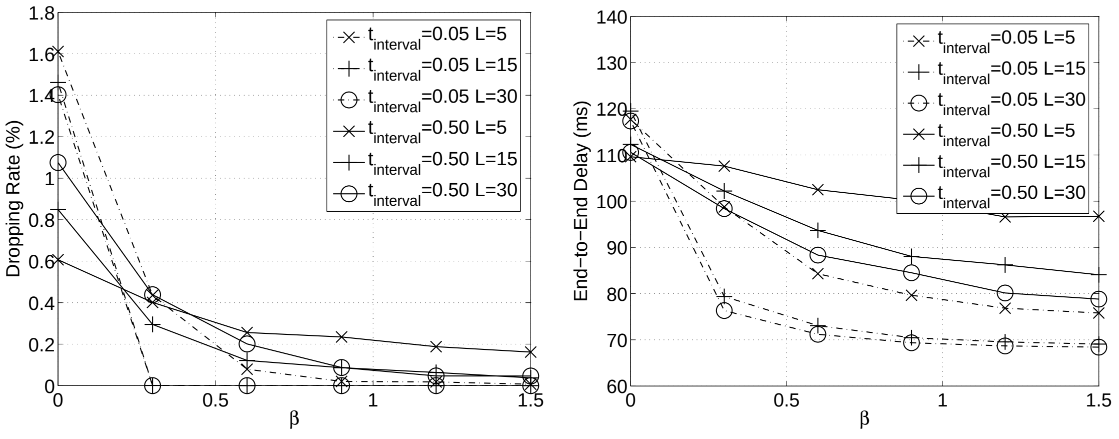


(a) Rising Edge (b) Falling Edge

Figure 6.2: Comparison of three source rate control schemes on a path of an inter-router link whose available bandwidth ($ABBW$) varies with time. The penalty factor (β) is set to be 0.7 for HAA.

Figure 6.2 shows how the three different source rate control schemes adapt the source rate to the available bottleneck bandwidth ($ABBW$). $ABBW$ is generated by a "slow" square wave (with a period of 5 seconds and a magnitude of 1 to $3Mbps$) plus a high-frequency variation. The average interval of the high-frequency component is 50ms and the magnitude is no greater than 500Kbps. It is shown that all three schemes can follow $ABBW$ on a large time scale, but the minimum-bandwidth scheme is too conservative to utilize the available bandwidth efficiently, and the mean-only scheme is too greedy, as the allocated bandwidth is higher than $ABBW$ for about half of the period, and thus leads to a large dropping rate. The proposed scheme (HAA) tends to adopt a rate between those by the mean-only and minimum-bandwidth schemes in an effort to optimize the performance such as throughput and dropping rate.

6.2.2 Feature Extractor Size and Average Bandwidth Updating Interval



(a) Dropping Rate (b) End-to-End Delay

Figure 6.3: Effects of feature extractor size (L) and $t_{interval}$. N=4, $\bar{\sigma}_{B_i}$=0.17 for all 4 links.

In Figure 6.3, effects of the feature extractor size (L) and average bandwidth updating period ($t_{interval}$) on the dropping rate and end-to-end delay are analyzed. Comparing $\beta = 0$ and $\beta > 0$ (HAA), it is seen that the improvement by HAA is more significant when the bandwidth varies more rapidly ($t_{interval}$=0.5, dashed lines in the figure). Also, the results indicate that the feature extractor size L should not be too small. The performance (in terms of the dropping rate and end-to-end delay) for L=15 or 30 is better than that for L=5, while the improvement by L=30 over L=15 is not very large, therefore, L=15 is used in most cases of the simulation.

6.2.3 HAA Compared with Packet Spacing Protocol (PSP)

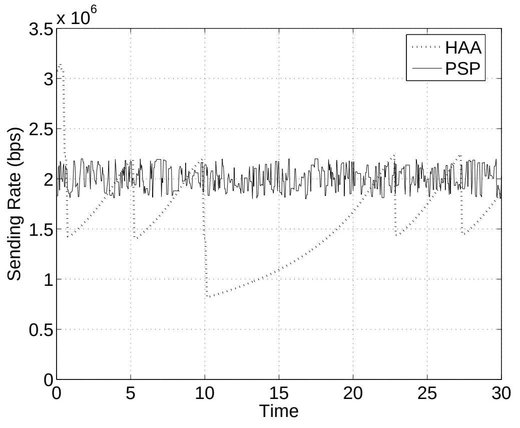
HAA essentially uses a packet spacing approach to adjust the source rate at the sender. It is compared with the Packet Spacing Protocol (PSP) [41][42], which is another packet spacing based rate control scheme. The comparison results are provided in Figure 6.4.

The penalty factor β is set to 0 in this simulation, i.e., non-adaptive and with no heterogeneity penalty. It can be seen that the source rate variation of the proposed scheme is much smaller than that of PSP (Figure 6.4-(a)). The effective throughput (Figure 6.4-(b)) of HAA is about 25% better than that of PSP, and the improvement becomes even larger when the heterogeneity ($\bar{\sigma}_B$) increases. The dropping rate is also decreased significantly, especially when $\bar{\sigma}_B$ is small, from 0.004 to almost 0. The only measure that is worse for the proposed scheme than for PSP is the end-to-end delay. However, it should be noted that PSP drops more packets which are not counted toward the end-to-end delay, and HAA achieves a significantly higher throughput.

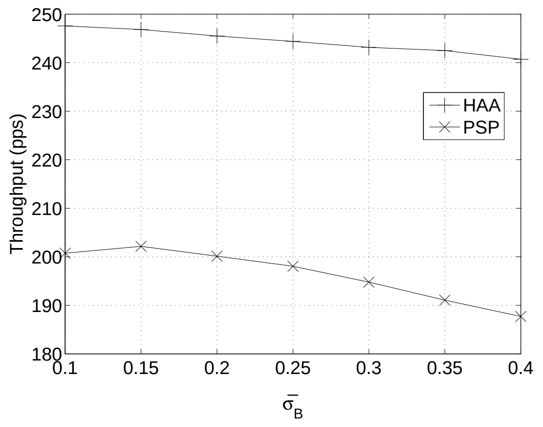
6.2.4 HAA versus Mean-Only and Minimum-Bandwidth Schemes

In Figure 6.5, the effect of β on the performance of HAA is analyzed and HAA is also compared with two other fixed rate control schemes (mean-only and minimum-bandwidth) under two different bandwidth varying scenarios: $t_{interval} = 0.05$s (fast) and $t_{interval} = 0.5$s (slow).

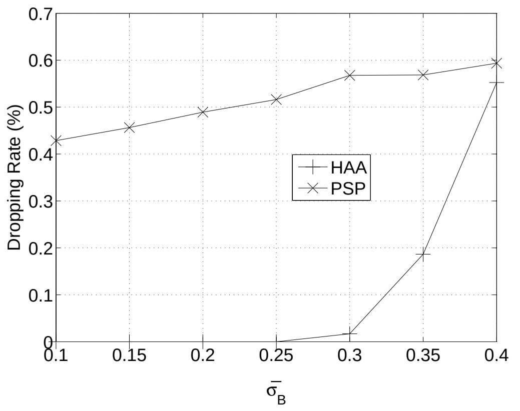
In each scenario, three sets of results are obtained for three different levels of heterogeneity ($\bar{\sigma}_{B_i} = 0.06$, 0.17, and 0.29, respectively) with β varied. The results for



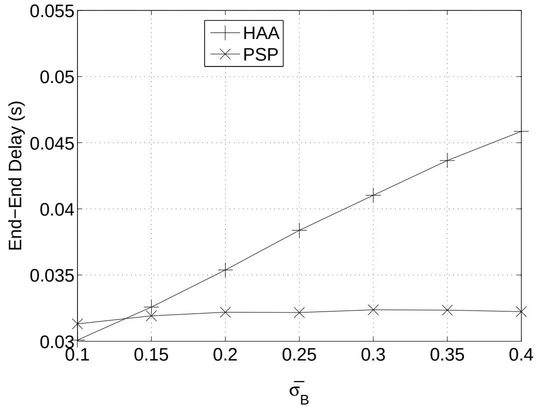
(a) Source Rate



(b) Effective Throughput



(c) Dropping Rate

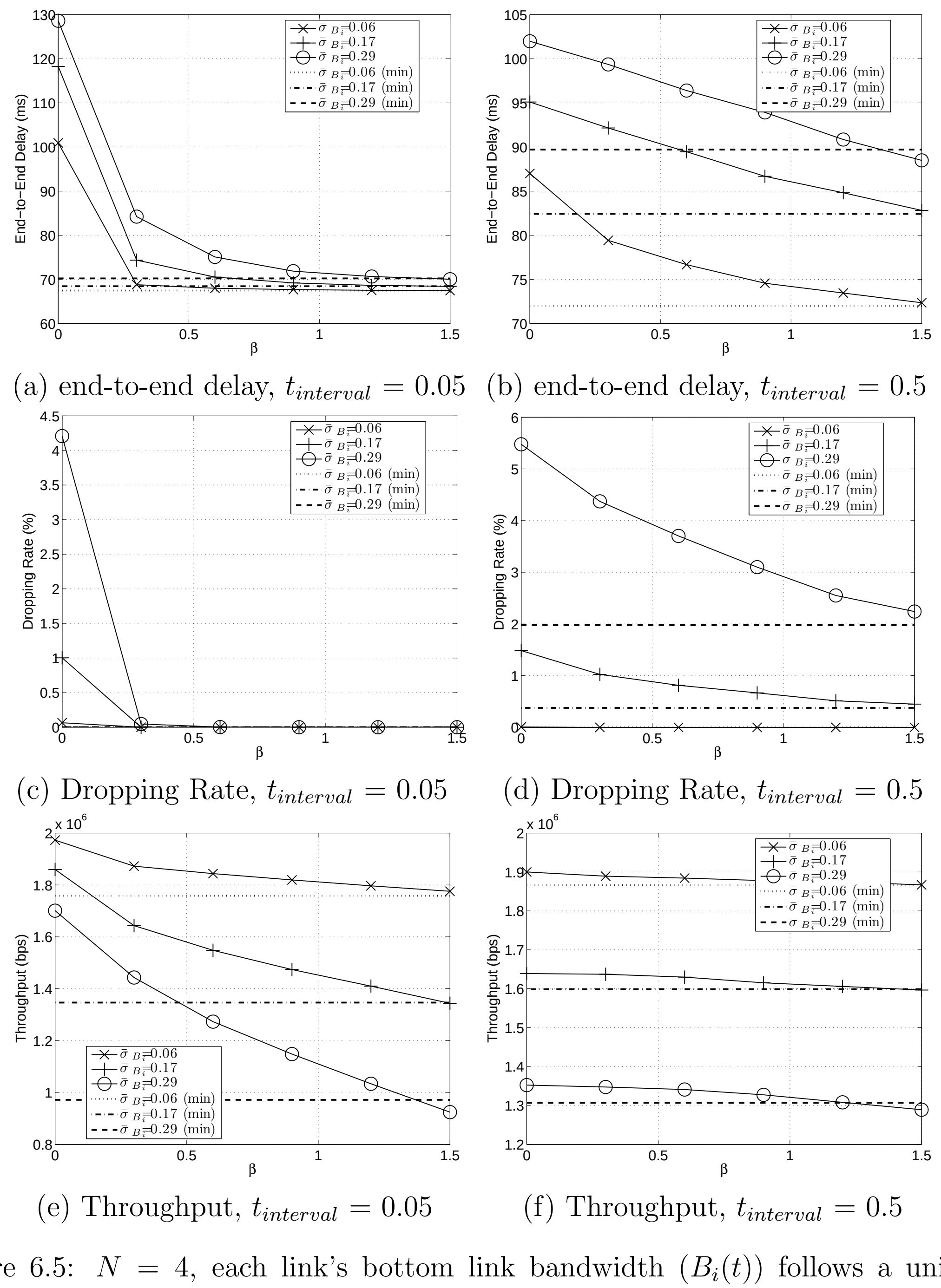


(d) End-to-End Delay

Figure 6.4: Comparison of HAA ($\beta = 0$) with PSP.

HAA are shown in the solid curves and the corresponding results for the minimum-bandwidth scheme in the dotted curves. Note that the results for the mean-only scheme correspond to the y-intercepts of the solid curves (β=0).

It can be seen that, there is a trade-off between the throughput and end-to-end delay/dropping rate. As β increases, the end-to-end delay and dropping rate decrease, but the effective throughput also decreases. It is the adaptive algorithm's task to find a proper value of β (refer to Section 5.2.5). Also, the effect of β becomes more significant for higher heterogeneity ($\bar{\sigma}_{B_i}$=0.29, open circles in the figures), especially for the dropping rate and throughput.


(a) end-to-end delay, $t_{interval} = 0.05$ (b) end-to-end delay, $t_{interval} = 0.5$

(c) Dropping Rate, $t_{interval} = 0.05$ (d) Dropping Rate, $t_{interval} = 0.5$

(e) Throughput, $t_{interval} = 0.05$ (f) Throughput, $t_{interval} = 0.5$

Figure 6.5: $N = 4$, each link's bottom link bandwidth ($B_i(t)$) follows a uniform distribution, with deviation $\bar{\sigma}_{B_i}$. "(min)" in the legend stands for the "minimum-bandwidth" scheme.

By utilizing the information on heterogeneity, the end-to-end delay (Figure 6.5-(a) and (b)) and the dropping rate (Figure 6.5-(c) and (d)) can be reduced significantly compared to the mean-only scheme (when β=0). The end-to-end delay and dropping rate in the case of a more rapidly varying bandwidth ($t_{interval}$=0.05 in Figure 6.5-(a), (c), and (e)) decrease even faster when the penalty factor (β) is increased, compared to the case of a slowly varying bandwidth ($t_{interval}$=0.5 in Figure 6.5-(b), (d), and (f)). However, the throughput also drops more for a more rapidly varying bandwidth. This is because when $t_{interval}$ is large (slowly varying bandwidth) and the extractor size (L) remains the same, the measured bandwidth deviation ($\bar{\sigma}_B$) would be less than the steady state (theoretical) deviation which is defined for a very long time duration, therefore, the throughput is degraded less.

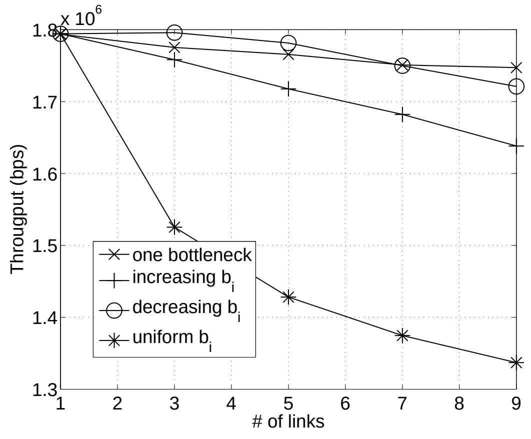
And as expected, the minimum-bandwidth scheme matches HAA when β is close to 1.5. The reason is that when the bandwidth variation follows a uniform distribution, the minimum value of the distribution is around $mean - dev * \sqrt{3}$, and $\sqrt{3} \approx 1.7$. Nevertheless, there is still a room for performance optimization. For example, when β is set to 1.0 for $t_{interval}$=0.05, performance of HAA is similar to that of the minimum-bandwidth scheme in terms of the end-to-end delay and dropping rate, although its throughput is significantly (at least 10%) higher than that of the minimum-bandwidth scheme.

It can also be seen that all the performance measures vary monotonically with β. Therefore, with the adaptive rate control scheme one could start from a conservative value of β, such as 2.0, which is slightly more conservative than the minimum-bandwidth scheme, and decrease β by a step size of $\Delta\beta$, until any of the requirements is violated (refer to Section 5.2.5).

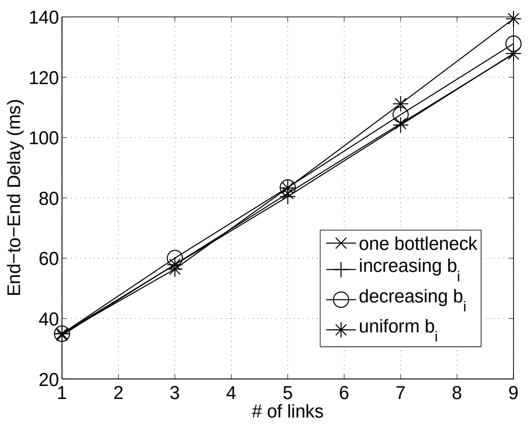
6.2.5 Effects of Spatial Heterogeneity

The effects of different spatial distributions of mean bandwidth (b_i) on each link i are studied and the results are shown in Figure 6.6. Four different scenarios are simulated on a path consisting of 1 to 9 links.

- scenario "one bottleneck": b_i is set to $2Mbps$ for the middle link and to $5Mbps$ for all the other links.
- scenario "increasing b_i": b_i linearly increases along the path from $2Mbps$ to $5Mbps$ (set to $2Mbps$ if there is only one link).
- scenario "decreasing b_i": b_i linearly decreases along the path from $5Mbps$ to $2Mbps$ (set to $2Mbps$ if there is only one link).
- scenario "uniform b_i": b_i is set to $2Mbps$ for all links.



(a) Throughput



(b) End-to-end delay

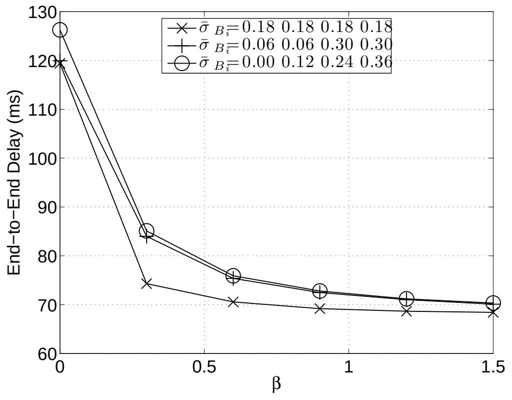
Figure 6.6: Effects of path length. All $\bar{\sigma}_{B_i}$ are set to 0.17.

Figure 6.6 shows that, the "uniform b_i" scenario has the worst performance in terms of throughput, because it does not have a fixed bottleneck, i.e., every link has the opportunity to be the bottleneck. According to the previous results (Equation 4.2) on effective bandwidth of a network path consisting of links with identical mean bandwidth, and the fact that the throughput can be no more than the effective bandwidth ($b_{effective}$) of the path, the throughput decreases when increasing the path length.

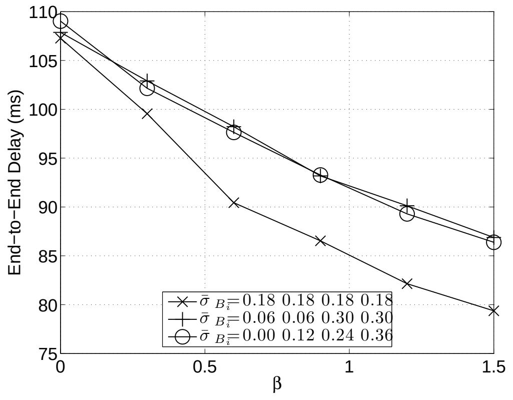
In the "one bottleneck" scenario, the throughput is almost independent of the path length. This is because the network path has only one bottleneck in a fixed position. All the other links' bandwidths are much larger than the bottleneck bandwidth in the first scenario, therefore, the range of the bandwidth variation of other links (from 3.5 to 6.5 $Mbps$) does not overlap with that of bottleneck link (from 1.4 to $2.6 Mbps$).

The "increasing b_i" and "decreasing b_i" scenarios fall between the two scenarios discussed earlier. It can be seen that, the throughput of the "increasing b_i" scenario is less and decreases more than that of the "decreasing b_i" scenario. This is because, when the links closer to the source node are slower, the faster links which are closer to the receiver node will be under-utilized. This effect becomes more significant when the path length increases.

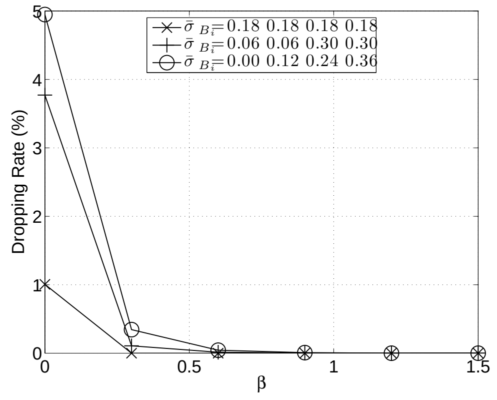
Figure 6.7 shows how different spatial distributions of temporal heterogeneity along the path affect the performance of the proposed scheme. Three different scenarios are simulated for a path of four links. Note that the average of $\bar{\sigma}_{B_i}$ along the path is the same in all cases. The first scenario (cross marks) has the smallest spatial heterogeneity in terms of link bandwidth variation (0.18 0.18 0.18 0.18) along the path, and the third scenario (circle marks) has the largest (0.0 0.12 0.24 0.36). It can be seen that the scenario with the largest spatial heterogeneity leads to the worst performance, with the lowest throughput and the largest dropping rate and end-to-end delay.



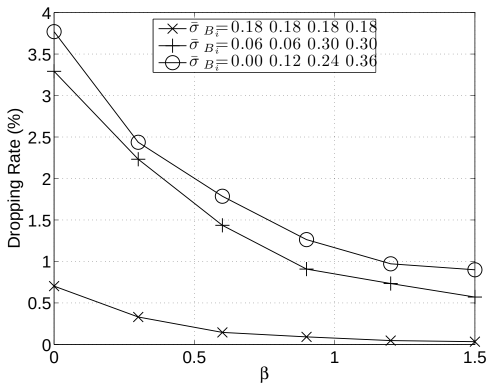
(a) end-to-end delay, $t_{interval} = 0.05$



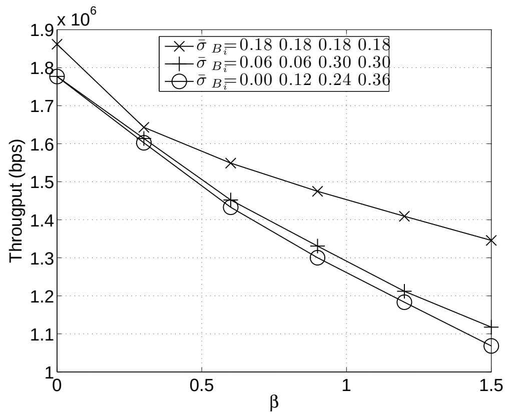
(b) end-to-end delay, $t_{interval} = 0.5$



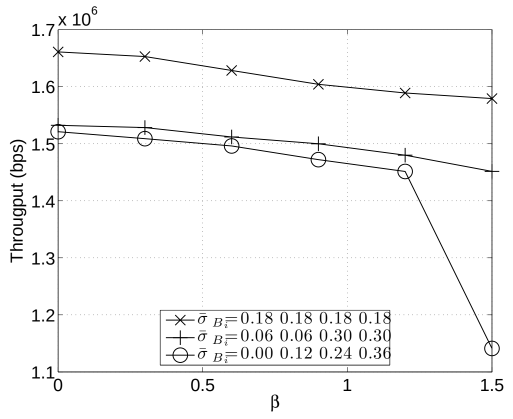
(c) Dropping Rate, $t_{interval} = 0.05$



(d) Dropping Rate, $t_{interval} = 0.5$



(e) Throughput, $t_{interval} = 0.05$



(f) Throughput, $t_{interval} = 0.5$

Figure 6.7: $N = 4$, heterogeneous links, all links have the same mean $ABBW_i$=$2Mbps$. Distributions of $\bar{\sigma}_{B_i}$ are as shown in the legends.

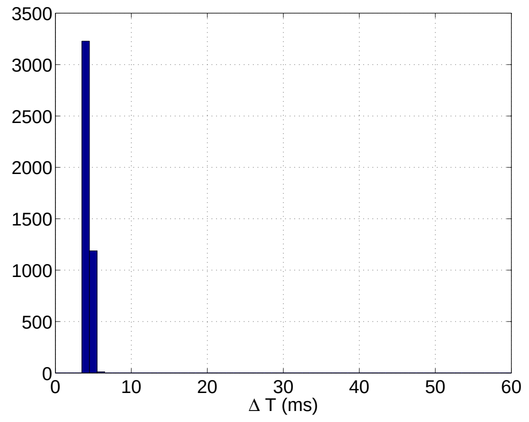
6.2.6 Packet Inter-arrival Time

The packet inter-arrival time (ΔT) is an important performance measure for a real-time media (video or audio) streaming application. A large variation of ΔT usually indicates a notable discontinuity or jitter on the receiver side. To provide a stable stream, ΔT should be as even as possible (assuming a fixed packet size). The histograms of ΔT for TCP and HAA flows are shown in Figure 6.8. It can be seen that ΔT has a much sharper histogram (i.e., a smaller variation of Δ) and a smaller mean (i.e., a higher average rate) for HAA than TCP. Also, note that there are some components located at much larger values than the main components (around 40ms for 4 links and 240ms for 12 links) in the TCP histograms. This is due to the bursty nature of TCP traffic.

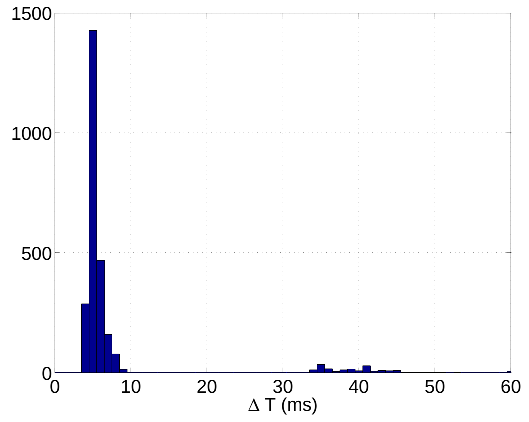
Figure 6.9 provides a qualitative comparison of TCP and HAA in the normalized deviation of ΔT ($\bar{\sigma}_{\Delta T}$), and it shows that $\bar{\sigma}_{\Delta T}$ of HAA is much smaller and less sensitive to (almost independent of) the path length than that of TCP.

6.2.7 Adaptive Penalty

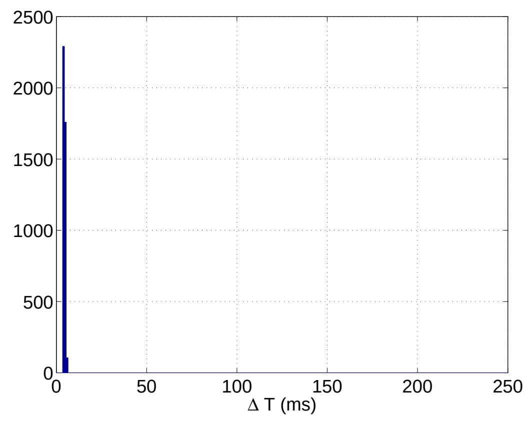
Figure 6.10 shows how the proposed adaptive scheme behaves when the upper-bound of end-to-end delay (Tee_{max}) is set to a specific value (0.07 seconds in this example). It is seen that β decreases from a conservative initial value of 1.9 until the QoS requirement for the end-to-end delay is violated (at around 25 seconds), and β converges to around 1.0, with the end-to-end delay close to 0.07.



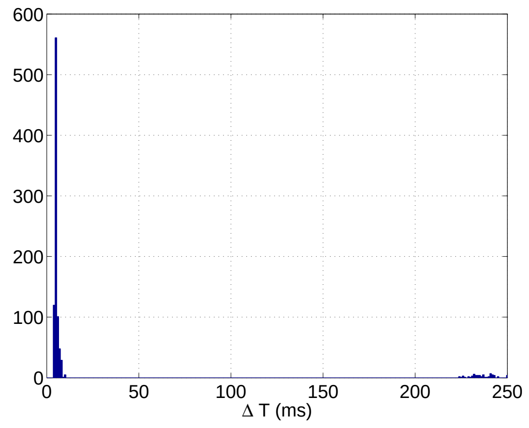
(a) 4 links (1 HAA flow)



(b) 4 links (1 TCP flow)



(c) 12 links (1 HAA flow)



(d) 12 links (1 TCP flow)

Figure 6.8: Histograms of inter-arrival times. 128 bins are used.

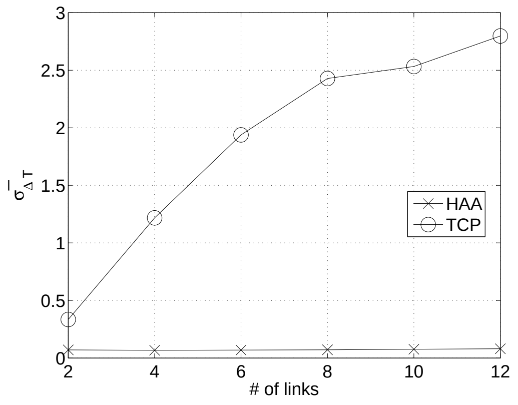


Figure 6.9: Deviation of inter-arrival time.

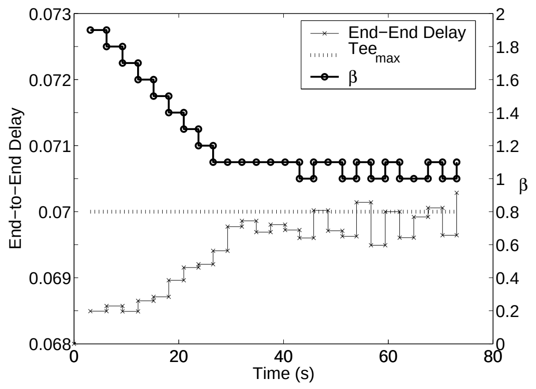


Figure 6.10: Illustration of the proposed adaptive rate control scheme.

6.2.8 Path Selection

Because the source rate (r_s), which is calculated on the receiver (refer to Equation 5.3), should be almost the same as the effective throughput of the network path when the dropping rate is very low, the solution to the path selection problem is straightforward.

An example network scenario consisting of 3 paths is shown in Figure 6.11. Note that both path 1 and path 2 have only one bottleneck link, while path 3 has two bottleneck links. However, the bottleneck bandwidth variation of path 3 is less than that of paths 1 and 2. Set the penalty factor (β) to 1.0 for all 3 paths so that the dropping rate is very low. The simulation results are shown in Figure 6.12.

It can be seen that even though path 3 has the lowest average bottleneck bandwidth (*abbw*) and the longest path length, it produces the highest effective throughput. This result reveals that *abbw* should not be the only measure used in the path selection problem, and that information on the bandwidth variation ($\bar{\sigma}_{ABBW}$) has to be considered. As can be seen in Figure 6.12, the source rate r_s matches the effective

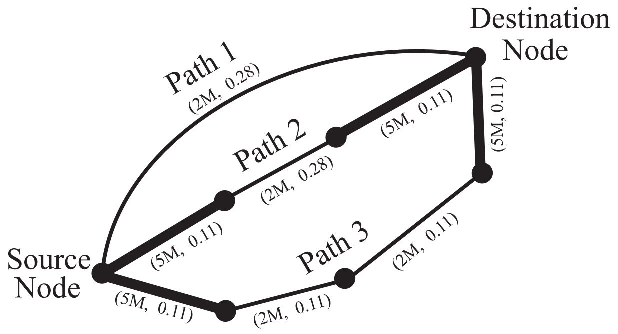
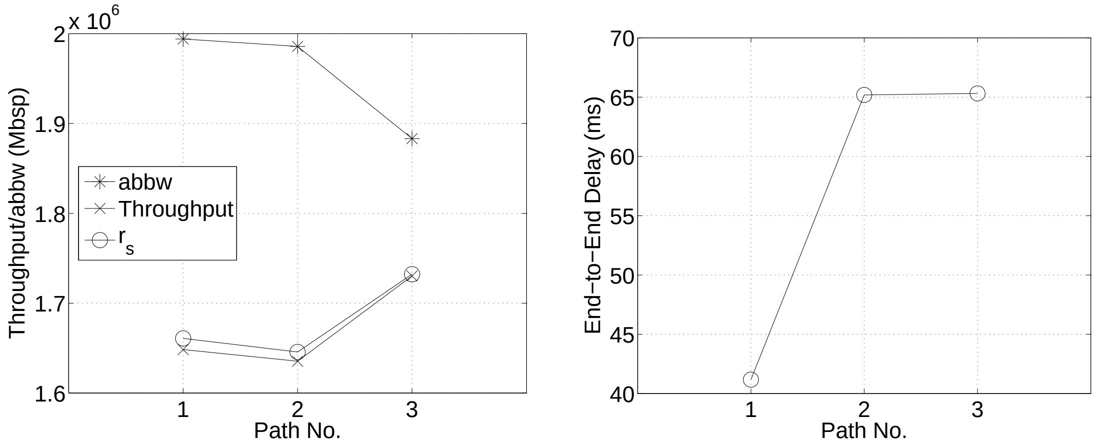


Figure 6.11: A parallel path where the number of links varies with path. The number pair in brackets over each link represents b_i and $\bar{\sigma}_{B_i}$ of the link.



(a) *abbw* and throughput (b) end-to-end delay

Figure 6.12: Throughput and End-to-End Delay when the penalty factor (β) is set to 1.0.

throughput quite well, therefore, it could be used as a good estimate of throughput when selecting the best path.

6.2.9 Multi-Path Media Streaming

The application of multi-path media streaming with inter-path synchronization is considered. The topology of the multi-sender (mirror) parallel data transmission system used in the simulation is shown in Figure 6.13, which is based on the framework developed in [47]. If all paths are synchronized "perfectly" and all connections are reliable, there will be no media packet missed or transmitted more than once. Note that due to heterogeneity, this perfect synchronization can never be achieved in a real network. In addition to the performance measures discussed in the previous sections (throughput, dropping rate, end-to-end delay), which are defined on the lower network layers, it is necessary to consider performance measures related to the degree of mis-synchronization on the application layer, *missing rate* and *duplicate rate*.

The missing rate is defined to be the ratio of “blank segments” to the whole length of the streams received. These blank segments are caused by both mis-synchronization on the application layer and packet dropping on the lower network layers. The duplicate rate is defined to be the ratio of duplicated (those received more than once) segments to the whole length of the stream received. Both are application-level measures and are calculated on the receiver side. To simplify the simulation, the segment size is set to be the same as the packet size.

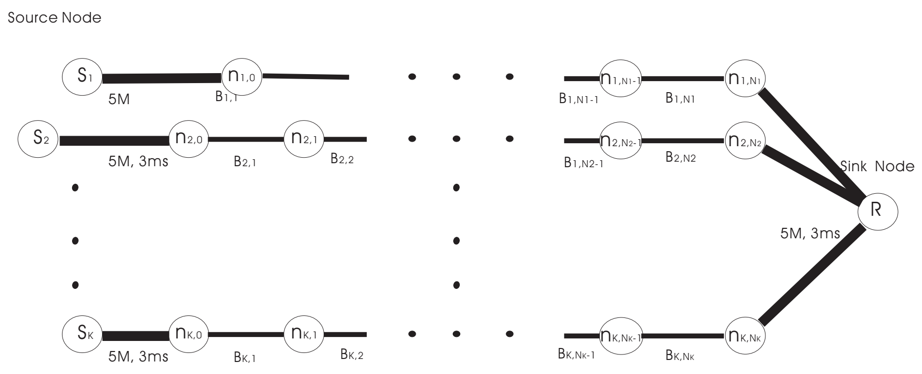
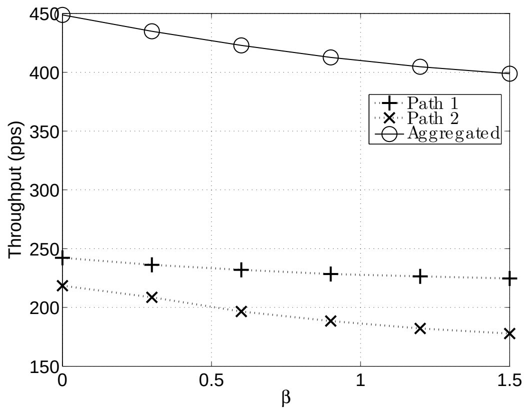


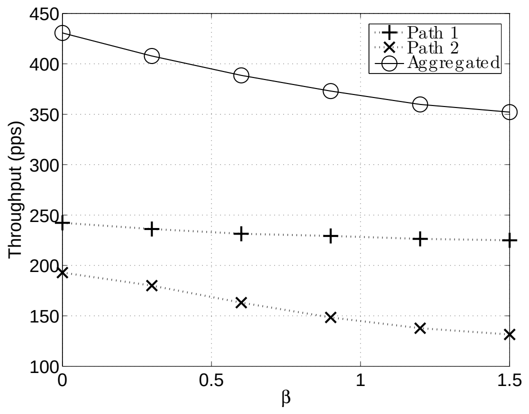
Figure 6.13: Simulation testbench topology for multi-path media streaming.

Simulation Results

Figures 6.14 and 6.15 show the simulation results for media streaming over a network with 2 parallel paths, each with 4 inter-router links. All links have the same average available bandwidth (b=2$Mbps$) and the links on path i have the deviation $\bar{\sigma}_{B_i}$ for $i = 1, 2$. It is seen that a higher source rate is allocated to path 1 which has a lower temporal heterogeneity ($\bar{\sigma}_{B_1}$=0.06), although the average available bandwidth is the same on both paths, as shown in Figure 6.14-(a) and (b).

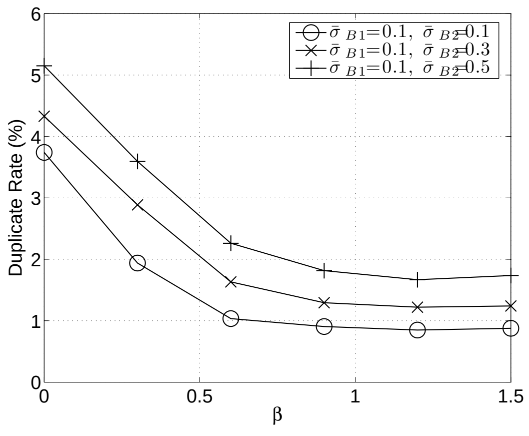


(a) $\bar{\sigma}_{B_1} = 0.06,\ \bar{\sigma}_{B_2} = 0.17$

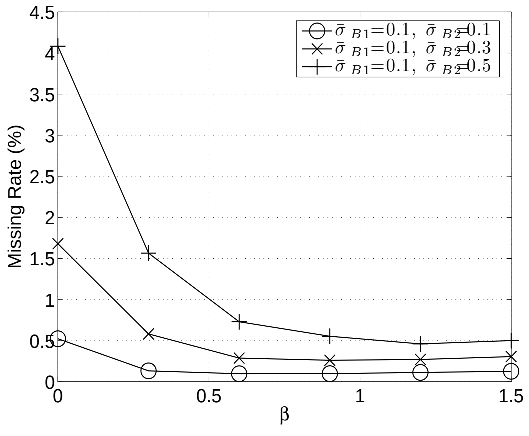


(b) $\bar{\sigma}_{B_1} = 0.06,\ \bar{\sigma}_{B_2} = 0.29$

Figure 6.14: 2 paths, each with 4 links, and all links have the same b_i=2$Mbps$, but different deviation $\bar{\sigma}_{B_i}$.



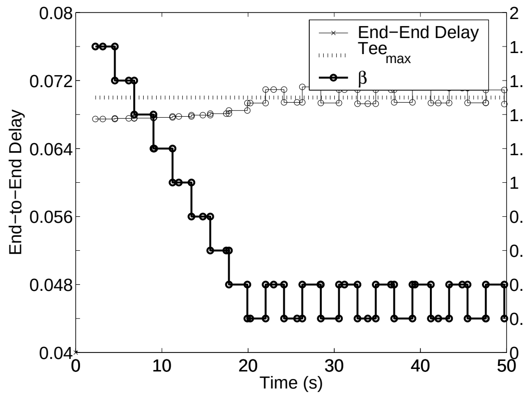
(a) Duplicate Rate



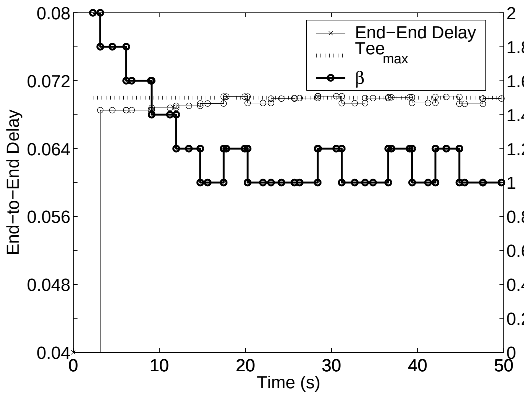
(b) Missing Rate

Figure 6.15: Application level measures for the scenario in Figure 6.14.

From Figure 6.15, it is interesting to see that both the missing and duplicate rates decrease significantly. When the penalty factor (β) increases from 0 to 1.5, the percentage reduction is up to 70% in the duplicate rate and 80% in the missing rate, while the throughput only decreases by 11% for the case of lower spatial heterogeneity ($\bar{\sigma}_{B_1} = 0.05,\ \bar{\sigma}_{B_2} = 0.17$) and 18% for the case of higher spatial heterogeneity ($\bar{\sigma}_{B_1} = 0.05,\ \bar{\sigma}_{B_2} = 0.29$) (Figure 6.14-(a) and (b)). This implies that the real-time QoS (missing and duplicate rates) can be improved without significantly degrading the effective throughput.



(a) Path 1



(b) Path 2

Figure 6.16: Adaptive rate control on a multi-path network.

Figure 6.16 shows how the proposed adaptive scheme works on a two-path network when the upper-bound of end-to-end delay (Tee_{max}) is set to 0.07 seconds. Note that β on each path varies independently and converges to a different value (about 0.3 on path 1 and about 1.0 on path 2). That is, it is shown that the proposed scheme is also able to handle this type of multi-path communication.

Performance Comparison with TCP

The proposed HAA scheme is compared with TCP on a 2-path network with each path consisting of 4 links. The results are shown in Figure 6.17. As expected, the aggregated throughput for HAA is much higher than that for TCP. However, the

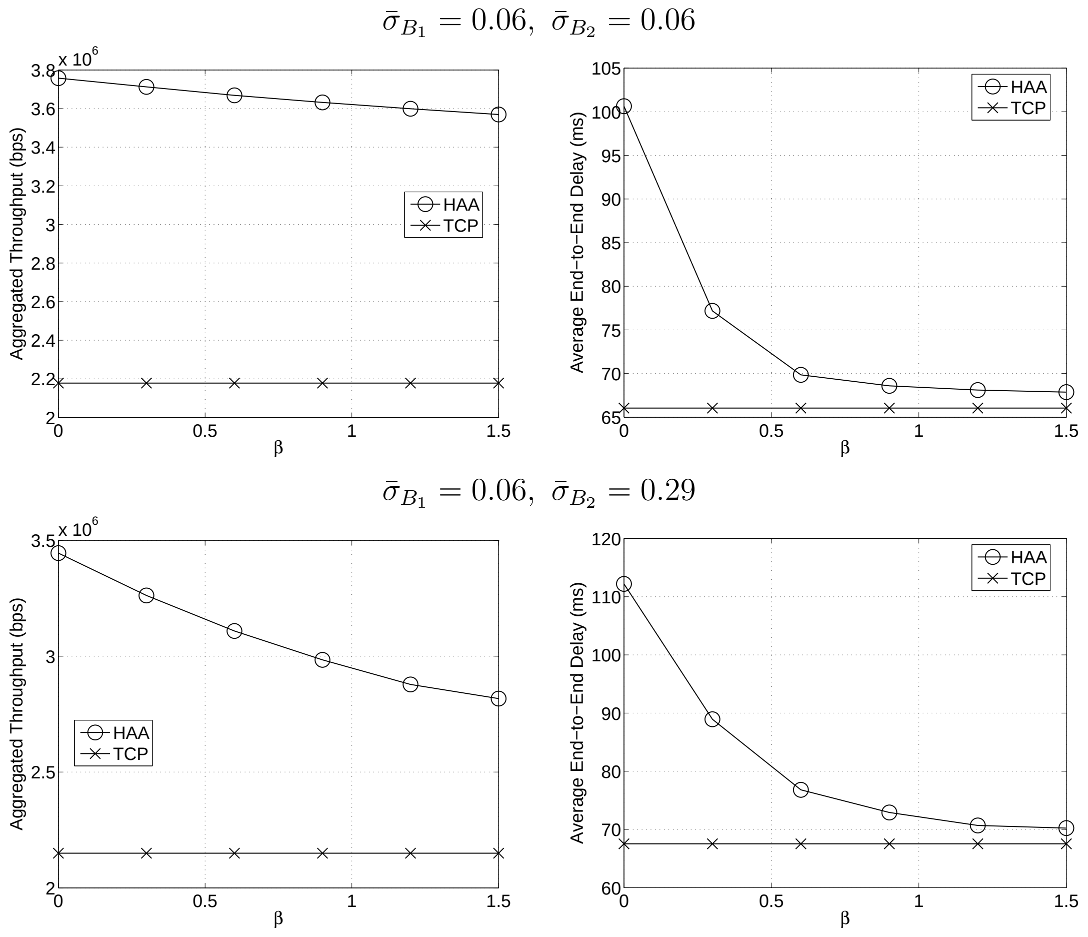


Figure 6.17: Comparison with TCP implementation for 2 paths, each consisting of 4 links.

end-to-end delay is also larger than that for TCP because more packets tend to be buffered under a higher transmission rate. By adjusting the penalty factor to 1.0, the End-to-End delay could be reduced significantly and the throughput can still be maintained at a much higher value than that by TCP.

Figure 6.18 compares HAA and TCP when the penalty factor (β) is fixed to 1.0. Two network paths are employed and the length of each path varies from 2 to 12. Note that the throughput achieved by TCP drops dramatically (more than 75%) as

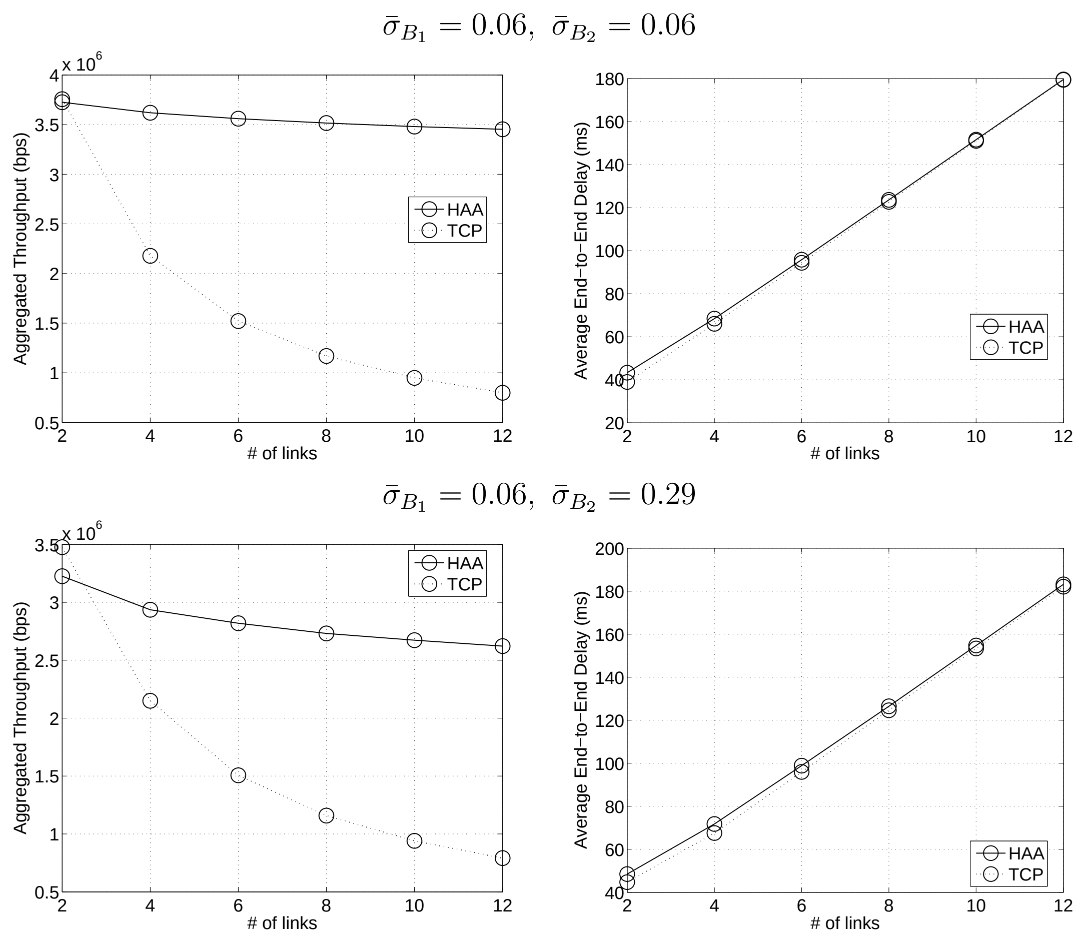
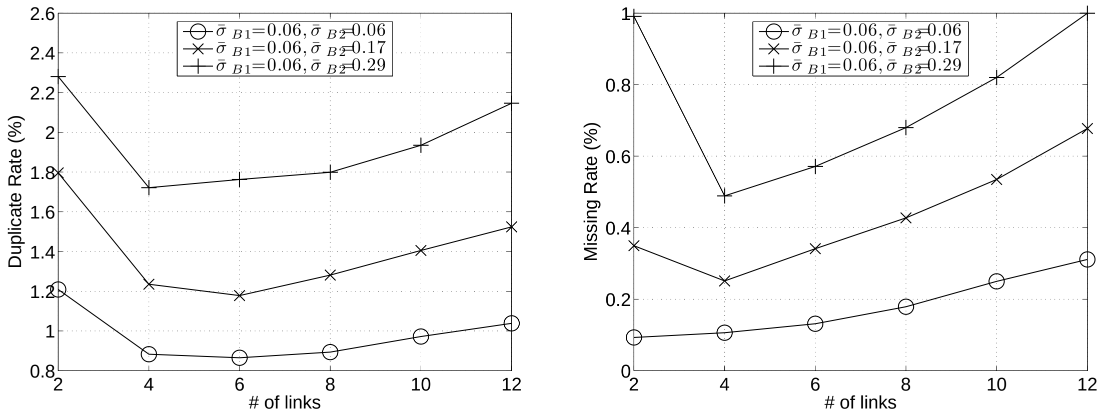


Figure 6.18: Comparison with TCP implementation (2 paths, β is fixed to 1.0).

the path length increases up to 12 links, but that by HAA only decreases by less than 15%.

Dependency of the missing and duplicate rates of HAA on the path length is shown in Figure 6.19. Both rates increase slightly when the network paths become longer, but are still at a reasonable level; the duplicate rate is around 2.3% and the missing rate is a little larger than 1.0% when the path length is 12.



(a) Duplicate Rate (b) Missing Rate

Figure 6.19: Application level measures (2 paths, β is fixed to 1.0).

6.3 Multiple Flows

All the results discussed above assume that the variation of the available bottleneck bandwidth ($ABBW$) is independent of the HAA source rate control, i.e., the introduction of HAA flows does not affect other background traffic. However, this may not be the case in real networks. Here, each flow is a part of background traffic of other flows and the dynamics of its source rate control can change the available bandwidth and behavior of other flows and, in turn, change its own available bandwidth.

In this section, multiple flows sharing a network path with varying bandwidth are studied. Figure 6.20 illustrates N_f flows sharing a network path consisting of N links. Each flow f_i is connected by a separate pair of sender (S_i) and receiver (R_i). Assuming that there is no coordination among the flows, i.e., the source rate of each flow is controlled independently by its sender and receiver pair. The queue management in this study adopts a basic “Drop-Tail-Round-Robin” scheme to ensure that all flows on the path have the same level of priority.

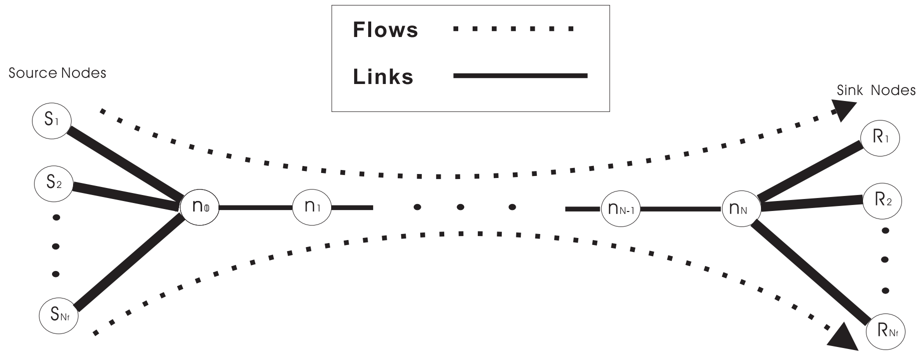


Figure 6.20: Topology of the testing network: multiple flows sharing a network path.

As a comparison reference, the bandwidth utilization of multiple TCP flows is first analyzed. Next, the performance of multiple HAA flows (with continuous and periodical On-Off traffic) in terms of aggregated throughput, average end-to-end delay, dropping rate, and fairness is investigated and compared with that of multiple TCP flows. Finally, TCP friendliness is verified to see if HAA flows can friendly share the network bandwidth with TCP flows.

6.3.1 Multiple TCP Flows

The aggregated throughput of multiple TCP flows is first investigated. According to the approximation model of TCP throughput from [48]

$$TH_{1TCP}(p) = min(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)}) \tag{6.1}$$

where W_{max} is the maximum window size, RTT is the round-trip time, T_0 is the timeout value, b is the number of packets of transmitted data that are acknowledged by one ACK from the receiver, and p is the loss ratio. It is not difficult to see that a single TCP flow throughput is no more than

$$TH_{1TCP} \leq \frac{W_{max}}{RTT} \tag{6.2}$$

And when there are N_f TCP flows, the aggregated throughput has an upper-bound of

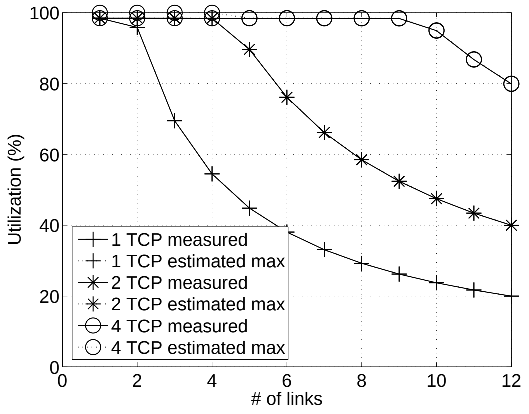
$$TH_{NTCP} \leq \sum_{i=1}^{N_f} \frac{W_{max_i}}{RTT_i} \tag{6.3}$$

From Equation 6.3, it can be seen that, when the maximum window size and N_f are fixed, a longer path length results in a larger RTT, and subsequently decreases the maximum possible throughput. This is verified by the simulation, and the results are shown in Figure 6.21. Both the measured (solid curves) and the estimated maximum utilization (dotted curves) are plotted, which are defined to be

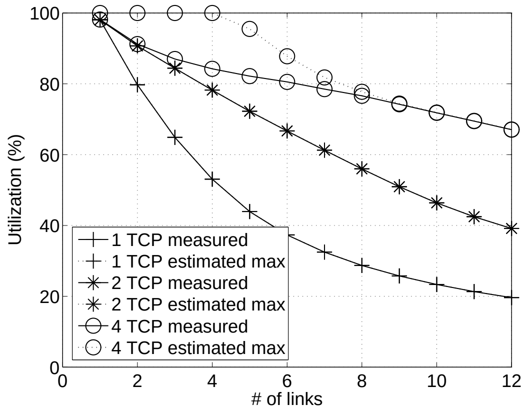
$$\begin{aligned} \text{Measured Utilization} &= \frac{TH_{NTCP}}{C} && (6.4) \\ \text{Estimated Maximum Utilization} &= min(1, \frac{N_f * W_{max}}{C * RTT}) && (6.5) \end{aligned}$$

where C is the capacity of the bottleneck link and it is set to $2Mbps$ in this simulation.

It can be seen from Figure 6.21 that the utilization becomes higher as more TCP flows are introduced, especially when the path is short. For example, on a path consisting of less than 9 links with no variation in $ABBW$, four TCP flows can achieve almost 100% utilization in the given scenario (Figure 6.21-(e)). However, as analyzed earlier, the bandwidth utilization drops significantly as the path length (number of links) increases, and it reduces even more when there is a larger available bandwidth variation ($\bar{\sigma}_B$).



(a) $\bar{\sigma}_{B_i}$=0.00



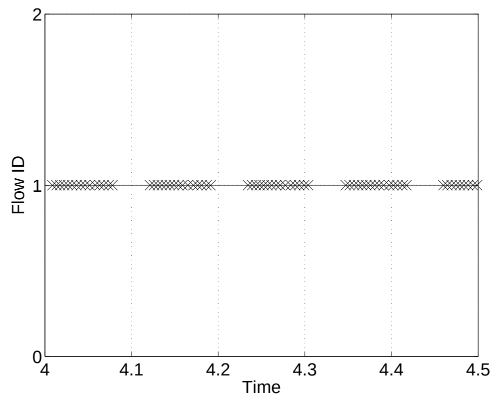
(b) $\bar{\sigma}_{B_i}$=0.29

Figure 6.21: Aggregated bandwidth utilization of multiple TCP flows. W_{max}=15. (note that some measured utilization curves overlap the estimated max ones)

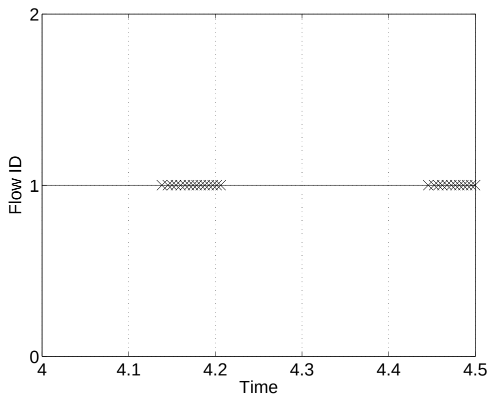
To further illustrate how TCP bandwidth utilization is affected by the number of flows and path length, the trace plots of packet arrival times for multiple (N_f=1,2, and 4) TCP flows are provided in Figure 6.22. The simulation is conducted on a network path consisting of two different numbers of links (N=4 links, and N=12 links). The result shows how the effective aggregated throughput varies as the path length increases. The $FlowID$ (from 1 to N_f) of each packet is plotted as a function of time, and each mark indicates a packet arrival event for the corresponding $FlowID$. The duration of each transition on $FlowID$ represents the time when no packet is being received, i.e., the idle time of the shared path.

It can be easily seen that multiple TCP flows are received by the receiver in a bursty and *multiplexed* manner. When there is only one TCP flow, the "duty cycle" is about 60% for a short path consisting of 4 links.

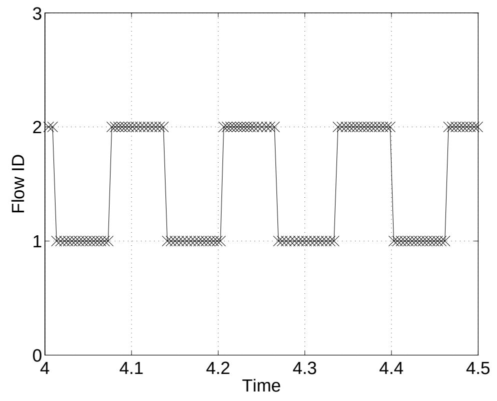
A larger number of flows increases the "duty cycle" significantly, for example, four TCP flows can achieve an almost full utilization on the 4-link path. However, it can also be clearly seen that the duty cycle (hence bandwidth utilization) shrinks as



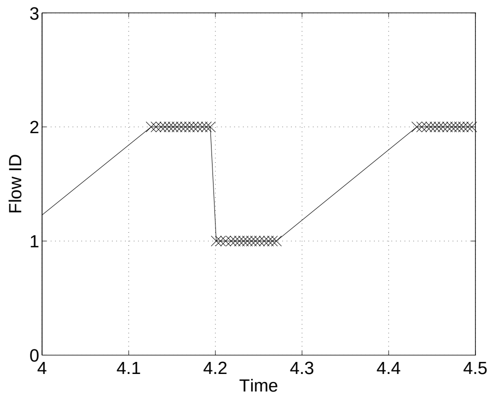
(a) 1 TCP flow, 4 links



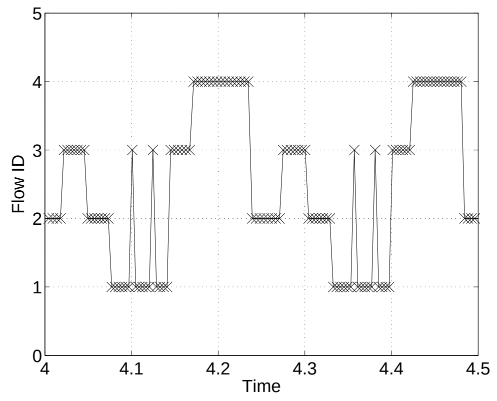
(b) 1 TCP flow, 12 links



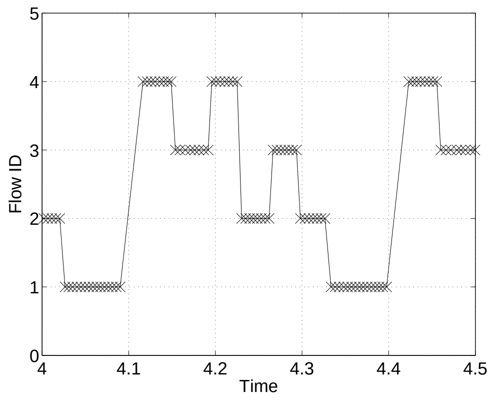
(c) 2 TCP flows, 4 links



(d) 2 TCP flows, 12 links



(e) 4 TCP flows, 4 links



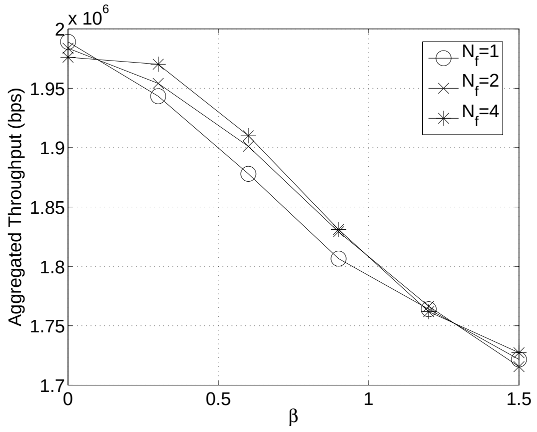
(f) 4 TCP flows, 12 links

Figure 6.22: Arrival time of multi-flow TCP packets.

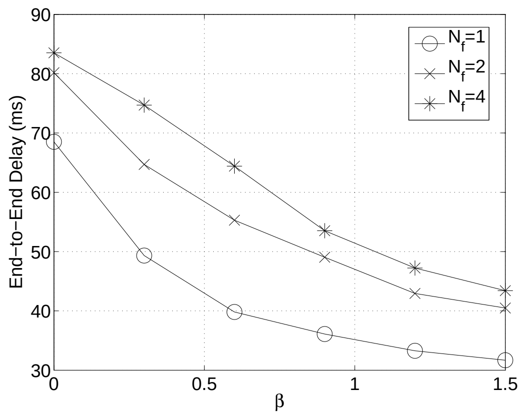
the path length increases. For example, when there is only one TCP flow, the duty cycle is only less than 20% on a 12-link path.

6.3.2 Multiple HAA Flows

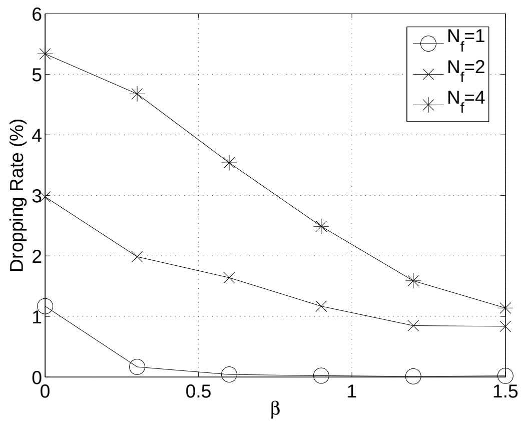
6.3.2.1 Overall Performance



(a) Aggregated Throughput



(b) End-to-End Delay

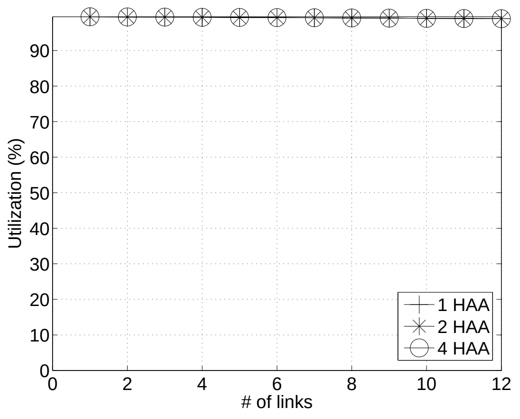


(c) Dropping Rate

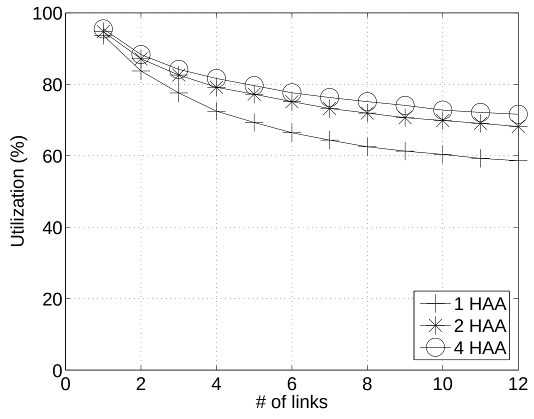
Figure 6.23: Multiple HAA flows sharing a network path consisting of 4 links, all links have the same b_i=$2Mbps$ and $\bar{\sigma}_{B_i}$=0.17.

Figure 6.23 shows the simulation results of N_f HAA flows sharing a varying available bandwidth on a network path consisting of 4 links. Each flow is sent continuously by the allocated source rate. It can be seen from Figure 6.23 that as the total number of HAA flows (N_f) increases, the aggregated throughput does not change very much (when β is fixed). This result shows that multiple HAA flows can still maintain the overall bandwidth utilization at almost the same level as that for a single HAA flow.

It should also be noted that, when introducing more flows, the dropping rate increases due to the interaction among multiple HAA flows and the limited buffer space available. By adjusting the penalty factor (β) to a larger value, the dropping rate can be reduced significantly. For example, when β is set to 1.5, the dropping rate is reduced to about 1.1%, but the aggregated throughput is degraded by no more than 12%.



(a) $\bar{\sigma}_{B_i}$=0.00



(b) $\bar{\sigma}_{B_i}$=0.29

Figure 6.24: Aggregated bandwidth utilization of multiple HAA flows.

Figure 6.24 shows the dependency of the bandwidth utilization by multiple HAA flows on the path length and the number of flows. When there is no variation in *ABBW* ($\bar{\sigma}_{B_i}$=0.00), multiple HAA flows can achieve almost 100% utilization and the utilization is almost independent of the path length. In the case with a larger bandwidth variation ($\bar{\sigma}_{B_i}$=0.29), the utilization decreases as the path length increases, but the effect of path length is much less significant than that in TCP. The comparison result will be shown in Section 6.3.3.

6.3.2.2 Dynamic Performance

To see if the HAA source rate control scheme produces any "exclusive flow" which permanently uses up all available bandwidth and refuses the injection of other flows, a scenario with On-Off flow patterns ("scenario On-Off") is designed. Instead of continuously sending out packets ("scenario Continuous"), each flow is toggled with a period of T_{switch} seconds. The duty cycle is set to be $\frac{2N_f-1}{2N_f}$, i.e., in each period, the flow is turned off for $T_{off} = \frac{T_{switch}}{2N_f}$ seconds, and turned back on for $T_{on} = \frac{T_{switch}(2N_f-1)}{2N_f}$ seconds. The switching time of each flow is evenly spaced out by T_{switch}/N_f so that at any time there is at least one flow active (unless N_f=1). A case with two On-Off flows is illustrated in Figure 6.25.

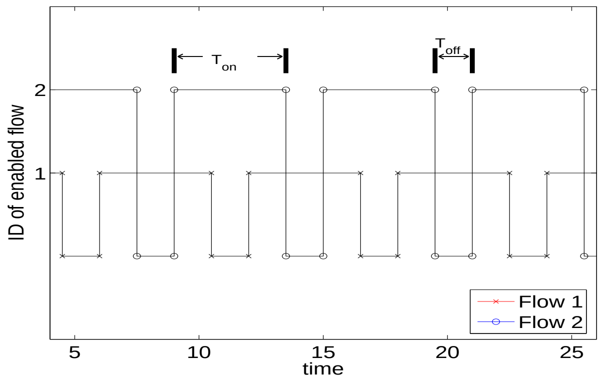
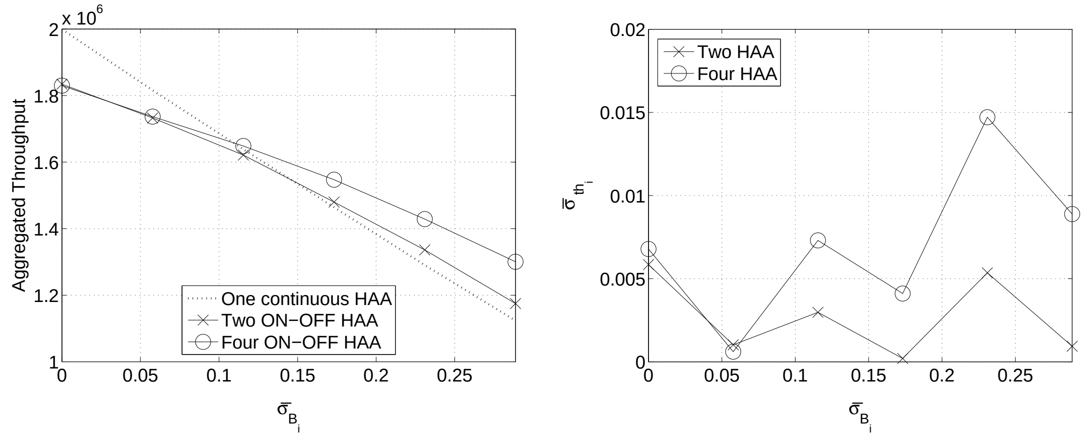


Figure 6.25: Two On-Off flows with T_{switch}=6.

When $N_f > 1$, the ideal aggregated throughput should be close to that in the scenario Continuous, because every time one flow is turned off, other active (enabled) flows are supposed to take over the bandwidth released by the closed flow.

Multiple On-Off flows are considered on a shared path consisting of 4 links in order to study the effects of bandwidth heterogeneity on the aggregated throughput.



(a) Aggregated throughput (b) Fairness among HAA flows

Figure 6.26: Dynamic performance of multiple On-Off flows. N=4, b_i=2$Mbps$ for all links, and β=1.0 for all flows.

The aggregated throughputs of 2 and 4 On-Off HAA flows are compared to the throughput of a single continuous flow for the same background traffic setup, and the results are shown in Figure 6.26-(a). It can be seen that the aggregated throughput of the scenario On-Off is close to the throughput of the scenario Continuous and even slightly better when there is a higher bandwidth heterogeneity ($\bar{\sigma}_{B_i} > 0.1$). This indicates that the active flows can, in a timely manner, take over the bandwidth released by the inactive flows. The normalized standard deviation of throughput among HAA flows in the scenario On-Off is plotted as an index of fairness in Figure 6.26-(b). It is seen that the throughput variation among the flows is very small (less than 2 percent), which means that there is no “irresponsible” flow.

6.3.3 Comparison with Multiple TCP Flows

6.3.3.1 Aggregated Bandwidth Utilization

The aggregated bandwidth utilization of multiple HAA flows is compared with that of multiple TCP flows. The improvement in bandwidth utilization is defined to be

$$\text{Improvement (\%)} = \frac{Ut_{HAA} - Ut_{TCP}}{Ut_{TCP}} \tag{6.6}$$

where Ut_{HAA} and Ut_{TCP} are the bandwidth utilizations of N_f HAA flows and N_f TCP flows, respectively. The penalty factor (β) of HAA flows is set to 1.0, and the maximum window size (W_{max}) of TCP is set to 15. The results are shown in Figure 6.27.

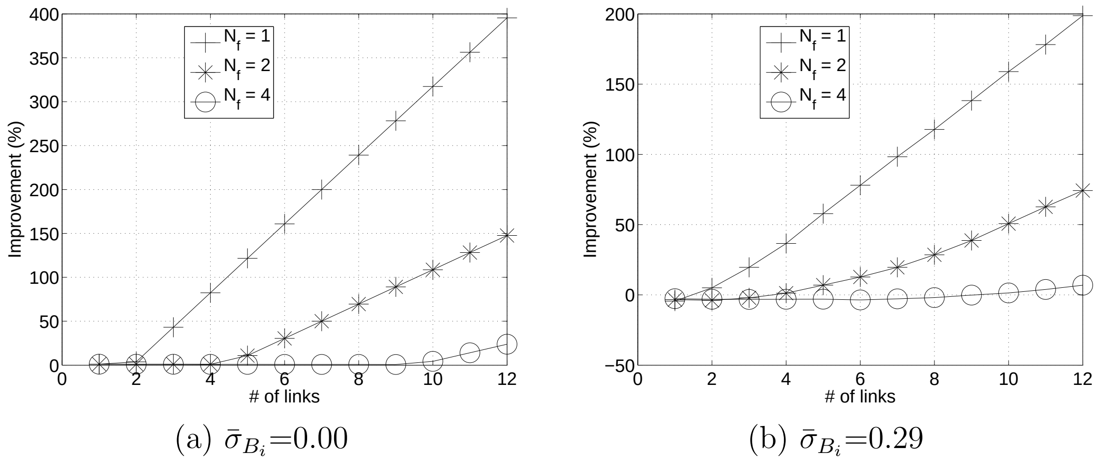
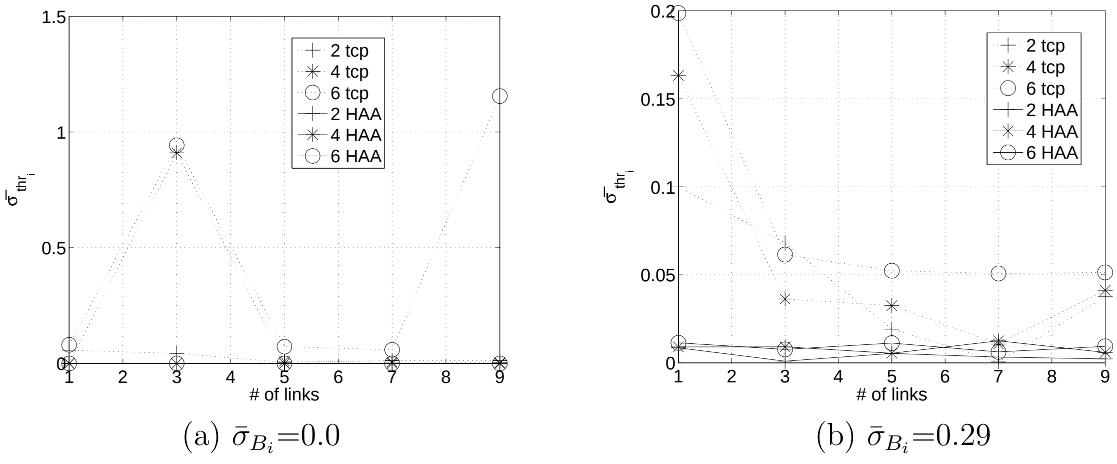


Figure 6.27: Improvement in aggregated bandwidth utilization.

The improvement in utilization becomes larger when the path length increases. Also, the improvement is more significant when network links have a smaller bandwidth variation ($\bar{\sigma}_{B_i}$=0.00). However, the improvement becomes smaller as the number of flows increases. For example, on the path of 4 links, the improvement by HAA over TCP is above 40% when there is only 1 flow, but there is almost no improvement when there are 4 flows.

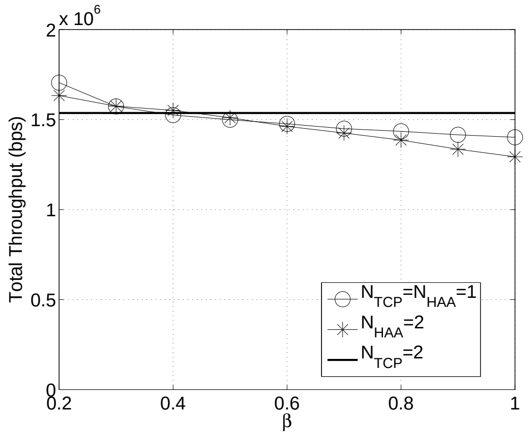
6.3.3.2 Inter-flow Fairness



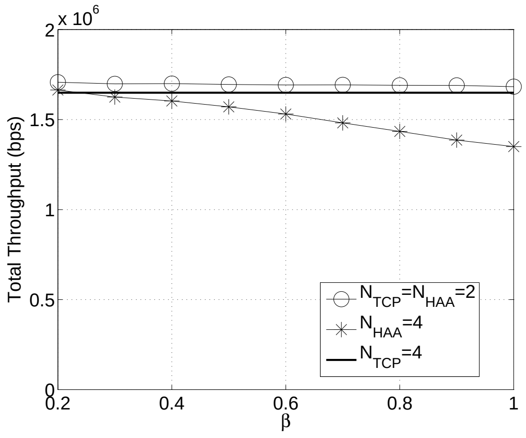
(a) $\bar{\sigma}_{B_i}$=0.0 (b) $\bar{\sigma}_{B_i}$=0.29

Figure 6.28: Fairness comparison with TCP.

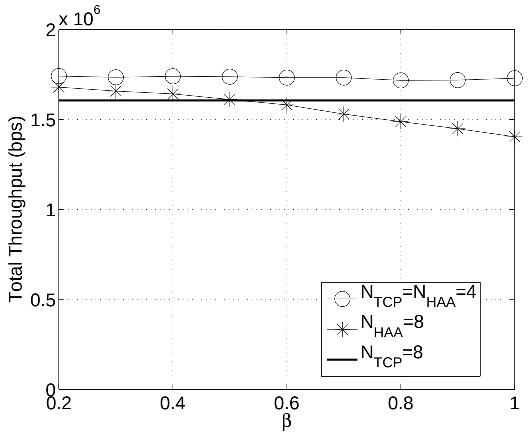
Figure 6.28 compares throughput variation $\bar{\sigma}_{thr_i}$, an index of inter-flow fairness, among multiple flows in both homogeneous ($\bar{\sigma}_{B_i}$=0.0) and heterogeneous ($\bar{\sigma}_{B_i}$=0.29) cases. It can be seen that in both cases the throughput variation among HAA flows is much smaller than that among TCP flows, i.e., the inter-flow fairness of HAA is better. Also, $\bar{\sigma}_{thr_i}$ is heavily dependent on the path length (or round-trip time) for TCP, but it is almost independent of the path length for HAA.



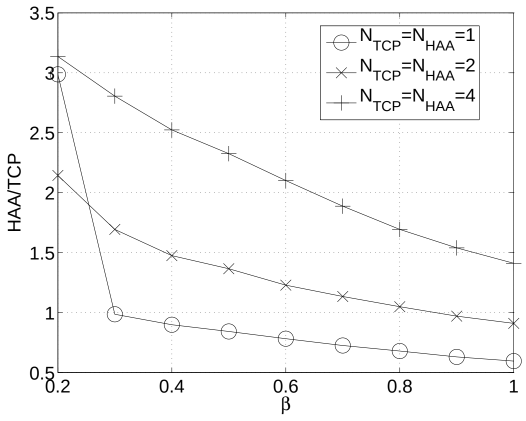
(a) 2 flows



(b) 4 flows



(c) 8 flows



(d) Throughput ratio (HAA/TCP)

Figure 6.29: Multiple HAA and TCP flows.

6.3.4 TCP Friendliness

Figure 6.29 shows the simulation results for multiple HAA flows sharing the available bandwidth with TCP flows on a network path of 4 links. In Figure 6.29-(a), (b) and (c), three sets of aggregated throughput are compared: (1) m HAA flows and m TCP flows (2) $2m$ HAA flows, and (3) $2m$ TCP flows, where m=1, 2 or 4. It can be seen that the throughput achieved by one TCP flow and one HAA flow together is slightly lower than that by two TCP flows when β is large (greater than 0.4), and is quite close to that by two HAA flows. However, when the number of flows is larger (4 or 8 flows), the mixed flow (HAA and TCP) achieves the highest throughput.

To prevent HAA flows from aggressively taking too large a share of the available bandwidth, β needs to be properly adjusted. Figure 6.29-(d) shows how to ensure that TCP flows get a “fairer” share by adjusting β. It can be seen that setting β to 1 usually achieves a fairly good bandwidth sharing between TCP and HAA flows, with a throughput ratio of between 1.0 and 1.5.

Chapter 7
Conclusion and Future Work

In this dissertation, two types of heterogeneity are defined for heterogeneous computing and communication systems. Temporal heterogeneity (TH) refers to variation in computing power or communication bandwidth available for a task along the time dimension on an individual computer or channel, while spatial heterogeneity (SH) refers to variation of the available computing power or communication bandwidth among computers.

Effects of TH and SH on computing and communication systems are first analyzed to show the possibility of improving the task performance by taking the heterogeneity information into account. In addition to the first order moment (mean), the second order moment (deviation) of resources availability is quantified and explicitly utilized to improve performance of computing and communication tasks in the heterogeneity aware approaches (HAA).

7.1 Computing Systems

The effects of heterogeneity are first studied in terms of average parallel execution time and standard deviation of parallel execution time on heterogeneous computing systems (Chapter 2). The results revealed that both TH and SH in computing power can have a significant effect on the parallel computing performance. To achieve the minimum τ on a spatially heterogeneous computing system, it is not optimal to distribute a target task linearly proportional to the average computing powers of computers.

If the computational load is perfectly balanced among computers in terms of the average computing power available, τ significantly increases as one or both types of heterogeneity (TH and SH) increase, while the average sequential execution time on a single computer is not affected by TH. As SH increases, τ increases more when the average TH (among computers) increases than when it remains fixed. As the number of computers employed for a target task increases, τ decreases initially and then may increase especially when SH increases fast. More frequent synchronization amplifies effects of heterogeneity and degrades performance of a target task more for a higher TH and/or SH. Performance degradation due to heterogeneity becomes even larger when the interval is longer (coarser granularity). The risk factor and variation of parallel execution time is quickly reduced as the fraction of a target task assigned to computers with smaller TH increases.

A two-step heuristic approach to partitioning a target task for minimizing its average parallel execution time (τ) is described with a theoretical model (Chapter 3). The proposed approach achieves a shorter τ by assigning a fraction of target task, which is larger than the fraction determined proportional to the average computing power, to a computer with a smaller TH or a higher average computing power.

The reduction in τ by the proposed approach is greater when (i) variation of the average computing power among computers is larger; (ii) variation of the standard deviation of computing power among computers is larger; or (iii) the number of computers employed is greater.

7.2 Communication Systems

The effects of TH on the QoS's such as throughput, end-to-end delay, and transfer time on a heterogeneous network are studied in Chapter 4. The effects become even larger when there also exists SH among links or paths. For the two applications, path selection and multi-path data transfer, it has been demonstrated that significant improvements in the end-to-end delay and transfer time are possible by considering the standard deviation as well as the mean of available bandwidth on each link.

A heterogeneity aware approach (HAA) to improving performance of communication tasks by utilizing the information on variation of available communication resources is proposed (Chapter 5). The proposed HAA source rate control scheme dynamically controls the source rate such that the rate is "penalized" more for a higher level of heterogeneity.

The performance of HAA source rate control has been analyzed in detail through simulation, and results may be summarized as follows:

- HAA can significantly increase the effective throughput and decrease the packet dropping rate, compared to the typical network measurement based approaches (the mean-only and minimum-bandwidth schemes) which do not utilize the heterogeneity information.

- Compared to TCP, the bandwidth utilization of HAA is much less sensitive to the path length (or round-trip time), and the aggregated throughput is even better than that of TCP when the path length is long. The inter-flow fairness of HAA is shown to be better. Also, HAA has a much less variation in inter-packet arrival time, i.e., it can produce a much more stable stream with less jitter than TCP. This is especially desirable in jitter-sensitive media streaming applications.

- The results from the multi-path media streaming application show that HAA can improve the real-time performance measures (missing and duplicate rates) without degrading the effective throughput substantially.

- It has been verified that the proposed scheme can be properly tuned to achieve fair sharing of bandwidth with other flows (HAA or TCP) on a network path and produce "responsible" traffic.

The proposed HAA source rate control scheme is very promising in various applications, especially where the real-time QoS's are concerned. By quantifying TH and SH in communication resources, it can produce a more comprehensive picture of resource variation and distribution, and provide a direct control of data transmission in a more predictable manner.

7.3 Future Work

Further improvements can be made for each component in the proposed schemes. For example, 1) more sophisticated resource measurement techniques can be designed to improve the accuracy of measurements; 2) the features (heterogeneity) of other network resources such as the packet buffer size can be utilized to further improve the performance; 3) the source rate function can be customized for different applications or services.

Also, applications to the following areas well deserve investigation:

7.3.1 Data-Intensive Computing Systems

So far, the HAA has been considered in computing and communication individually. However, in many applications, such as data-intensive high performance computing, both computing and communication are to be considered. It would be worthwhile to investigate interaction between them in application of the HAA to such tasks.

7.3.2 Application to Overlay Networks

Overlay networks have received a great deal of attention recently, and it can provide another framework for applying HAA without worrying about technical details on lower layers of the network (physical, data link layers, etc.) Resilient Overlay Network (RON) [49][50], Multicast Backbone (M-Bone)[51], and X-Bone [17][18][19] are three well-known overlay frameworks. Specifically, HAA can be applied to the path selection and dynamic routing problems of an overlay network.

Bibliography

[1] R. Buyya. High Peformance Cluster Computing. 1999.

[2] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1988.

[3] R. F. Freund and H. J. Siegel. Heterogeneous Processing. *Computer*, pages 13–17, June 1993.

[4] M. Maheswaran and H. Siegel. A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems. In *Proceedings of Heterogeneous Computing*, pages 57–69, 1998.

[5] T. Thanalapati and S. Dandamudi. An Efficient Adaptive Scheduling Scheme for Distributed Memory Multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, pages 758–768, July 2001.

[6] Y. Zhang, A. Sivasubrmaniam, J. Moreira, and H. Franke. Impact of Workload and System Parameters on Next Generation Cluster Scheduling Mechanisms. *IEEE Transactions on Parallel and Distributed Systems*, pages 967–985, September 2001.

[7] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking (TON)*, August 2001.

[8] S. Ali, H. Siegel, M. Maheswaran, D. Hensgen, and S. Ali. Task Execution Time Modeling for Heterogeneous Computing Systems. In *Proceedings of the 9th Heterogeneous Computing Workshop*, pages 185–199, May 2000.

[9] R. Armstrong. Investigation of Effect of Different Run-Time Distributions on SmartNet Performance. Master's thesis, Department of Computer Science, Naval Postgraduate School, 1997.

[10] J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson. Modeling Parallel Applications Performance On Heterogeneous Systems. In *Proceedings of IPDPS 2003, Workshop on Advances in Parallel and Distributed Computational Models*, Nice, France, April 2003.

[11] S. M. Figueira and F. Berman. A Slowdown Model for Application Executing on Time-Shared Clusters of Workstations. *IEEE Transactions on Parallel and Distributed Systems*, pages 653–670, June 2001.

[12] C.-Z. Xu, L. Y. Wang, and N.-T. Fong. Stochastic Prediction of Execution Time for Dynamic Bulk Synchronous Computations. In *Proceedings of International Parallel and Distributed Processing Symposium*, San Francisco, April 2001.

[13] X. Zhang and Y. Yan. Modeling and Characterizing Parallel Computing Performance On Heterogeneous Networks of Workstations. In *Proceedings of Seventh IEEE Symposium. Parallel and Distributed Processing*, pages 25–34, October 1995.

[14] A. Dogon and F. Ozguner. Trading off Execution Time for Reliability in Scheduling Precedence-Constrained Tasks in Heterogeneous Computing. In *Proceedings of International Parallel and Distributed Processing Symposium*, San Francisco, April 2001.

[15] J. Schopf and F. Berman. Stochastic Scheduling. In *CS Dept. Technical Report (CS-99-03)*, University of California, San Diego, 1999.

[16] D. Anderson, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, October 2001.

[17] J. Touch. Dynamic Internet Overlay Deployment and Management Using the X-Bone. *Computer Networks*, pages 117–135, July 2001.

[18] J. Touch and S. Hotz. The X-Bone. In *Proceedings of Third Global Internet Mini-Conference at Globecom*, Sydney, Australia, July 1998.

[19] J. Touch. Those Pesky NATs. *IEEE Internet Computing*, page 96, 2002.

[20] J. Kim and D. Lilja. Multiple Path Communication. *High Peformance Cluster Computing: Architectures and Systems*, pages 364–382, 1999.

[21] T. Nguyen and A. Zakhor. Path Diversity with Forward Error Correction System for Packet Switched Networks. In *Proceedings of INFORCOM 2003*, San Francisco, April 2003.

[22] J. Huang and S.-Y. Lee. Effects of Spatial and Temporal Heterogeneity on Performance of a Target Task in Heterogeneous Computing Environmentss. In *Proceedings of ISCA 15th International Conference on Parallel and Distributed Computing Systems*, pages 301–306, September 2002.

[23] S.-Y. Lee and J. Huang. A Theoretical Approach to Load Balancing of a Target Task in Temporally and Spatially Heterogeneous Grid Computing Environment. In *the 3rd International Workshop on Grid Computing*, pages 70–81, November 2002.

[24] M. Andrews, S. Borst, F. Dominique, P. Jelenkovic, K. Kumaran, K. G. Ramakrishnan, and P. Whiting. Dynamic bandwidth allocation algorithms for high-speed data wireless networks. *Technical Journal of Bell Labs*, July-September 1998.

[25] X. Liu, E. Chong, and N. Shroff. Transmission Scheduling for Efficient Wireless Utilization. In *Proceedings of the IEEE INFOCOM, Alaska*, pages 776–785, April 2001.

[26] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan. Achieving MAC Layer Fairness in Wireless Packet Networks. In *Proceedings of ACM Mobicom*, Boston, MA, August 2000.

[27] S. Lee and M. Gerla. Dynamic Load-Aware Routing in Ad Hoc Network. In *Proceedings of ICC 2001, Helsinki, Finland*, June 2001.

[28] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pages 289–298, September 1993.

[29] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Proceedings of ACM SIGCOMM*, September 1991.

[30] S. Keshav. Packet Pair Flow Control. *IEEE/ACM Transactions on Networking*, Feburary 1995.

[31] R. L. Carter and M. E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. *Performance Evaluation, TR-96-006, Boston University Computer Science Department*, March 1996.

[32] K. Lai and M. Baker. Measuring Bandwidth. In *Proceedings of IEEE INFOCOMM*, March 1999.

[33] K. Lai and M. Baker. Measuring Link Bandwidth Using a Deterministic Model of Packet Delay. In *Proceedings of the ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[34] Transmission Control Protocol Specification. *RFC 793*, September 1981.

[35] V. Jacbson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM*, pages 314–329, August 1988.

[36] Y. R. Yang, M. S. Kim, X. Zhang, and S. S. Lam. Two Problems of TCP AIMD Congestion Control. In *Technical Report TR-00-13*, Department of Computer Sciences, The University of Texas at Austin, June 2000.

[37] J. Widmer, R. Denda, and M. Mauve. A Survey on TCP-Friendly Congestion Control. *Special Issue of the IEEE Network Magazine Control of Best Effort Traffic*, pages 28–37, May/June 2001.

[38] D. Sisalem and A. Wolisz. LDA+ TCP-friendly adaptation: A measurement and comparison study. In *10th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'2000)*, pages 1619–1622, June 2000.

[39] L. Zhang, S. Shenker, and D. Clark. Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two Way Traffic. In *Proceedings of Proceedings of the ACM SIGCOMM 91 Conference on Communications Architectures and Protocols*, pages 133–147, 1991.

[40] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *Proceedings of the IEEE INFOCOM Conference on Computer Communications*, pages 1157–1165, 2000.

[41] A. C. Feng and A. C. Kapadia. Packet Spacing: An Enabling Mechanism for Delivering Multimedia Content in Computational Grids. *The Journal of Supercomputing*, pages 51–66, 2002.

[42] A. Kapadia, A. Feng, and W. Feng. The Effects of Inter-Packet Spacing on the Delivery of Multimedia Content. In *Proceedings of The 21st IEEE International Conference on Distributed Computing Systems*, 2001.

[43] H. David. *Order Statistics*. John Wiley and Sons Inc, 1970.

[44] K. Fall and K. Varadhan. The ns Manual (formerly ns Notes and Documentation). *web resource available at http://www.isi.edu/nsnam/ns/ns-documentation.html*.

[45] E. Gafni and D. Bertsekas. Dynamic Control of Session Input Rates in Communication Networks. *IEEE Trans. on Automatic Control*, pages 1009–1016, January 1984.

[46] A. G. Fraser. Designing a Public Data Network. *IEEE Communications Magazine*, pages 31–35, October 1991.

[47] T. Nguyen and A. Zakhor. Distributed Video Streaming over Internet. In *Proceedings of SPIE Multimedia Computing and Networking*, pages 186–195, San Jose, California, January 2002.

[48] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, September 1998.

[49] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of 18th ACM SOSP*, Banff, Canada, October 2001.

[50] D. G. Andersen. Resilient Overlay Networks. Master's thesis, Massachusetts Institute of Technology, May 2001.

[51] M. R. Macedonia and D. P. Brutzman. MBone Provides Audio and Video Across the Internet. *IEEE Computer*, pages 30–36, April 1994.