ABSTRACT

XINBING WANG. Performance Analysis of TCP over Wired and Wireless Networks. (Under the direction of Professor Do Young Eun.)

Transmission Control Protocol (TCP) currently accounts for about 90% applications and 80% data of network traffic, and plays the dominant role in Internet transmission. In this dissertation, we present three studies of TCP performance over wired and wireless networks. In the first study, we mainly focus on the stability of TCP/AQM (Active Queue Management) systems for wired network environment. In particular, we study the local and global stability of TCP-newReno/RED under many flows regime. By using a normalized discrete-time model, which is simple, we analyze the global stability in a very efficient manner, and the results show that by properly choosing RED parameters, we can always make the TCP-newReno/RED system globally stable.

The second study concerns the TCP performance when the buffer size is scaled up differently from the traditional one at a link with large capacity shared by many flows. Specifically, we consider the buffer size chosen on the order of (N^α) $(0 < \alpha < 1)$, where N flows share the link of capacity $O(N)$. We then develop a doubly-stochastic model for a TCP/AQM system with many flows by taking into account the packet-level dynamics over fine time scales. We show that, under our scale, the system always performs well in the sense that the link utilization goes to 1 and the loss ratio decreases to zero as the system size N increases. We verify our results using extensive ns-2 simulations.

Finally, we then analyze the impact of TCP flows on the wireless networks from the resource allocation point of view. We propose a TCP-AIMD aware call admission control scheme over wireless networks. Our scheme is based on a two level CAC framework, which takes both the call level and the packet level dynamics into account. The interaction between two levels is characterized through a single metric called the Quality of Service (QoS) guaranteed capacity. The performance of the proposed scheme is then analyzed and extensive simulation results are presented under different scenarios. Our results show that the proposed scheme can improve the system performance both on call and packet levels.

Performance Analysis of TCP over Wired and Wireless Networks

by

Xinbing Wang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical and Computer Engineering

Raleigh

2006

Approved By:

Dr. Arne Nilsson	Dr. Peng Ning
Dr. Wenye Wang	Dr. Robert Martin

Dr. Do Young Eun
Chair of Advisory Committee

To my parents and my wife.

献给我的父母和妻子。

Biography

Xinbing Wang was born in Jilin, China. He received the B.S. degree (with honors) from the Department of Automation, Shanghai Jiaotong University, Shanghai, China, in 1998, and the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2001. From 2001 to 2006, he has been pursuing his Ph.D. degree major in the Department of Electrical and Computer Engineering, minor in Department of Mathematics, North Carolina State University, Raleigh, NC, USA. From 2006, he will be a faculty member in Department of Automation, Shanghai Jiaotong University, Shanghai, China. His research interests include Resource Allocation and Management in Mobile and Wireless Networks; TCP Asymptotics Analysis; Wireless Capacity; Cross Layer Call Admission Control; Asymptotics Analysis of Hybrid Systems, and Congestion Control over Wireless Ad-hoc and Sensor Networks.

Acknowledgements

First and foremost, I want to thank my advisor, Professor Do Young Eun, who led me into this exciting area of networking performance analysis. As his first Ph.D. student, I am very fortunate to share more opportunities to work closely with him. I cannot imagine how much time and patience he spent to train such a 'stubborn' student like me and explain the problems, ideas, thinking, insights and the way how to be a genuine researcher, which becomes the most valuable experience in my life. I am privileged to have such a wonderful advisor, who is always enthusiastic, inspired, patient, helpful and encouraging. I cannot overstate my gratitude and appreciation for his encouragement and support. All I can say is that I can hardly ask for any more from an advisor, I will also try to be such a devoted teacher and a great mentor to my future students, I will make sure that when they write acknowledgements, the thanks also go to Professor Eun. At the same time, I would like to thank my co-advisor Professor Wenye Wang for her great patience and the extensive training, without which I could not have finished my Ph.D. study. I would also give many thanks to Professor Peng Ning, Professor Arne Nilsson and Professor Robert Martin for teaching me important research tools and for their kind guidance. In addition, I am particular grateful to Professor Bibhuti B. Bhattacharyya in Statistics Department and Professor Min Kang in Mathematics Department, who taught me a rich of modern probability and stochastic processes theory that will definitely benefit me in the rest of my life. Thanks also to Professor Xiuli Chao and Professor Shu-Cherng Fang in Industry Engineering Department for enlightening me with queueing theory and programming theory, respectively.

Besides these professors, I feel very lucky to meet my friends like Tie Liu, Peng Xu, Xiang Zhou, Haidong Guo, Tiejun Li, Dingbang Xu, Pan Wang, Hua Li, Chuan Lin, Yipeng Yang and many others, who give a lot of help to me or have valuable discussions with me. Also, I am grateful to my officemates including Yuh-Ming Chiu, Han Cai, Sungwon Kim, Terrence A. Vanvalkenhoef, Wei Liang, Ming Zhao, Fei Xing, Yi Xu, Jangeun Jun, Rong Peng and Suyoung Yoon, etc.

Finally, I dedicate this dissertation to my parents and my wife, who unconditionally love me and support me during the various difficulties and hard times during the course of my five-year Ph.D. study. Meanwhile, their love makes my life meaningful and enjoyable.

Contents

List of Figures viii

List of Tables xii

1 Introduction 1
- 1.1 Motivation . . . 2
  - 1.1.1 TCP Stability Analysis . . . 2
  - 1.1.2 TCP Performance under a Large Scale . . . 4
  - 1.1.3 TCP Performance over Wireless Networks . . . 6
- 1.2 Contribution . . . 8
- 1.3 Outline . . . 11

2 Background of TCP 12
- 2.1 TCP over Wired Networks . . . 12
- 2.2 TCP over Wireless Networks . . . 24
- 2.3 Other Related Work of TCP . . . 28
- 2.4 Active Queue Management: AQM . . . 34

3 Stability Analysis of TCP under Linear Scale 45
- 3.1 Introduction and Background . . . 45
  - 3.1.1 Definition of Lyapunov Stability . . . 46
  - 3.1.2 Problems with the Existing Results of Stability . . . 47
  - 3.1.3 Motivation and Objectives . . . 50
- 3.2 System Model . . . 51
  - 3.2.1 Problem Description . . . 51
  - 3.2.2 Normalized Model . . . 52
  - 3.2.3 Model Comparison . . . 54
- 3.3 Local Stability Criterion . . . 55
  - 3.3.1 Linearizing System around Equilibrium Point . . . 57
  - 3.3.2 Local Stability Criterion . . . 59
- 3.4 From Local Stability to Global Stability . . . 61
  - 3.4.1 Convergence From Outside a Box . . . 62

3.4.2 Choosing Free Parameters to Ensure Global Stability 63
3.4.3 Discussion 66
3.5 General AIMD (a, b) 66
3.5.1 Local Stability Criterion 68
3.5.2 Discussion 70
3.6 Heterogeneous RTTs 70
3.6.1 Normalized Model for Heterogeneous RTTs 73
3.6.2 Equilibrium Point for Heterogeneous RTTs 74
3.6.3 Stability Condition for Heterogeneous RTTs 75
3.6.4 Decomposition of Stability Condition for Heterogeneous RTTs . . . 76
3.6.5 Discussion 78
3.7 Simulation Results 79
3.7.1 Homogeneous RTTs 80
3.7.2 Heterogeneous RTTs 80
3.8 Conclusion 82

4 TCP Performance under Aggressive Scale 84
4.1 Preliminaries 84
4.1.1 Scaling Buffers and AQMs inside a Network 84
4.1.2 Aggressive Packet Marking 85
4.1.3 A Closer Look at the Queue Dynamics 87
4.2 Main Results 89
4.3 Numerical Results 90
4.3.1 AQM Configurations 92
4.3.2 Homogeneous RTTs 92
4.3.3 Heterogeneous RTTs 95
4.3.4 Mixture of Long-lived and Short-lived Flows 96
4.3.5 Multiple Links with Heterogeneous RTTs 99
4.4 Conclusions 101

5 Control TCP over Wireless Networks: A Resource Allocation Point of View 102
5.1 Preliminary: TCP AIMD Algorithm 103
5.2 Framework: A Two-Level Model 104
5.3 TCP-AIMD Aware CAC Scheme 107
5.3.1 Discussion on the Packet Loss in the Middle Links of Networks . . . 110
5.3.2 Further Discussions on Implementing Rate Control in Reality 112
5.4 Performance Analysis: Packet Level 116
5.4.1 $M/M/1$ Model 117
5.4.2 A Priority Queue Model 118
5.4.3 Effective Bandwidth Model 120
5.4.4 Gaussian Approximation Model 121
5.4.5 Discussion 122
5.5 Performance Analysis: Call Level 123
5.5.1 Call Blocking Probability: A General Form 123

5.5.2 Call Blocking Probability for Case I 125
5.5.3 Call Blocking Probability for Case II 126
5.5.4 Insensitivity Discussion 128
5.6 Simulation Results 129
5.6.1 Case I: Independent Transmission Time 130
5.6.2 Case II: Dependent Transmission Time 137
5.7 Conclusion 142

6 Conclusions and Future Work **144**
6.1 Conclusions 144
6.2 Future Work 145

Bibliography **147**

List of Figures

2.1 Performance: Tuning TCP buffers and parallel streams [48]. 13
2.2 Performance Comparison: Aggregate Bandwidth. Auto: the proposed scheme; Hiperf: Tuning by Hand; Default: Default setting by NetBSD [176]. 14
2.3 Performance Comparison: Peak usage of memory. Auto: the proposed scheme; Hiperf: Tuning by Hand; Default: Default setting by NetBSD [176]. 15
2.4 Illustration: Traditional TCP and STCP [106]. 16
2.5 Fairness of HSTCP over regular TCP [70]. 17
2.6 TCP Fairness [197]. 18
2.7 Fairness of TCP SABUL [78]. 21
2.8 TCPWestwood and TCP-Reno: Throughput vs. bottleneck bandwidth [144]. 22
2.9 TCPWestwood and TCP-Reno: Throughput vs. bottleneck buffer size [144]. 23
2.10 Interplanetary Internet [8]. 23
2.11 State Transition Diagram [8]. 24
2.12 Three Types of Loss in Cellular Networks. 25
2.13 Two State Markov Chain for Loss Channel [83]. 25
2.14 Network Scenario for 802.11b WLAN [29]. 26
2.15 Three State Markov Chain for Error [29]. 26
2.16 An Example of Bandwidth Variation over time [20]. 27
2.17 An Example of Round Trip Time Variation over time [20]. 27
2.18 Illustration of TCP Connection Establish [37]. 32
2.19 Throughput Comparison of Tahoe, Reno and Sack for (a), (b) correlated and (c) independent losses [154]. 35
2.20 Performance of RED compared to Droptail. First column is droptail; Second column is RED without ECN; Third column is RED with ECN [69]. 37
2.21 REM Marking Function compared to RED [18]. 38
2.22 AVQ: Virtual Queue is not full [122]. 39
2.23 AVQ: Virtual Queue is full [122]. 40
2.24 AVQ Algorithm in Pseudocode [122]. 40
2.25 AVQ Performance compared to RED, PI, REM, GKVQ with regard to Average Queue Length [122]. 41
2.26 BLUE: Core Algorithm in Pseudocode [60]. 42
2.27 Blue: Throughput [60]. 42

2.28 PI Controller [35]. 44

3.1 The stability convergence with the number of flows N 48
3.2 An Example to illustrate the relationship between RED slope ρ and RTT: the round trip physical propagation delay is 200 ms; capacity for each flow C is 10 packets/sec. 49
3.3 System model and RED algorithm. 52
3.4 Model comparison when the system is unstable. 55
3.5 Model comparison when the system is stable. 56
3.6 Model comparison when the system is stable. 57
3.7 The Bound of Q(t) and W(t). 62
3.8 Example I: the stable region of Q(0) and W(0) with $Q(\infty) = 1.2764$, $W(\infty) = 12$ for $\Delta q = 0.3, 0.4, 0.5, 0.6$. 64
3.9 Example II: the stable region of Q(0) and W(0) with $Q(\infty) = 3.8237$, $W(\infty) = 6.70$ for $\Delta q = 0.05, 0.07, 0.08, 0.1$. 65
3.10 The impact of b and a to the system stability with $C = 20$, $\rho_1 = 0.01$. . . . 71
3.11 The impact of b and a to the system stability with $C = 20$, $\rho_1 = 0.01$. . . . 72
3.12 The comparison of stability region and decomposition for Heterogenous RTTs. 78
3.13 The average window size and normalized queue-length under different number of flows N. 81
3.14 The average window size and instantaneous queue-length $Q^N(t)$ with different initial window size $W(0)$ and under fixed $Q(0) = 5$. 82
3.15 The decomposition for heterogenous RTTs. 83

4.1 Examples of marking functions with aggressive marking scale of N^α. 86
4.2 Queue-length fluctuation using *ns*-2: (a) stable queue; (b) unstable queue with aggressive marking; (c) comparison of (a) and (b) over the same y-axis. 88
4.3 The CLT argument in [17] for $o(N^{0.5})$. 90
4.4 The CLT argument for our scheme in [51] for $o(N^\alpha), 0 < \alpha < 0.5$. 91
4.5 Homogeneous RTTs: Performance metrics with increasing number of flows (N) for fixed $\alpha = 1$. 93
4.6 Homogeneous RTTs: Performance metrics with increasing number of flows (N) for fixed $\alpha = 0.2$. 94
4.7 Heterogeneous RTT: Performance metrics with increasing number of flows, N for fixed $\alpha = 1$. 96
4.8 Heterogeneous RTTs: Performance metrics with increasing number of flows (N) for fixed $\alpha = 0.2$. 97
4.9 Webmice traffic: Performance metrics with increasing number of persistent flows, N for fixed $\alpha = 1$. 98
4.10 A mixture of short- and long-lived flows: Performance metrics for $\alpha = 0.2$ as the number of persistent flows (N) increases. 99
4.11 Simulation topology: multiple bottleneck links with heterogeneous RTTs. . 100
4.12 Multi-hop Topology: Performance metrics with increasing number of flows (N) for fixed $\alpha = 0.2$. 100

5.1 The TCP AIMD algorithm under a constant loss rate. 104
5.2 The system model and the interaction between call level and packet level. . 105
5.3 The algorithm for the base station to obtain $p(t)$. 112
5.4 The rate control scheme based on $p_i(t)$: stochastic packet dropping. 114
5.5 Stochastic admission control for a system with a queue. 115
5.6 $M/M/1$ model of packet arrival/serivce process. 117
5.7 The system model with high priority queue and the priority weight function with queue length: $g_1(x)$ and $g_2(x)$ are monotone nondecreasing functions with $g_1(x_0) > g_2(x_0), \ \forall x_0$ 119
5.8 The blocking probability comparison for with and without TCP AIMD control. 130
5.9 The throughput comparison for with and without TCP AIMD control. . . . 131
5.10 The link utilization comparison for with and without TCP AIMD control. . 132
5.11 The performance metrics of TCP-AIMD aware with the two-dimensional increase of both λ_1 and λ_2. 133
5.12 The blocking probability of Class 1 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively. 134
5.13 The blocking probability of Class 2 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively. 134
5.14 The throughput of Class 1 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively. 135
5.15 The throughput of Class 2 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively. 136
5.16 The link utilization for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively. 137
5.17 The performance comparison under different distributions of transmission time: Exponential distribution; Lognormal distribution; Uniform distribution. $p = 0.1667$. 138
5.18 The link utilization for different arrival rate: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$. 139
5.19 The link utilization for different arrival rate:(i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$. 140
5.20 The link utilization for different arrival rate:(i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$. 141
5.21 The link utilization for different arrival rate: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$. 141
5.22 The link utilization for different arrival rate: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$. 142

5.23 The performance comparison under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution. Dynamic $p(t) \in [0.0234, 0.1667]$. 143

List of Tables

2.1 Design of Reno, HSTCP, STCP and FAST [100] 19
2.2 Equilibrium Transmission Rate [100] 19
2.3 TCP Header Checksum Option [25]. 28
2.4 TCP Header Checksum ACK Option [25]. 28

3.1 Parameters for model comparison 54
3.2 Simulation Parameters 79

4.1 Performance Comparison of the queues in Figure 3.5. 88
4.2 AQM parameters for *ns*-2. 92

5.1 System parameters: call level and packet level. 106

Chapter 1

Introduction

Currently, most of today's Internet traffic are carried out by Transmission Control Protocol (TCP) [91], which accounts for 90% applications and 80% data. As the network evolves, many improvements and modifications are proposed to make TCP meet various requirements such as the environment of high-bandwidth delay product or over wireless links [166, 48, 94, 176, 10, 1, 179, 70, 65, 66]. TCP becomes the de-facto standard for reliable transmission protocol and this situation is expected to last in the foreseeable future. Therefore, it would be important and interesting to understand the underlying schemes regulating TCP behaviors.

In addition to TCP end-to-end algorithm itself, various Active Queue Management (AQM) algorithms are proposed [69, 18, 122, 36, 159, 60] to enhance TCP performance, where the one-bit congestion signal is generated by the intermediate routers to notify the upcoming congestions to the TCP sources. Specifically, TCP is an end-to-end congestion control protocol and it counts only on the end-to-end information, i.e, TCP sources get feedback only from the corresponding TCP destinations. AQM is introduced at the intermediate routers to help TCP control network congestion. Extensive research has been conducted in this area and the most popular and widely accepted one is Random Early Detection (RED) [69, 62, 49, 146, 39, 64, 204, 34, 131, 16, 4, 142, 182]. In addition, many other alternative AQM schemes are proposed as well: REM [18, 135, 19, 155, 198, 5, 133, 199, 35, 156], AVQ [122, 120, 116, 118, 128, 119, 117], PI [35, 34, 36], GREEN [195], BLUE [60, 58, 59, 53, 54, 57] and PURPLE [159], etc, all of which claim to improve the TCP performance in

one way or the other.

On the other hand, due to the enormous success of TCP in the wired network [166, 48, 94, 176, 65, 66], it is naturally applied to a wireless environment to obtain the reliable end-to-end transmission. Extensive research has been conducted to make the TCP over wireless feasible such as ATCP [132], M-TCP [32], W-TCP [165], ATP [181], I-TCP [24], and significant efforts have been made to improve the performance of TCP in a wireless link (either infrastructure or ad-hoc mode) [144, 80, 84, 73, 150].

1.1 Motivation

Recently, the researchers intent to explain the TCP the behavior with rigorous approach, rather than test their ideas and algorithms only with simulators, for example NS2. More importantly, theoretical analysis may shed the insight on understanding the underlying reasons of TCP behavior. This research focuses on theoretical analysis of TCP on both wired and wireless networks.

We should admit that the topic of performance analysis of TCP over wired and wireless is too abroad to be accommodated into one thesis work. The three problems investigated in this thesis work may seems to be in an ad-hoc fashion. However, as we shall see, there still exist some connections between the following three pieces of work. In fact, the first work and the second work treats a same problem from two different perspectives, i.e., classical control point of view and modern stochastic control point of view. The third piece of work tries to treat the TCP over wireless network issue from a new angle, i.e., from the resource allocation or admission control perspective. Instead of thinking of one particular flow, we look at the TCP flows sharing a access base station or access point, and studies the multiplexing behavior among all the flows. As we shall see, by intensionally controlling transmission rate of the TCP flows, we can improve the system performance from call admission control point view.

1.1.1 TCP Stability Analysis

Stability has been one of the important issues in designing and analyzing TCP/AQM (Active Queue Management) [138, 137, 156, 34, 35, 36, 134, 140, 160, 43, 120, 85, 81]. Ac-

cording to [138], an unstable TCP/AQM system may increase delay and jitter, which can be detrimental to some delay-jitter sensitive applications, and cause link under-utilization. The existing research on stability of TCP/AQM can be broadly classified into two groups: local stability and global stability.

The local stability is obtained by first linearizing the system around its equilibrium point. Then, by applying the Nyquist theorem to that linearized system, one can have a criterion for the local stability of the system [137, 36, 34, 35, 116]. However, due to the linearization involved, the local stability criterion is applicable only when the system operates near its equilibrium point. If the system operates on a region far away from its equilibrium point, then the nonlinear components may kick in and make such a linear approximation inaccurate. We define a *stability region* of a system by a set of initial points, from which the system eventually converges to its equilibrium point. Then, for a locally stable system, this stability region is usually small and limited around the equilibrium point.

On the other hand, the global stability requires that starting from *any* initial points (not limited to the points near the equilibrium point), the system eventually converges to the equilibrium point. Obviously, the global stability implies the local stability. In general, the global stability is obtained by properly constructing a Lyapunov function [163, 164, 12, 13, 126, 194, 45, 160, 193, 158]. In [45], the authors present a criterion for the global stability of a congestion controller with a single link and a single flow. Further, they show that the global stability criterion for a system with many identical flows sharing a bottleneck link converges to that of a single flow [43]. In other words, the condition for global stability with a single flow can be applied to the case of multiple flows as well.

Although the existing results regarding stability reveal valuable insight into the system design, they have their own limitations. For local stability, it is mostly applied to a system with TCP-Reno [138, 34, 36] instead of TCP-newReno, which is an improved version of TCP-Reno and widely deployed in practice. The major difference between TCP-NewReno and TCP-Reno is that for TCP-Reno, multiple packet losses or packet marks within one round trip time (RTT) results in multiple reductions of window sizes, while for TCP-newReno, only one reduction of window sizes is incurred (see [68] for more details). To the best of our knowledge, there is no result regarding the stability criterion of TCP-newReno under many flows.

Meanwhile, the existing local stability criterion is designed for a relatively small scale system (i.e., the number of flows in the system is on the order of tens), while we

consider a large system where the number of flows is hundreds or thousands level. According to [72], the core router is shared by more than $10,000$ flows, and we show that for such a large system, the local stability criterion is different from that of a relatively small system. For example, in a small system, the long RTT of a single flow will affect the stability criterion [138]. However, for a large system, we will show in this paper that the RTT for a single flow or a small number of flows will not make any impact on the stability of the system as a whole.

As to the global stability, the AQMs studied in the literature are mostly limited to rate-based marking schemes [43, 45], and not applicable to any other popular queue-based marking schemes such as RED [69]. Still, to the best of our knowledge, there has been no result about the global stability of a TCP-newReno/RED system with many flows. This is mainly due to the following two reasons. First, as the network evolves, the core router is shared by increasingly large number of flows. Since each flow has its own window size dynamics, it becomes more difficult to find an appropriate Lyapunov function for such a system with many flows. Second, the packet marking function for RED is a non-linear (usually piecewise-continuous) function, and this makes any theoretical analysis formidable.

All the above observations become our motivation and the techniques to address these problems are provided in Chapter 3.

1.1.2 TCP Performance under a Large Scale

Up to now, the most popular approaches for TCP/AQM congestion control have been two-fold: (i) a global optimization problem that maximizes the total system utility where a certain set of fairness objectives can be met by appropriately choosing the cost (or utility) functions, or (ii) a so-called "fluid modeling" approach, in which the end-user chooses certain data rate and injects its data precisely at this rate like *fluid*. These two approaches are closely related. In the first approach, the congestion control is viewed as a distributed, iterative solution to the global optimization problem [105, 136, 123], where one iteration is performed per each round-trip-time (RTT). On the other hand, the focus in the fluid-modeling approach is to describe the interaction between the source algorithm and the network algorithm (AQM) over a series of RTTs, via a set of difference/differential equations. Numerous issues have so far been addressed in these areas, including global stability of the congestion control algorithm [46, 11, 200, 161], local stability (in the sense

of Lyapunov) [148, 86, 36, 157, 190, 139, 104, 121, 124], the impact of feedback delay and random noise [101, 178, 203], and the asymptotic behavior of the system when the number of users is large [87, 178, 44, 186, 183].

However, one thing is common in all of the above. The stability analysis and all the design guidelines have been made by focusing only on the dynamics over a series of RTTs, where all the packet-level behavior *within each RTT* are "averaged out" and lumped into a single quantity such as the window size or the transmission rate of each user per RTT.[1] Such an abstraction has been extremely useful for the analysis and control of large networks. For example, when there are N flows with capacity NC and linear scaling ($O(N)$) is employed for the buffer size and AQM (e.g., all the buffer thresholds for packet marking are $O(N)$), it has been shown that the system dynamics can be described by the *normalized* version of the system [87, 186] via the law-of-large-numbers type of arguments. Also, by applying standard techniques from traditional control theory, several stability criteria were obtained in terms of parameters of the normalized system [46, 178, 139]. The main underlying idea behind these results would be the notion of 'first-order' principle; all the first-order behavior (or mean behavior) of the system can be captured through the 'normalized' system, and any other second-order dynamics over finer time scales will not impact the first-order rule provided that the averaged behavior remains the same.

However, such an approach based on the mean behavior of the system (or on averaged quantities) can be fallible and may pose severe limits on our ability to obtain optimal design guidelines for large-bandwidth systems. Our observation is that, when the system is large with many flows, there will be a large number of packets coming from different sources per RTT and their dynamics in time (i.e., random arrival instants) often become too rich to be captured by such a single (averaged) quantity. For instance, if we look back on the issue of buffer sizing for large systems and demonstrate that TCP/AQM systems work well with all the desirable performance even under a much smaller buffer size and corresponding AQM scale, which as we will show would make the system linearly unstable and thus has been 'prohibited'. Specifically, we present an example of a TCP/AQM system such that, as the size of the system increases, the performance metrics approach to the optimal case (e.g., 100% throughput, zero loss ratio, etc.) while the system is becoming more unstable (in

[1] An exception is in [47] where packet arrivals within an RTT is explicitly modeled. However, only two cases ($\alpha = 0$ or $\alpha = 1$) were considered, and their approaches are not applicable to any other scales of our interest.

the sense of Lyapunov), implying possible performance degradation. While our extensive simulation results to shows that the system always performs well in the sense that the link utilization goes to 1, the queueing delay goes to zero and the loss ratio decreases to zero as the system size N increases.

We address this counter-intuitive discrepancy in Chapter 4.

1.1.3 TCP Performance over Wireless Networks

Due to the enormous success of TCP in the wired network [166, 48, 94, 176, 65, 66], it is naturally applied to a wireless environment to obtain the reliable end-to-end transmission. Extensive research has been conducted to make the TCP over wireless feasible such as ATCP [132], M-TCP [32], W-TCP [165], ATP [181], I-TCP [24], and significant efforts have been made to improve the performance of TCP in a wireless link (either infrastructure or ad-hoc mode) [144, 80, 84, 73, 150]. A key feature of TCP congestion control is so called Additive-Increase-Multiple-Decrease (AIMD) algorithm. If there is no packet loss within the current round trip time (RTT), the window size increases by one during the next RTT; Otherwise, the window size decreases by half. Even though the existing literature regarding TCP over wireless may differ in different aspects, they share a same TCP essential congestion control algorithm, i.e, AIMD that determines the packet-level TCP dynamics.

In this thesis, we show that such AIMD algorithm plays an important role in affecting the future packet-switched wireless networks such 3G and 4G. Since the packet-switched technology yields more efficient utilization of the scared wireless resource than that of circuit-switched networks, the future 3G and 4G wireless network will be packet-based networks [90, 31, 88, 2] and this is especially true for 4G system [2]. In a packet-switched wireless networks, two level performance metrics come into the picture of Call Admission Control (CAC): (i) call level parameters: call blocking probability and handoff dropping probability; (ii) packet level parameters: packet loss probability and link utilization.

Traditionally, CAC schemes deal with call level performance metrics only: call blocking probability and handoff dropping probability [191, 192, 9, 129, 98, 7, 52, 89]. In [130], the admission control algorithm for integrated voice and data traffic is proposed and analyzed. The idea is based on completed sharing method, where the channels within the cell are shared by all the traffic, rather than reserved for a specific class. The au-

[2] We note that 3G is circuit and packet switched networks, but 4G is pure packet switched networks.

thors demonstrate that the proposed scheme achieves both service guarantee and service differentiation for voice and data with better bandwidth utilization. In [9], user mobility is considered by the CAC scheme with arbitrary call-arrival rate, which requires the global information on the call arrival process and user mobilities. The proposed scheme is analyzed in terms of a more realistic network topology and their results show that the CAC algorithm achieves high throughput with guaranteed Quality of Service (QoS) and call blocking probabilities. In [129], the mathematical model for on/off traffic model is obtained for a cellular networks provisioning voice and best-effort data service. The traffic is characterized by a three-dimensional birth-death model that effectively captures the complicated interaction between on/off voice traffic and best-effort data traffic under a complete sharing situation. The close form for the performance metrics are derived and the minimum amount of resources to guarantee the required QoS is obtained.

All these traditional CAC schemes are limited to the call level performance. Recently, packet level performance is taken into account by [75, 202, 201]. The authors propose an efficient CAC scheme for heterogeneous services in wireless ATM networks in [202, 201]. The packet level constraints such as delay and jitter are incorporated into the scheme to guarantee the predefined QoS. A higher priority is assigned to handoff calls to improve the handoff dropping probability by reserving certain amount of bandwidth for potential handoff calls and such guard-channel like idea is widely applied to the CAC field as in [52, 89]. The simulation results indicate that the proposed CAC scheme can achieve both packet level and call level performance in terms of link utilization and handoff dropping probability. Further in [75], the authors claim that they can even totally remove the handoff dropping event, which is quite desirable from the CAC point of view. The idea is that as long as a handoff call arrives, the system admits it at the expense of increasing load of the packet level, because each call will inject some amount of packets into the network. As the number of calls in the system increases, the load of the packet level increases and beyond certain threshold, the packet loss occurs indicating that the system is near to the overflow status. The system will block the new call to control the packet loss probability within the predefined level. Ultimately, the zero handoff dropping probability is achieved at the expense of new call blocking probability, and this is accomplished by combining both packet level and call level into account.

However, none of these schemes considers the impact of TCP applications on the packet level performance. Due to the trend of incorporating TCP over the future packet-

switched wireless network, it is imperative to investigate the effect of TCP-AIMD algorithm on CAC. Unlike the existing work on TCP over wireless [132, 32, 165, 181], in which only a single or several TCP flows are considered and the objective is to improve the throughput for a single TCP flow, we consider a different and more realistic scenario, where many TCP calls arrive at the system, ask for the resources, leave the system and release the resource after the service is finished. Our focus is to investigate the aggregate packet-level behavior of those TCP calls determined by the AIMD congestion control algorithm.

Intuitively, the AIMD algorithm makes the TCP call keep increasing the rate injecting to the network until there is a packet loss due to the limited capacity. There are two implications by such AIMD algorithm. First, TCP call based AIMD algorithm will exhaust all the available system capacity and thus the system will block/reject any call requests coming in. In other words, the call level performance (call blocking probability and handoff dropping probability) is degraded in such a situation. Second, once there is a packet loss, the TCP call will decrease the sending rate by half, which could be far below the system capacity, and hence lead to poor packet level performance (i.e., low link utilization).

Techniques to address this problem are provided in Chapter 5.

1.2 Contribution

The contributions of this dissertation are three fold:

- First, we analyze the stability of TCP over a large system under a linear buffer size. We study the local and global stability of TCP-newReno/RED under many flows regime. The existing results of the local stability are mostly for TCP-Reno, not for newReno. These results are obtained based on a small scale model with a few number of flows and thus cannot be blindly applied to a large system with many flows. Moreover, traditional approaches for the global stability based on Lyapunov functions is not suitable for a system with a large amount of flows due to its complexity. Motivated by this, we present a normalized discrete-time model to capture the essential dynamics of TCP-newReno/RED with many flows and obtain its local stability criterion. The normalized model allows us to proceed numerical iterations to analyze the global stability in an efficient manner. Our results show that by properly choosing some 'free' parameters, we can always ensure that a locally stable TCP-newReno/RED

system is in fact globally stable. Our results become more accurate as the number of flows increases. Finally, we extend our normalized model to the case of heterogeneous RTTs.

- Second, we study the TCP performance under the aggressive scale, i.e., the buffer size is on the order of $(N^\alpha), 0 < \alpha < 1$. Our results indicate that even though the system is not stable under such configuration, the key performance measures still remain good. Specifically, we consider a TCP/AQM system with large link capacity shared by many flows. Traditional approaches via fluid-based modeling require that the buffer size be chosen in proportion to the number of flows (N) for system stability, while recent research outcomes suggest $O(\sqrt{N})$ for buffer sizing by exploiting the independence among different flows when the number of flows (N) is large. However, they all invariably fail to predict the true system performance if the buffer size at the bottleneck link and the corresponding AQM scheme are scaled as $O(N^\alpha)$ with $\alpha < 0.5$. We show that, under the scale of $O(N^\alpha)$ with $\alpha < 0.5$, the system always performs well in the sense that the link utilization goes to 1 and the loss ratio decreases to zero as the system size N increases. We verify our results using extensive ns-2 simulations under different AQM schemes and show that the buffer size can be chosen much smaller than $O(\sqrt{N})$ (as recently proposed) without sacrificing all the key performance metrics. We therefore believe that, under our scale for the buffer size and AQMs for large-bandwidth networks, fluid-based approaches can be somewhat misleading and the packet-level dynamics must be taken into account.

- Third, A TCP-AIMD aware call admission control scheme over wireless networks. Traditional Call Admission Control (CAC) schemes only consider call level performance and are believed to be sufficient for the circuit-switched wireless network. Since the future wireless network will become packet-switched, the packet level performance cannot be ignored. This is especially true when the TCP-type of applications are running over such packet-switched wireless networks, because TCP Additive-Increase-Multiple-Decrease (AIMD) congestion control algorithm will exhaust all the available resource until packet loss occurs. In order to regulate the TCP applications to friendly coexist with other types of services, we propose a TCP-AIMD aware CAC scheme based on a two level CAC framework, which takes both call level and packet level into account. The packet level dynamics is captured by four models: (i) $M/M/1$ model;

(ii) a priority-queue model; (iii) effective bandwidth model; (iv) Gaussian approximation model, and call level feature is characterized by Markov process and embedded Markov process for different cases, respectively. The interaction between two levels is revealed through Quality of Service (QoS) guaranteed capacity. The performance of the proposed scheme is analyzed and extensive simulation are proceeded under different scenarios. Our results show that the proposed scheme can significantly improve the system performance of both call level and packet level in terms of call blocking probability, call level throughput (call/min) and link utilization.

We note that the first and second pieces of work are more complementary, than contradict to each other. The main reason is that for the first work, we approach the problem from the traditional control point of view. We try to make the system stable in the sense of 'Lyapunov' stability definition. The trajectory of the system dynamics should eventually go to the equilibrium point. Many researchers believe that the stability is a desirable property for the target system [138, 137, 34, 35, 36, 134, 140, 160, 43, 120, 81], where the system always preserve 100% link utilization (here, we refer to the outgoing links, not incoming links) and zero packet loss probability. On the other hand, the second works treat the system from the modern stochastic control point of view, where the system is stable is in the sense that the system will eventually go to steady state. In other words, the system has the stationary state distribution. We know that if the system has a steady state distribution, its trajectory may not converge to some fixed equilibrium point. Since the modern stochastic control theory mainly deal with the system with randomness, the system dynamics cannot be expressed by a deterministic model and hence the traditional definition of stability (Lyapunov sense) cannot be applied to those systems, and naturally, the steady state distribution becomes a good candidate that allows us to investigate the system performance. In the second work, we obtain the asymptomatic results that even though the system is unstable, the link utilization goes to 100% as the number of flows goes to infinity (however, it could go to 100% very quick, or very slow), and the queuing delay goes to zero (but not zero).

1.3 Outline

The rest of the dissertation is organized as follows: Chapter 2 describes the background of TCP. Chapter 3 analyzes the stability of TCP over a large wired networks with many flows, where the linear buffer size is applied. Chapter 4 presents the analysis of TCP performance over a large network, where the aggressive buffer size is considered. Chapter 5 presents the study of TCP over wireless networks from a resource allocation point of view. Chapter 6 concludes the dissertation and discusses some future work.

Chapter 2

Background of TCP

In this chapter, we summarize some background and related literature of TCP for the convenience of readers. We note that most of the figures presented in this chapter are courtesy from the corresponding literature as cited. The interested audiences are referred to those papers for the detailed information.

2.1 TCP over Wired Networks

The rapid development of high-speed networks makes the applications such as realtime environment monitoring, scientific collaboration and telemedince possible nowadays. In general, these applications have two features: high-bandwidth and real time.

Since TCP becomes the standard of data transportation protocol, it is naturally deployed by these applications. However, when TCP is applied to high speed network (say, 10 Gbps) with large delay (say, RTT is around 100 ms), the network resource suffer severely under utilization [65, 70, 106, 100]. For example, the bandwidth is 10 Gbps and the packet size is 1500 bytes. It takes 83,333/2 RTTs (RTT=100ms) for TCP to increase its window size from half utilization to full utilization of the total bandwidth. That is, the time for TCP slow start up is almost 1 hour [197]. Moreover, within 1 hour TCP cannot bear any loss of packets, otherwise the window size of TCP will be cut into half. This requires the network successfully transmit over 2,600,000,000 packets with even one loss, which is

impossible from the theoretical results [65, 70].

Fine-Tuning TCP

One possible solution is called fine-tuning TCP parameters [166, 48, 94, 176, 10, 1, 179]. They approach this problem by adjusting receive window size and network interface buffers etc. For instance, in [179], the packet size is suggested to be increased up to 8 Kbytes and multiple TCP connection is used. Moreover, for each RTT, the window size has a larger increment instead of 1. The good thing for these methods are that the changes to the standard TCP are small, while yield better performance compared to standard TCP.

In [48], Dunigan et al. proposed a method called "TCP-tuning Daemon". TCP instrumentation data from the Unix kernel is deployed to adjust TCP parameters for specified flows along the network paths. This modifications is transparent to the upper level of application, and the user does not need to understand the underlying network conditions: specifically TCP parameters. The improved performance of tuning technique is depicted in Figure 2.1, which shows that the throughput can be increased by this tuning technique.

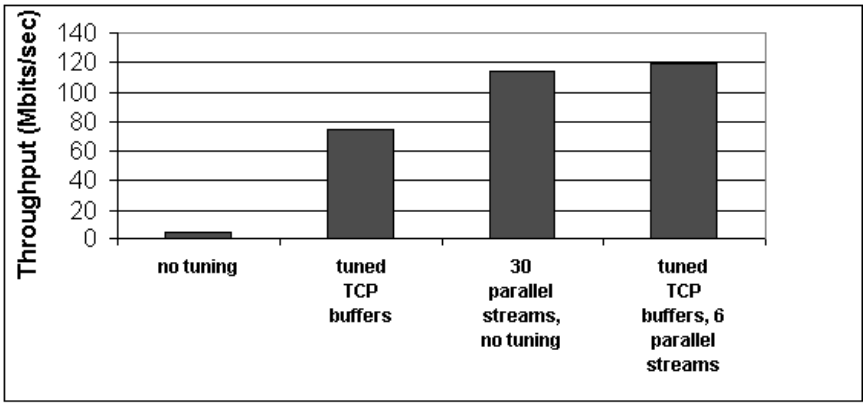


Figure 2.1: Performance: Tuning TCP buffers and parallel streams [48].

Semke et al. suggested TCP buffer Tunning to improve the TCP performance over high bandwidth link in [176]. The main idea is to adjust socket buffer to maximize the

transmission rate. They make the modification to the existing BSD-Socket interface and TCP stack. A significant performance improvement is claimed by the authors. Another benefit is the memory usage reduction, which make the host to provide at least twice simultaneous connections as before. Figure 2.2 and 2.3 show the performance comparison to the other two schemes: default setting by NetBSD V1.2 and tunning by hand. We can observe that the proposed scheme has the best throughput with better memory usage.

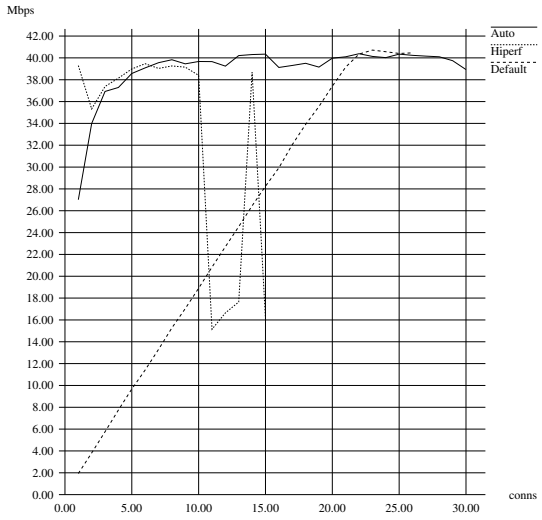


Figure 2.2: Performance Comparison: Aggregate Bandwidth. Auto: the proposed scheme; Hiperf: Tuning by Hand; Default: Default setting by NetBSD [176].

In order to achieve better solution to this problem, many new protocols are proposed. These works have two important features: (I) TCP friendliness (II) scalability.

Scalable-TCP: STCP

In [106], T. Kelly proposed a new TCP variation called scalable TCP. The basic idea behind STCP is that unlike traditional TCP, whose loss recovery time is proportional to both congestion window size and RTT, the loss recovery time for STCP only depends on RTT. The algorithm of STCP can be generalized as [106]

$$cwnd = cwnd + a \tag{2.1}$$

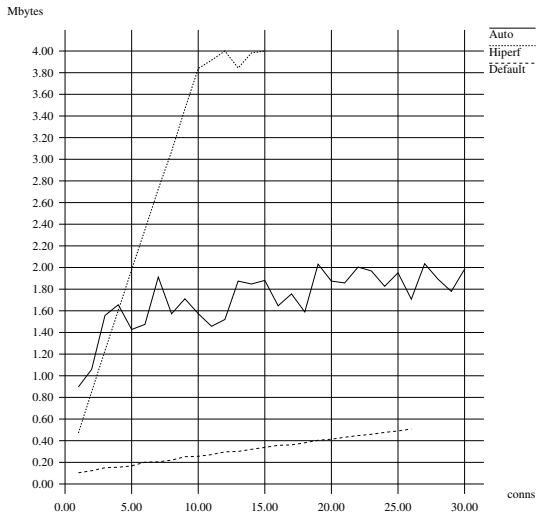


Figure 2.3: Performance Comparison: Peak usage of memory. Auto: the proposed scheme; Hiperf: Tuning by Hand; Default: Default setting by NetBSD [176].

for each acknowledgement without a loss packet, where $0 < a < 1$. And for each loss occurs

$$cwnd = cwnd * (1 - b) \tag{2.2}$$

where $0 < b < 1$. The values of a and b depend on implementation and deployment such as legacy traffic, bandwidth and flow rate variance.

Compared to the standard TCP, we can observe that standard TCP is a special case of STCP with $a = \frac{1}{cwnd}$ and $b = \frac{1}{2}$. This is demonstrated in Figure 2.4. We can see that Packet loss recovery times for a traditional TCP connection are proportional to the connections window size and round trip time. A Scalable TCP connection has packet loss recovery times that are proportional to the connections round trip time only; this invariance to link sizes allows Scalable TCP to outperform traditional TCP in high speed wide area networks.

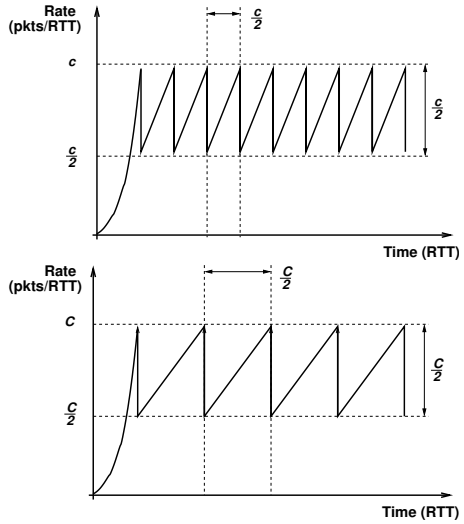


Figure 1: Traditional TCP scaling properties.

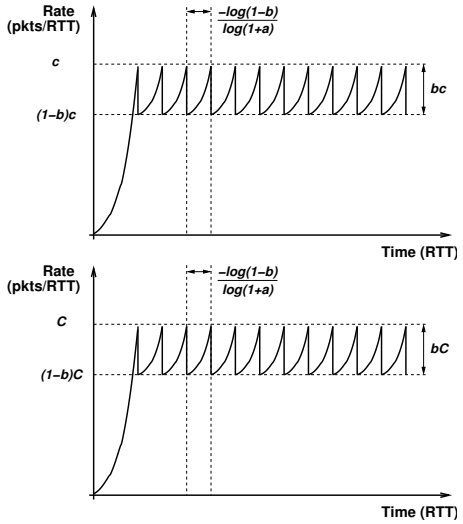


Figure 2: Scalable TCP scaling properties.

Figure 2.4: Illustration: Traditional TCP and STCP [106].

High Speed TCP: HSTCP

Floyd, et al. modified TCP as HSTCP in [70, 65, 66]. The objective of HSTCP is to design a congestion control algorithm which obtains a higher sending rate at a given loss rate, while keep the property of TCP-friendliness. The algorithm is also a kind of additional increase and multiplicative decrease (AIMD) with different addition and multiplication.

The objective of HSTCP is to achieve transmission rate of 10 Gbps under realistic packet drop rate environment. Since Floyd has not solidify the design detail, we don't know how it works so far. It has the following benefits as pointed out in the draft.

- Maintain high transmission rate with reasonable loss rate.
- Reach steady transmission rate pretty fast during slow-start.
- Fast recovery during time-out or congestion loss.
- Incremental Deployment: compatible with existing TCP. That is, it should be no harmful to the existing TCP. When competing resource, it is fairness for existing

TCP.

- No router involvement.

Figure 2.5 plots the fairness of HSTCP over regular TCP. As the loss probability increases, the HSTCP tends to have the fair throughput compare to regular TCP.

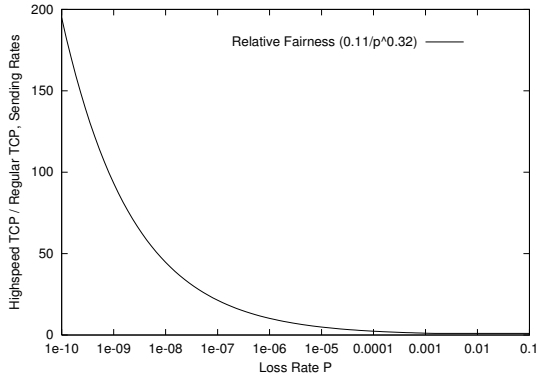


Figure 2.5: Fairness of HSTCP over regular TCP [70].

BIC-TCP: BTCP

Moreover, Xu, et al. discussed the binary increase congestion control for high-speed networks in [197]. The authors claims that BIC-TCP achieve the following properties:

- Scalability: HSTCP can obtain 10 Gbps bandwidth with loss rate 1e-7, while BIC-TCP would achieve 10 Gbps with 3.5e-8 loss rate.
- RTT fairness: RTT fairness is defined as the RTT ratio of two competing flows. The order of RTT unfairness is about $\frac{1}{\sqrt{RTT}}$.
- TCP friendliness: For all window size, BIC-TCP can acquire bounded TCP fairness. With high loss rate, BIC-TCP friendliness is the same order of STCP's

- Fairness and convergence: BIC-TCP has better sending rate fairness compared to both HSTCP and STCP. And the convergence of BIC-TCP is faster as well.

Comparison of TCP fairness between BTCP and other TCP variation is depicted in Figure 2.6.

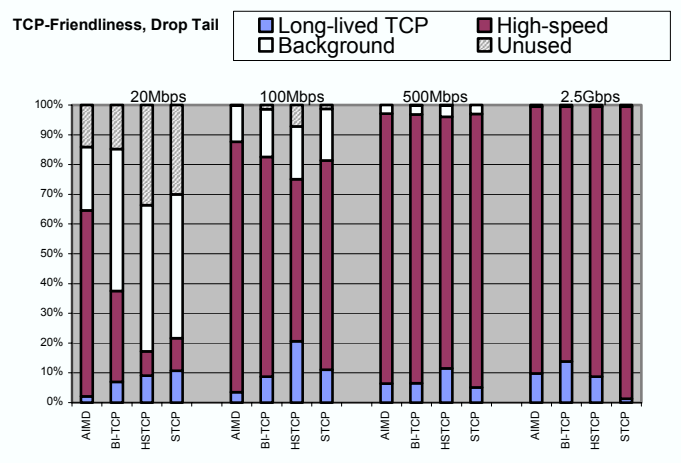


Figure 2.6: TCP Fairness [197].

Fast TCP: FTCP

Jin, et al. proposed a Fast TCP, which deploys a delay-based congestion control [100, 99]. In fact, using delay to estimate congestion level was already proposed by [95, 143]. The advantage of delay-based over loss-based method is not significant for low speed networks, while as to high speed networks, delay-based may reach a big improvement.

FTCP combines delay-based approach and loss-based method. There are two merits to deploy queueing delay as a congestion measure.

- In reality, queueing delay can be obtained more accurately than packet loss probability. We know that packet losses are around the order of 10^{-8} for high speed networks.

Moreover, the queueing sampling gives much finer information than loss sampling.

- The dynamics of queues has the right scaling with regard to network capacity, which will help TCP stability as the network capacity scales up.

Flow Level Design:

Table 2.1: Design of Reno, HSTCP, STCP and FAST [100]

	$\kappa_i(\omega_i, T_i)$	$\mu_i(\omega_i, T_i)$	p_i
Reno	$\frac{1}{T_i}$	$\frac{1.5}{w_i^2}$	loss probability
HSTCP	$\frac{0.16b(\omega_i)\omega_i^{0.8}}{(2-b(\omega_i))T_i}$	$\frac{0.08}{\omega_i^{1.20}}$	loss probability
STCP	$\frac{a\omega_i}{T_i}$	$\frac{\rho}{\omega}$	loss probability
FAST	$\gamma\alpha_i$	$\frac{\alpha_i}{x_i}$	queueing delay

where T_i is the round trip time (RTT), ω_i is the window size, p_i is the congestion measure, $x_i = \omega_i/T_i$. And $a, b(\omega_i), \rho, \gamma, \alpha_i$ are protocol parameters.

The equilibrium transmission rate, x_i, is given by

Table 2.2: Equilibrium Transmission Rate [100]

	Transmission rate
Reno	$x_i = \frac{1}{T_i}\frac{\alpha_i}{p_i^{0.50}}$
HSTCP	$x_i = \frac{1}{T_i}\frac{\alpha_i}{p_i^{0.84}}$
STCP	$x_i = \frac{1}{T_i}\frac{\alpha_i}{p_i}$
FAST	$x_i = \frac{\alpha_i}{p_i}$

XCP

Katabi, et al. designed eXplicit Control Protocol (XCP) in [102]. Unlike explicit congestion control (ECN), which uses only one bit to indicate the network congestion, XCP has the senders explicitly know the congestion degree. To control TCP fairness, the authors

introduce a new concept called "decoupling of utilization control from fairness control". Based on utilization control, XCP reach stability under high speed network with large delay, which is believed to be very important to the congestion control protocol. Moreover, XCP has almost all the desirable properties: efficiency, fairness, scalability.

The additional benefits of deploying XCP are the following:

- By deploying XCP, drops due to congestion becomes unusual, because before a drop occurs, an explicit notification is sent back to the sender to reduce the window size.
- XCP is easier to detect attacks and separate the unresponsive flows, which facilitate the network security.
- Gradual deployment: Since XCP is a TCP-friendly congestion control protocol, it can be fairly co-exist with current TCP and its variations.
- XCP enhances network security. For example, XCP can easily detect and isolate the unresponsive sources due to two characteristics: (i) operation with near-zero drops, (ii) explicit feedback.
- QoS: Since XCP decouple of utilization from fairness, it makes Differv service easier to design and implement. In detail, by using efficiency controller, XCP guarantees high utilization and small queues. Therefore, Diffserv is just to apply the aggregate feedback to the individual flows to make them converge to the desired rates.

The design rationales behind XCP are from several aspects.

Extensive simulation results indicate that XCP outperforms TCP over conventional AQM (Active Queue Management) such as RED, REM, AVQ, CSFQ (Core Stateless Fair Queue).

TCP-SABUL

Gu, et al. proposed SABUL in [78], which combines UDP and TCP. There are two types of connections in SABUL: (i) control connection, which is transmitted by TCP. (ii) data connection, which is delivered by UDP.

SABUL tries to make a balance between TCP-friendliness and higher transmission rate, which is shown in Figure 2.7.

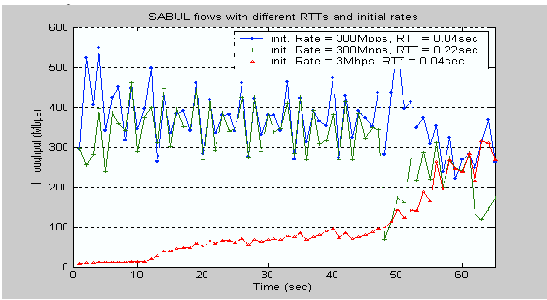


Figure 2.7: Fairness of TCP SABUL [78].

TCP-Westwood

TCP Westwood [144] is designed to improve TCP performance in both wired and wireless links. Instead of using packet loss, TCP Westwood deploys end-to-end bandwidth as an indicator for network congestion control. Unlike packet loss indicator, end-to-end bandwidth can tell the difference between the packet loss due to network congestion and the packet loss caused by wireless channel fading.

In general, this difference is important for us to reduce the packet loss. Specifically, if the packet loss is the result of network congestion, the TCP should decrease the window size to reduce the work load to the networks. However, if the packet loss is caused by wireless channel fading, the TCP should send more packets to recover the loss ones. We can see that two causes of packet loss may result in the very contrary result. Therefore to apply TCP into wireless link, we should enable TCP to discriminate this difference.

An important feature of TCP Westwood is that it fully complies with TCP design principle: only using end-to-end control and no intermediate router involvement necessary. This means TCP Westwood will not interact with Active Queue Management (AQM) router. It will reduce the complexity of both TCP end and intermediate server. Also it may decrease

the overhead to the network design and implementation. In reality, simple approach always gives rise to the robustness.

In detail, TCP Westwood decides a slow start threshold (ssthresh) and a congestion window (Cwnd), which leads to a effective bandwidth corresponding to the congestion. This scheme is called "fast recovery". Compare with previous TCP version, which simply reduces the Cwnd to half, TCP Westwood can effectively cope with wireless link. Shortly, end-to-end bandwidth estimation is achieved by three steps: (i) ACK-based measurement (ii) Filtering ACK reception rate (iii) detect on the delay and cumulative ACKs. Finally, it is also claimed that TCP Westwood has good throughput and TCP friendly as well, which is displayed in Figure 2.8 and 2.9.

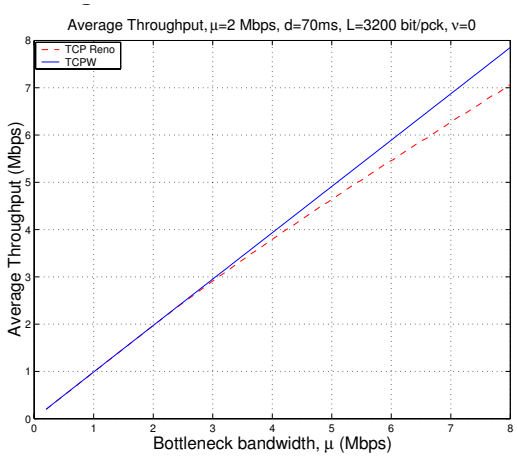


Figure 2.8: TCPWestwood and TCP-Reno: Throughput vs. bottleneck bandwidth [144].

TCP-Planet

A reliable Transport Protocol for interplanetary internet by Ozgur, Akan in [8], which is suitable to apply to the extremely large propagation delays, very high link errors, asymmetrical bandwidth, and blackouts.

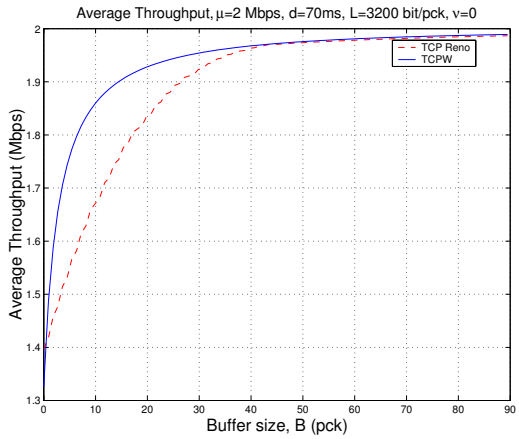


Figure 2.9: TCPWestwood and TCP-Reno: Throughput vs. bottleneck buffer size [144].

TCP-Planet also deploys AIMD congestion control algorithm. While AIMD here is tuned to avoid throughput degradation. Further, TCP-Planet modifies the slow start algorithm, which is not suitable to inter-planet transmission, to a "Initial State Algorithm", which would capture the available resource in a fast manner. Moreover, a new congestion control algorithm is develop for inter-planet transmission. They introduce "blackout" state into the new protocol to reduce the effect of "blackout". Finally, a delayed selective acknowledgement (DSACK) is proposed to address the bandwidth asymmetry problem.

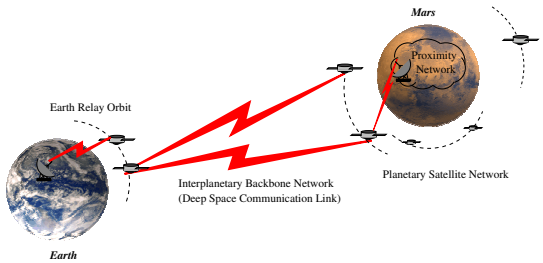


Figure 2.10: Interplanetary Internet [8].

Figure 2.11 indicates the state transition among the protocol.

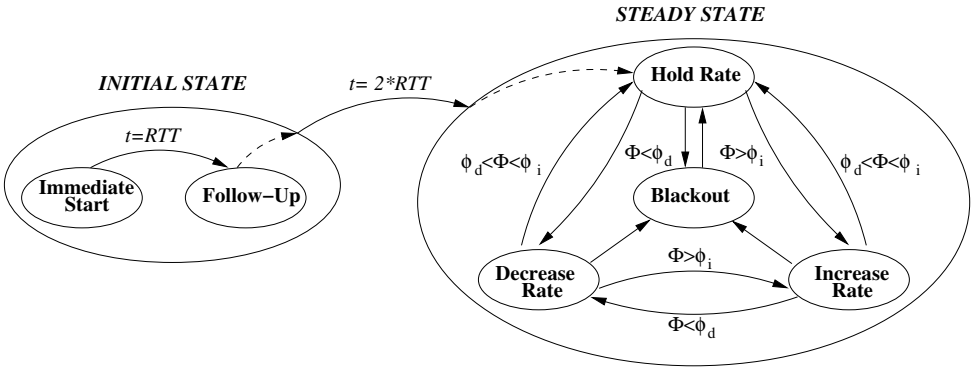


Figure 2.11: State Transition Diagram [8].

2.2 TCP over Wireless Networks

With the booming development of wireless networks, TCP over wireless becomes a natural result due to the demand of application. By introducing wireless to TCP, TCP should consider 3 major changes related with loss, because the main objective of TCP is reliable transmission.

1. Congestion loss: Wireless links introduce more traffic to the existing wired backbone. And congestion loss may happen at wireless router, wireless gateway in cellular networks

2. Handoff loss: This type of loss is unique to TCP over cellular networks. In other environment, either wired or wireless ad-hoc network has no such kind of loss. Since handoff is the procedure that a mobile terminal transfer from one access point to another access point, this process may introduce potential packet loss.

3. Error loss: Due to multi-path fading property of wireless link, TCP over wireless suffers more error loss than TCP over wired link.

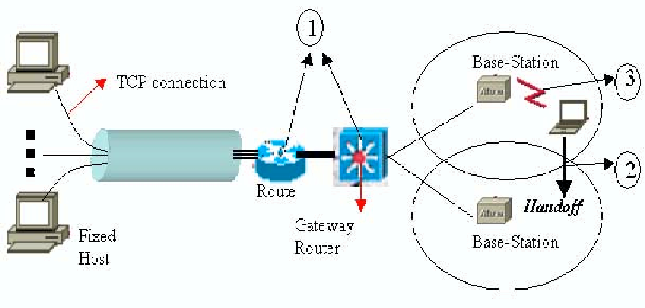


Figure 2.12: Three Types of Loss in Cellular Networks.

TCP over Cellular Networks

In [83], Hu, et al. studied the TCP performance in the case of cellular networks. Specifically, the authors proposed a scheme to improve TCP through during handoff. The merit of suggested scheme is that it is kind of fine-tuning to existing TCP, which may reduce the cost in real implementation.

Moreover, a simple Finite State Markov Chain (FSMC) model is suggested to analyze error loss over wireless channel. They use FSMC to study the Rayleigh fading channel. And, coding scheme is also taken into account by FSMC. Finally, they verify their results by NS simulator.

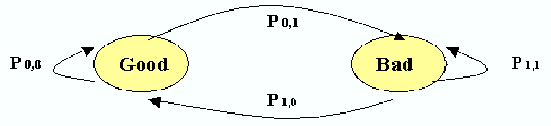


Figure 2.13: Two State Markov Chain for Loss Channel [83].

Another important issue, TCP fairness over cellular networks, is explored in [141].

TCP over Ad-hoc, Multi-hop Network

Again, fairness is an important issue for TCP field. Many research has been done to explore the fairness in wired networks. Bottigliengo, et al. studied TCP fairness for 801.11b WLANs in [29].

Traditionally, QoS in wired networks concerns about (i) traffic burstiness (ii) unpredictable nature of the number of users (iii) random access protocols. In addition to these obstacles, WLANs impose more challenges including poor channel qualify, interference from hidden terminals, etc.

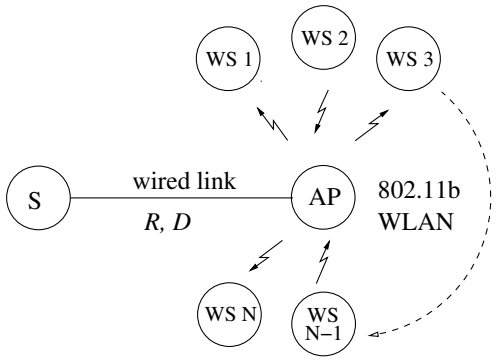


Figure 2.14: Network Scenario for 802.11b WLAN [29].

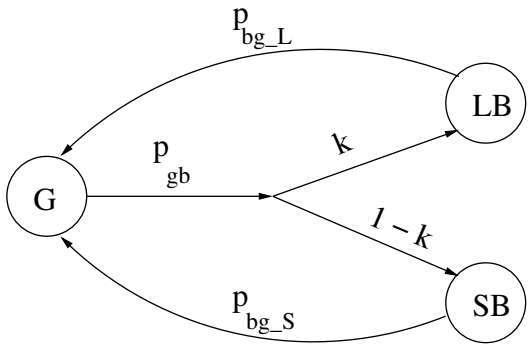


Figure 2.15: Three State Markov Chain for Error [29].

In order to achieve fairness, the authors propose an LLC-layer algorithm to guaranteeing fair access to the medium to every user by awarding longer transmission opportunities to short channel failures. Such mechanism can protect short-lived TCP flows.

TCP API for Wireless

In the real world, the implementation of wireless API for TCP is discussed in [20]. In wireless environment, the bandwidth and RTT have a large scale variation, which is unlike the wired link, because the distance between the mobile terminal and the base station is changed over time and the interference received by mobile terminal variation from time to time as well. Fig 1 and 2 in [20] clearly demonstrate such variations.

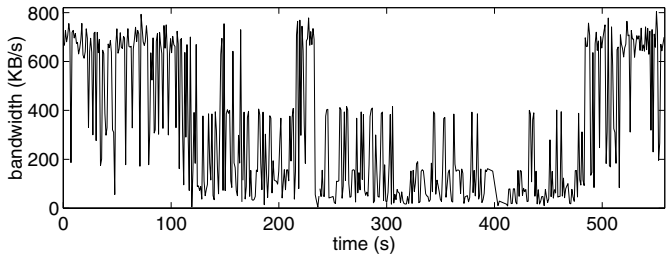


Figure 2.16: An Example of Bandwidth Variation over time [20].

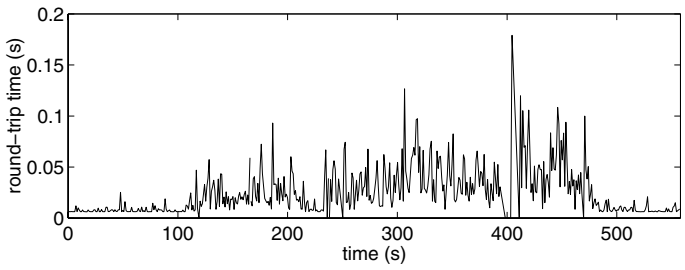


Figure 2.17: An Example of Round Trip Time Variation over time [20].

Atkin, et al., propose an network-aware interface (NAI) for adaptive applications over wireless networks. The API can help the upper level application accurately match the operation to the available bandwidth, i.e, cope with the burst data.

The idea to serve adaptive application is also discussed in [71, 151, 111].

TCP Wireless Header

Another TCP variation happens at TCP header. In [25], Balan, et al. presented a modification to TCP header. This revise enhances TCP to cope with lossy environment. In a short, they deployed two TCP options: (i) TCP header checksum option, (ii) TCP header checksum ACK option, which are given as follows:

Table 2.3: TCP Header Checksum Option [25].

Kind=14	Length=4	1's complement checksum of TCP header and pseudo-IP header

Table 2.4: TCP Header Checksum ACK Option [25].

Kind=15	Length=6	32-bit sequence number of corrupted segment to resend

2.3 Other Related Work of TCP

With the wide deployment of TCP, accurate prediction of TCP parameters becomes more and more important in order to guarantee different level of customers and business services.

TCP Prediction by Sampling

Application sampling is a good way to monitor TCP performance, since it is simple to be deployed. In [76], Goyal, et al. investigate the method to sample the network to predict network parameters such as RTT, loss rate and throughput.

Two sampling methods are proposed in [76].

- (I) Probing: This approach probes the network with end-to-end pings, for instance. Each ping can provide an RTT sample, which can be used to estimate retransmission timeout, T_0, along with clock granularity, G.

  $$T_0 = srtt + max\{G, 4 \times rttvar\}$$

  where $srtt$ is smoothed RTT, and $rttvar$ is the standard deviation of RTT samples.

  The pros of probing is that they provide the information of complete networks. Therefore, by probing, we can estimate the end-to-end packet loss probabilities and round-trip times. The cons of probing are that

  1. Probing packets do not work the same as the actual TCP flows.
  2. Packets loss by probing maybe chang significantly with different number of packets in one round probe, the inter-probe delay, the frequency of probing, and the probing packet size.
  3. The convergence of packet loss probabilities is slow.
  4. Due to IP routing policy, probing packets may not follow the same path as the packets in real transmission.

- (II) Polling: This method is to periodically send query message to the SNMP or MIB maintained by the routers to retrieve interested information such as the number of transmitted packets, the number of lost packets and the output queue length. With those information, we can estimate packet loss rate and queuing delay at router. The pros of MIB is that it collects statistics data by itself, rather than introduce a large amount of probe packets into the networks. Of course, the number of polling requests will determine the overhead of MIB polling. The cons of MIB's polling are that the information provided by one MIB is limited to the specific router. To obtain the end-to-end statistics information requires polling to multi MIBs along the path.

One thing to avoid by such sampling approach is over-sampling, which may incur unnecessary overhead to the networks. In general, session level and network level information is used to predict the throughput of TCP flows. The session level parameters consist of packet size, window size, retransmission timer and ACK. The network level parameters include RTT and loss rate.

TCP Prediction by Passive Measurements

The idea of passive measurement is new and interesting [96]. Instead of measuring the TCP connection at end point, the authors keep track of TCP parameters in the middle way of an end-to-end path.
The benefits of estimating RTT and CWND are that

1. They help us know the factors affecting TCP throughput.
2. They can be used to probe nonconforming TCP senders, and to distinguish different type of TCP implementations in reality.
3. They also can provide information on revealing the enhanced capability of new version TCP. For example, how often TCP Reno's fast recovery is actually deployed in practice.

Observations:

1. The bottleneck for the throughput is in most cases that the send do not have enough data to send, rather than the network's capability.
2. It shows that up to now, TCP/newreno is the dominant implementation at end host.
3. Another important observation is that in general, the RTTs of one connection don't suffer large variation during the transmission. This is a good news for end-to-end multimedia transmission. That is, the bandwidth will not great degradation.

RTT estimation is described in [96]:

$$RTT = d1 + d2$$

Potential estimation error:

1. Under-estimate of cwnd: When duplicate ACKs loss happens, their method may indicate that the sender will launch a fast retransmission, which may cause cwnd and ssthresh decreased accordingly. Therefore, the proposed method will count ssthresh reduction twice. However, due to loss ACKs, the actual sender only reduce ssthresh once.

2. Over-estimate of cwnd: There are two cases, which may cause a over estimation for their measurement. Case I: The acknowledgements lost after they pass the measurement point. Case II: All of the data belongs to the current window lost before they reach the measurement point.

TCP Start-Up Prediction

In general, TCP start-up performance refers to the delay of three-way handshake for connection establishment and slow start. Cardwell, et al. proposed a model for TCP latency in [37].

There are three major assumptions for their model:

1. The sender is TCP/Reno family. However, the model can be applied to TCP/Newreon, SACK or FACK.

2. The random variable, r.v. for he loss in one round is I.I.D (Independent Identically distributed).

3. The connection could be few packet loss and be dominated by TCP connection setup and initial slow start.

Connection Setup Latency:

The procedure of three way handshake is illustrated in Figure 2.18.

The predicted handshake time is given by

$$E[D_h] = RTT + T_s \times (\frac{1-p_r}{1_2 p_r} + \frac{1-p_f}{1-2p_f} - 2)$$

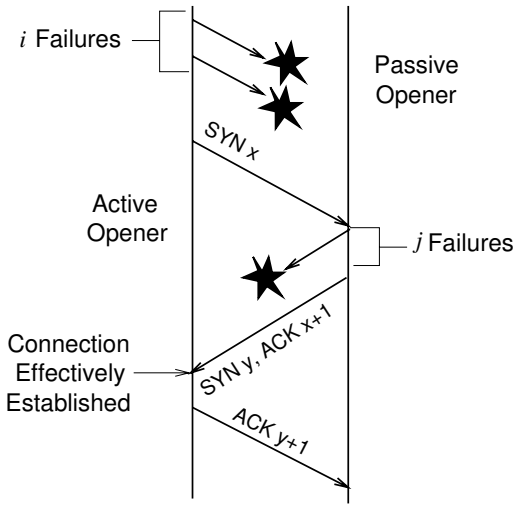


Figure 2.18: Illustration of TCP Connection Establish [37].

where T_s is the SYN timeout, p_f is the forwarding packet loss rate and p_r is the reverse packet loss rate.

Slow Start Latency:

The predicted slow start latency is given by

$$E[T_{ss}] = \begin{cases} RTT \times [\log_r (\frac{W_{max}}{\omega_1}) + 1 + \frac{1}{W_{max}}(E[d_{ss}] - \frac{\gamma W_{max} - \omega_1}{\gamma - 1})] & \text{if } E[W_{ss}] > W_{max} \\ RTT \times \log_r (\frac{E[d_{ss}](\gamma - 1)}{\omega_1} + 1) & \text{if } E[W_{ss}] \leq W_{max} \end{cases}$$

where $E[d_{ss}]$ is the number of data segment to be sent in the slow start phase given by

$$E[d_{ss}] = \frac{(1 - (1-p)^d)(1-p)}{p} + 1$$

where p is the data segment loss rate.

And $E[W_{ss}]$ is the expected window size after slow start given by

$$E[W_{ss}] = \frac{E[d_{ss}](\gamma - 1)}{\gamma} + \frac{\omega_1}{\gamma}$$

where γ is the rate of exponential increase of *cwnd* during slow start and ω_1 is the initial value of *cwnd*

TCP Latency and Throughput Prediction

Prediction of TCP performance has been extensively studied in many literatures [110, 154, 127, 145, 93, 149, 152]. In [110], Sikdar, et al. analyzed latency and steady state throughput of the major TCP versions: Tahoe, Reno, SACK.

The assumption they use are the following:

1. Only the delays of TCP performance is considered and other delays such as buffer delay is not taken into account.
2. The sender always has data to transmit, i.e., the congestion window is fulfilled always.
3. The window size of the receiver is not changed over time.
4. The acknowledgement by the receiver is the same as RFC 2581.
5. Nagle's algorithm and silly window avoidance is ignored.
6. TCP latency is presented in terms of RTT.

Generally speaking, TCP Tahoe deploys the slow start, congestion avoidance and fast recovery algorithm to TCP [91]. TCP Reno adds fast-retransmission to TCP. And TCP Sack improves Reno on recovery for multi-loss in one timeout.

TCP Latency:

The predicted transmission time is generalized as follows [110]

$$T_{transfer}(N) = t_{setup} + t_{dack} + (1-p)^N t_{nl}(N) + p(1-p)^{N-1} E\{t_{sl}(N)\} + t_{ml}(N)$$

where $T_{transfer}(N)$ is the transmission time for N packets, t_{setup} is the TCP setup time. t_{dack} is the delayed acknowledge time. t_{nl} is the transmission time for N packets. t_{sl} is the time for N packets with a single loss indication. And t_{ml} is the transmission time for N packets with M loss indication.

For example, in the case of TCP Tahoe, t_{setup}, $t_{nl}(N)$, $t_{sl}(N)$, $t_{ml}(N)$ are given by [110]

- $$t_{setup} = RTT + 2T_s(\frac{1-p}{1-2p} - 1)$$

- $$t_{nl}(N) = \begin{cases} \lceil 2\log_2(\frac{2N+4+3\sqrt{2}}{2\sqrt{2}+3(2)^{5/8}})\rceil RTT & \text{if } N \le N_{exp} \\ \left[n_{wm} + \lceil \frac{N-N_{exp}}{W_{max}} \rceil\right] RTT & \text{o.w.} \end{cases}$$

- $$t_{sl}(N) = \begin{cases} [t_{nl}(i) + r(n) + 1 + t_{lin}(a,b)]RTT & \text{if } nrnd_i^1 \ge 3 \\ \left[t_{nl}(i) + r(n) + t_{lin}(a,b) + I(nrnd_i^1 > 0) + E[TO]\right] RTT & \text{o.w.} \end{cases}$$

- $$t_{ml}(N) = E\{t_{sl}(m-1)\} + (M-2)E\{t_{M_loss}(D_{ave})\}$$

TCP Throughput:

The throughput can be represented by [110]

$$Thp = \frac{dMSS}{t_{ss}(p)} = \frac{MSS}{t_{ss}(p)p}$$

where $t_{ss}(p)$ is the expected time to transmit d packets given by

$$t_{ss}(p) = \frac{2}{c_{wm}(c_{wm}+1)} \sum_{h=1}^{c_{wm}} \sum_{j=1}^{h} t_{M_loss}(d)$$

where c_{wm} is defined by

$$c_{wm} = \min\{ \left[\frac{-1 + \sqrt{\frac{1+16}{p}}}{2} \right]^+ , W_{max}\}$$

Even though the close form of the throughput is different from that in [154], the results match with each other. Figure 2.19 compares the throughput of three schemes.

2.4 Active Queue Management: AQM

AQM is first proposed by Sally Floyd in 1993 [69]. The authors believes TCP end-to-end congestion control is not good enough. With the help of intermediate router, TCP

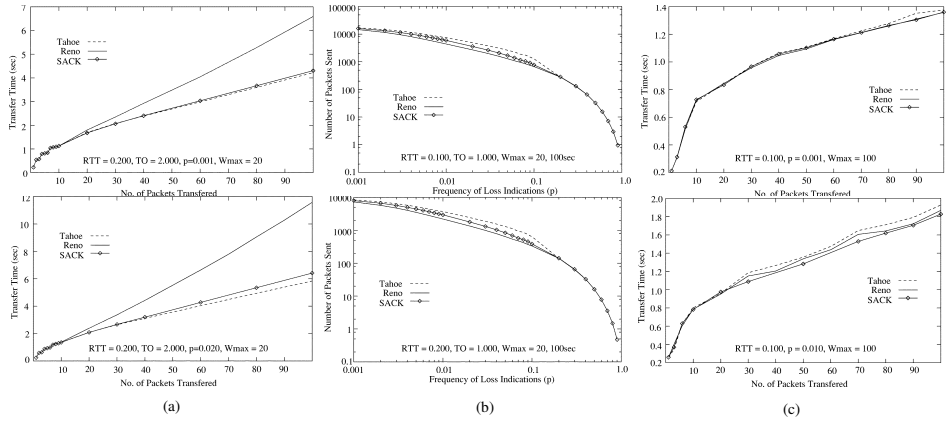


Figure 2.19: Throughput Comparison of Tahoe, Reno and Sack for (a), (b) correlated and (c) independent losses [154].

may yield better performance. In general, there are three representative AQM schemes: Random Early Detection (RED), Random Exponential Marking (REM) and Adaptive Virtual Queue (AVQ). Based on RED, there are some derivatives such as BLUE, GREEN and PI.

RED

Sally Floyd and V. Jacobson revolutionarily propose Random Early Detection (RED see [69]) routers and Explicit Congestion Notification (ECN see [62]) to help TCP end-to-end congestion avoidance in packet-switched networks.

The basic idea of RED plus ECN is that in addition to congestion control by TCP sender and receiver, the intermediate router take part in congestion avoidance through measuring the average queue size within the router (not instantaneous queue size). Then by dropping the packets or marking a bit in those packet headers, the router could inform the TCP sender that congestion will happen before it does. Specifically, when the average queue size is smaller than a preset threshold called (thresh_min), the router need not to do anything, which is the same as no RED deployment. While the average queue size is greater than thresh_min, the router will drop or mark every incoming packet according to a

certain probability, where the dropping or marking probability is determined by a function of the average queue size. By marking a packet, the router set a the ECN bit in the TCP header to notify the receiver that there is congestion in the middle of the network. After receiving such an notification, the TCP sender will reduce the window size, i.e., decrease the transmission rate, which may mitigate the workload to the network. With the help of the third party, TCP congestion control may outperform TCP stand alone.

Moreover, routers with RED keep the average queue size low while allowing occasional bursts of packets in the queue. During congestion, the probability that the router notifies a particular connection to reduce its window is roughly proportional to that connections share of the bandwidth through the router. RED routers are designed to cooperate with TCP. Further, as the authors pointed out that RED router has no bias against bursty traffic and avoids the global synchronization of many connections decreasing their window at the same time. The performance of RED plus ECN is depicted in Figure 2.20.

From 1993, RED experienced a rapid development such as FRED, SRED, BLUE etc. Most of research work can be categorized into the following groups by Sally Floyd [61], which gives us a very good overview on AQM (cf. many thanks to Sally Floyd).

- Enhance RED queue management [30, 67].
- Evaluation of RED [49, 146, 39, 64, 204, 34].
- Modification of RED and other AQM [18, 22, 23, 55, 56, 57, 34, 35, 92, 115, 116, 153, 169, 170, 205].
- RED with Drop Preference [41, 26].
- RED with per Flow Information [131, 16, 4, 142, 182].
- RED Implementation [63, 125, 168, 40, 38, 175, 74]

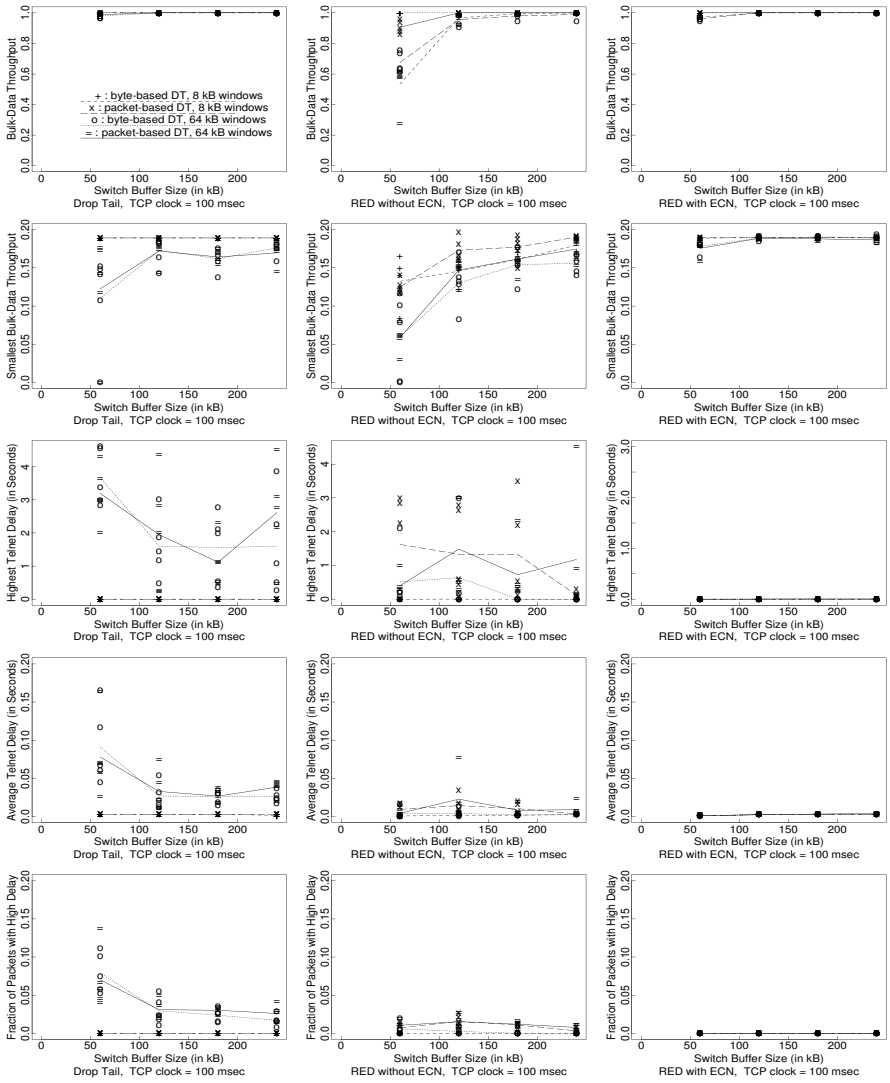


Figure 2.20: Performance of RED compared to Droptail. First column is droptail; Second column is RED without ECN; Third column is RED with ECN [69].

REM

Steven Low, et al. propose another AQM: Random Exponential Marking (REM) in [18, 135, 19, 155, 198, 5, 133, 199, 35, 156] to overcome the shortcoming of RED such as poor utilization and stability.

Two features which make REM different from REM are that (i) REM deploy a different marking function. Compared to RED, which uses uniform distributed marking function, REM uses exponential distributed marking function. The comparison is illustrated in Figure 2.21.

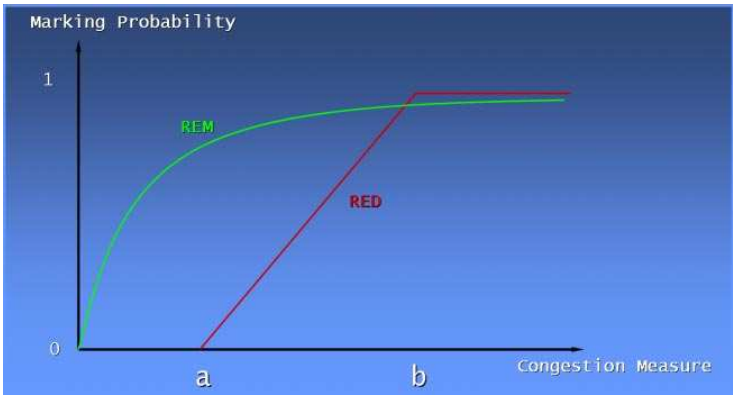


Figure 2.21: REM Marking Function compared to RED [18].

(ii) The measurement of congestion of REM is different from that of RED. RED uses average queue size as the congestion measure, while REM defines a price function as the measurement of congestion, which is given by [18]

$$p_l(t+1) = [p_l(t) + \gamma(\alpha_l(b_l(t) - b_l^*) + x_l(t) - c_l(t))]^+$$

where $p_l(t)$ is the price for queue l in period t, $\gamma > 0$ and $\alpha > 0$ are small constants, $b_l(t)$ is the aggregate buffer size at queue l in period t, b_l^* is target queue length, $x_l(t)$ is the aggregate input rate to queue l in period t, and $c_l(t)$ is the available bandwidth to queue l at period t. The term of $x_l(t) - c_l(t)$ gives the rate difference and the term of $b_l(t) - b_l^*$ shows queue difference.

AVQ

Another important AQM is Adaptive Virtual Queue (AVQ), which is proposed by R. Srikant et al in [122, 120, 116, 118, 128, 119, 117]. AVQ is motivated by the objective to design an AQM scheme with a low-loss rate, smaller-delay and higher link utilization.

The basic idea of AVQ is that the routers construct a virtual queue. The capacity of the virtual queue, called virtual capacity, is smaller than the actual capacity of the link. When there is a packet getting in the actual queue of the real link, the virtual queue is also updated to reflect such an arrival in the real queue. The router will check if the virtual queue buffer overflows or not. If yes, packets in the real queue are marked or dropped. By ECN, AVQ can collaborate with TCP end. Moreover, the virtual capacity has the capability to adapt to the total flow entering each link to obtain a higher utilization of the link. Figure 2.22and 2.23 display the operation of AVQ.

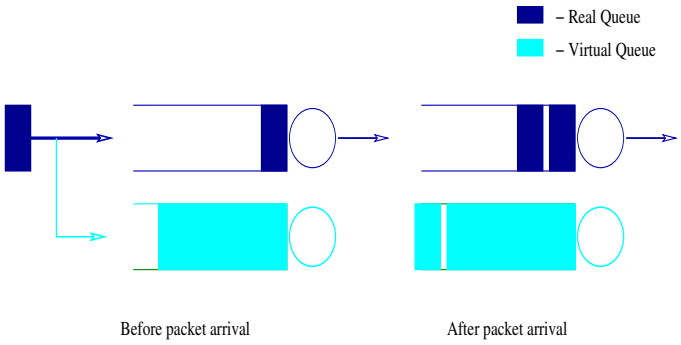


Figure 2.22: AVQ: Virtual Queue is not full [122].

AVQ can be characterized by the following pesudocode.

Extensive experiments have been done to illustrate the performance of AVQ. As the page limit of this report, we only borrow one result here, which is depicted in Figure 2.25.

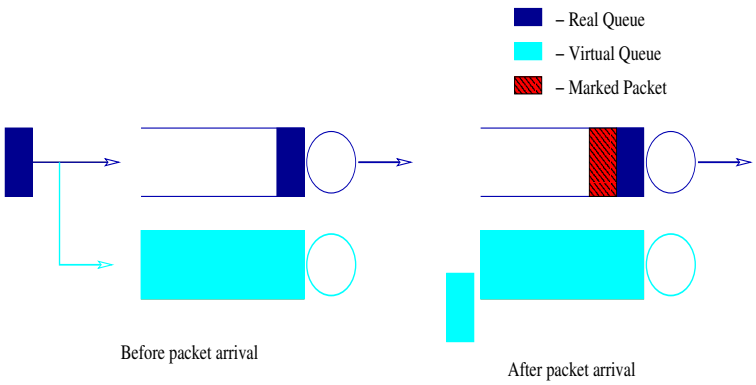


Figure 2.23: AVQ: Virtual Queue is full [122].

The AVQ Algorithm

At each packet arrival epoch do

$VQ \leftarrow \max(VQ \quad \tilde{C}(t \quad s), 0)$ /∗ Update Virtual Queue Size ∗/

If $VQ + b > B$

Mark or drop packet in the real queue

else

$VQ \leftarrow VQ + b$ /∗ Update Virtual Queue Size ∗/

endif

$\tilde{C} = \max(\min(\tilde{C} + \alpha * \gamma * C * (t \quad s), C) \quad \alpha * b, 0)$ /∗ Update Virtual Capacity ∗/

$s \leftarrow t$ /∗ Update last packet arrival time ∗/

Figure 2.24: AVQ Algorithm in Pseudocode [122].

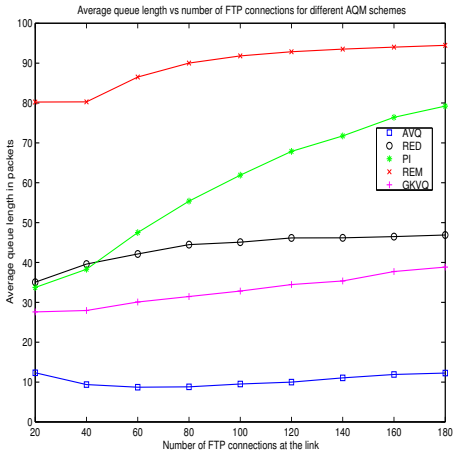


Figure 2.25: AVQ Performance compared to RED, PI, REM, GKVQ with regard to Average Queue Length [122].

BLUE

W. Feng et al. propose a new queue management algorithm, which is different from RED, called "BLUE" in [60, 58, 59, 53, 54, 57]. This work is awarded as the IEEE Communications Society 2003 William R. Bennett Prize for Best Paper in IEEE/ACM Transactions on Networking and the Best IBM Research Paper Award in Computer Science, Electrical Engineering, in Math 2002 at University of Michigan, Ann Arbor.

BLUE is a fundamentally different active queue management algorithm. Compared to RED using queue length as congestion measure, Blue uses packet loss and link utilization history to control congestion. BLUE calculate a probability to mark (or drop) packets. If the queue is continually dropping packets due to buffer overflow, BLUE increments the marking probability, thus increasing the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle, BLUE decreases its marking probability. The core algorithm is displayed in Figure 2.26

Through simulation, Feng et al, demonstrate BLUE outperform RED.

Upon packet loss (or $Q_{len} > L$) event:
    if ((now - $last_update$) > *freeze_time*)
        p_m := p_m + δ_1
        $last_update$:= now
Upon link idle event:
    if ((now - $last_update$) > *freeze_time*)
        p_m := p_m - δ_2
        $last_update$:= now

Figure 2.26: BLUE: Core Algorithm in Pseudocode [60].

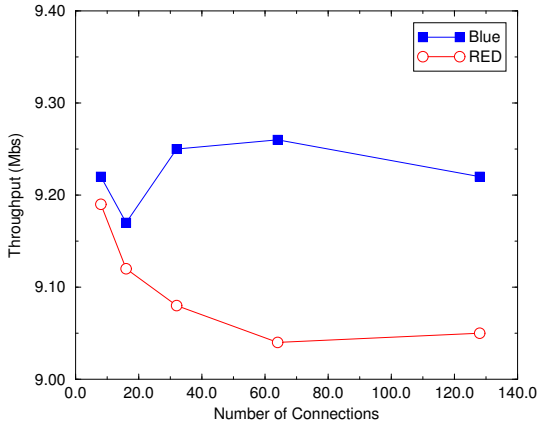


Figure 2.27: Blue: Throughput [60].

GREEN

Green is proposed by Bartek Wydrowski et al. in [195], which can be treated as a modification of REM queue management scheme. The GREEN algorithm is a rate-based control scheme which adapts the rate of congestion notification to the congestion signal, x_{est}, the estimated data arrival rate, which is based on flow rate. GREEN uses a threshold function. If the links estimated data arrival rate, x_{est} is larger than the link capacity, c_l, the probability of congestion notification, P, is increased by ΔP. Correspondingly, if x_{est} is smaller than c_l, P is decreased by ΔP. The algorithm deploys probabilistic marking of incoming packets as well, either by dropping packets, or setting the bit of ECN. Let the step function $U(x)$ be defined by [195]:

$$U(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{o.w.} \end{cases}$$

and P is updated by

$$P = P + \Delta P U(x_{est} - c_t)$$

where c_t is the target link capacity, which is assigned a value just below the actual link capacity c, typically $0.97c$, so that the queue size converges to 0. Incoming data rate estimation is updated using exponential averaging [195]:

$$x_{est} = (1 - e^{-\frac{Del}{K}}) \times \frac{B}{Del} + e^{-\frac{Del}{K}} \times x_{est}$$

where Del is the inter-packet delay, B the packet size and K the time constant.

PURPLE

Purple [159] stands for "Predictive Active Queue Management Utilizing Congestion Information".

Traditional AQM mechanisms have been built on heuristics to achieve the balance between maximize accommodating the possibly huge transient bursts and minimize queue size to avoid introducing unnecessary end-to-end delays. It often require manual tuning or have resulted in slow convergence. In contrast, the PURPLE approach predicts the impact

of its own actions on the behavior of reactive protocols and on the short-term future traffic without keeping per-flow state. PURPLE allows much faster convergence of the main AQM parameters, at least towards a local optimum, thereby smoothing and minimizing both congestion feedback and queue occupancy. To improve the quality of the prediction, the authors also passively monitor (using lightweight operations) information pertaining to the amount of congestion elsewhere in the network, for example, as seen by flows traversing this router.

PI

PI is the abbreviation of Proportional Integral controller proposed by Vishal Misra et al in [35, 34, 36]. They treat AQM as a control system. Thus adding a integral feedback to improve the performance.

A PI controller has a transfer function with the form given by

$$C(s) = K_{PI}\frac{s/z + 1}{s}$$

And the marking probability is updated by

$$p[k+1] = p[k] + a(q[k+1] - q_0) - b(q[k] - q_0)$$

where $p[k]$ is the marking probability at time slot k, $q[k]$ is the queue size at time slot k, q_0 is the desired queue length, and $a > 0,\ b > 0$ are constants determined by the design rules [35]. Figure 2.28 indicates the implementation of PI

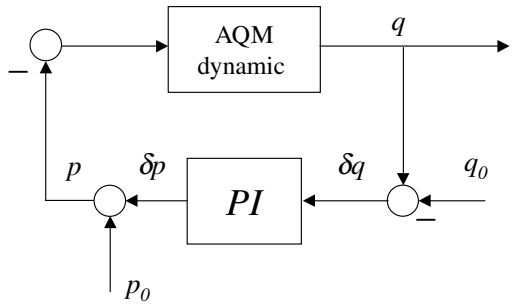


Figure 2.28: PI Controller [35].

Chapter 3

Stability Analysis of TCP under Linear Scale

3.1 Introduction and Background

In this chapter, we focus on "stability", one of the important issues of TCP/AQM [138, 137, 156, 34, 35, 36, 134, 140, 160, 43, 120, 85, 81]. From a practical point of view, an unstable TCP/AQM system may increase delay and jitter, which can be detrimental to some delay-jitter sensitive applications, and cause link under-utilization [138]. On the other hand, from a theoretical point of view, stability is the core problem in control field and extensively studied in both classic and modern control theory, where many rigorous and important results are achieved. The existing research on the stability of TCP/AQM can be broadly classified into two groups: local stability and global stability.

The local stability is obtained by first linearizing the system around the equilibrium point. Then, by applying Nyquist theorem to that linearized system, one can have a criterion for the local stability of the system [137, 36, 34, 35, 116]. However, due to the linearization, the local stability criterion is applicable only when the system operates near its equilibrium point. If the system operates on a region far away from its equilibrium point, then the nonlinear components may kick in and make such a linear approximation inaccurate. We

define a stability region of a system by a collection of initial points, from which the system eventually converges to its equilibrium point. Then, for a locally stable system, this stability region is usually small and limited around the equilibrium point.

On the other hand, unlike the local stability, the global stability requires that, starting from any the initial points (not limited to the points near the equilibrium point), the system converges eventually to the equilibrium point. Obviously, the global stability implies the local stability. In general, the global stability can be obtained by properly constructing a Lyapunov function [126, 194, 45, 85, 193, 155].

3.1.1 Definition of Lyapunov Stability

In order to further discuss stability rigorously, we need to know what is the stability rigorously. In this section, we will present the definition of Lyapunov stability and Lyapunov criterion. The stability we refer to in the following context is in the sense of Lyapunov's definition [108]. Consider a system given by the differential equation

$$\begin{aligned} \dot{x} &= f(x) \\ x(0) &= x_0 \end{aligned}$$

and suppose that the system has the unique equilibrium point 0, i.e., $f(x) = 0$ and $x = 0$.

The definition of Lyapunov stability is:

- The equilibrium point is said to be stable, if given an $\epsilon > 0$, there exists a $\delta > 0$, such that
$$||x(t)|| \leq \epsilon, \quad \forall t \geq 0, \quad \text{if} ||x_0|| \leq \delta$$
- The equilibrium point is asymptotically stable, if there exists a $\delta > 0$, such that
$$\lim_{t \to \infty} ||x(t)|| = 0, \quad \text{for all} ||x_0|| \leq \delta$$
- The equilibrium point is globally, asymptotically stable, if
$$\lim_{t \to \infty} ||x(t)|| = 0, \quad \text{for all initial conditions} ||x_0||.$$

The Lyapunov stability can be interpreted as follows. If the system is perturbed by a small amount from the equilibrium point, the system will stay around the equilibrium

point. Further, the asymptotic stability means that the trajectory of the system will come back the equilibrium point eventually. However, asymptotic stability requires the initial value of the system must be less than certain threshold. Global asymptotic stability enhance the asymptotic stability by any initial condition. And Lyapunov gave a sufficient condition to check stability.

We state Lyapunov criterion for three types of stability: Consider a continuously differentiable function $V(x)$ such that

$$V(x) > 0, \quad \forall x \neq 0$$

and $V(0) = 0$. Then

- if $\dot{V(x)} \leq 0, \ \forall x$, then the equilibrium point is stable.
- In addition, if $\dot{V(x)} < 0, \ \forall x \neq 0$, then the equilibrium point is asymptotically stable.
- Futher, if $V(x)$ is radially unbounded, i.e.,

  $$V(x) \to \infty, \quad \text{as } ||x|| \to \infty$$

  then the equilibrium point is globally asymptotically stable.

We note that Lyapunov criterion is only a sufficient conditions for stability. That is, if we cannot find a function $V(x)$ satisfying the criterion, we cannot say that the system is unstable.

3.1.2 Problems with the Existing Results of Stability

Although the existing results regarding stability disclose the valuable insight into the system design, they have their own limitations. For local stability, they are mostly designed for TCP-Reno instead of TCP-newReno [138, 34, 36]. [1] The major improvement of TCP-NewReno over TCP-Reno is that for TCP-Reno, multiple packet losses or packet marks within one RTT results in multiple reductions of window sizes, while for TCP-newReno, only one reduction of window sizes is incurred (see [68] for more details). We expect that TCP-newReno will be more stable than TCP-Reno as there will be at most one

[1] To the best of our knowledge, there is no result regarding the stability criterion of TCP-newReno under many flows.

reduction in the window size during each update period (RTT), thus it deviates less from the equilibrium point. For example, consider a system with $N = 1000$ TCP connections, the total capacity $C = 8.117 \times N$ packtes/sec, the slope for the marking function in RED (ρ_1 in Figure 3.3) $0.01/N$ and $RTT = 0.783$ sec (sum of two-way propagation delays and the queueing delay). Then, according to Corollary 6 in [138], the TCP-Reno/RED system will be unstable. While the same configuration for TCP-newReno/RED, our analysis in Section III shows that the norm of all the eigenvalues of the system with the same parameters (after suitably linearized) is 0.9346 (less than 1) and thus the system is locally stable. This is also verified by *ns*-2 [3] simulations displayed in Figure 3.1.

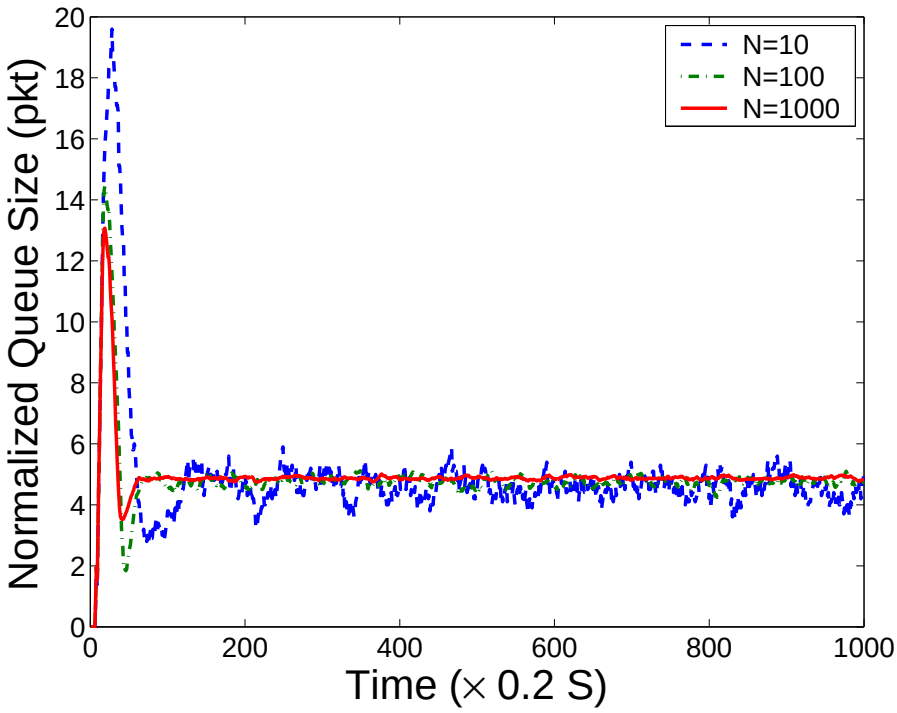


Figure 3.1: The stability convergence with the number of flows N .

Moreover, the current local stability criterion is a function of the marking slope of RED (ρ_1, cf. Figure 3.3) and Round Trip Time (RTT) with other parameters (say, the system capacity C and the number of flows, N). However, ρ_1 and RTT are closely related, which is ignored by the existing literature. In other words, RTT is not an independent parameter but a function of ρ_1. Specifically, by changing ρ, we can change RTT dramatically. Figure 3.2 shows an example that with the increase of ρ, the RTT decreases from more than

600 ms to around 200 ms, which should not be ignored when we determine the stability region with regard to ρ_1.

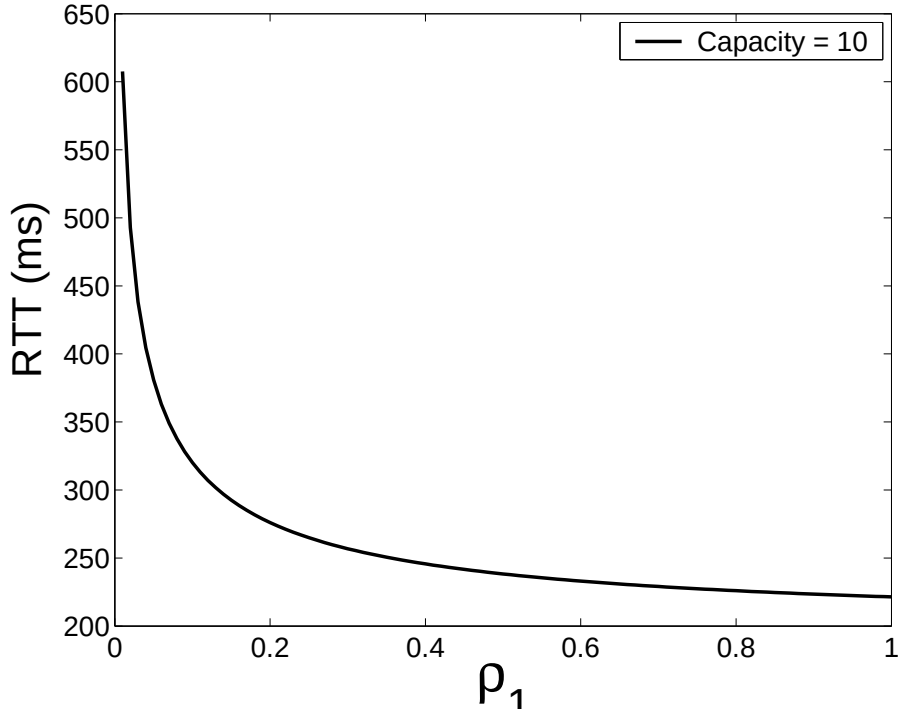


Figure 3.2: An Example to illustrate the relationship between RED slope ρ and RTT: the round trip physical propagation delay is 200 ms; capacity for each flow C is 10 packets/sec.

Furthermore, in [36, 34, 35], Hollot *et al.* used a fluid model to characterize the average behavior of system dynamics, which is only applied to the case of homogeneous RTTs. In [138, 137], Low *et al.* extended the results to the heterogenous RTTs case, while the fluid model is used to capture the behavior of *each* flow, which does not reflect the stability variation with the increase of the number of ongoing flows. Specifically, if we increase the number of flows (N) by 10 times, the system capacity (C) by 10 times and reduce the slope for marking in RED (ρ_1) by 10, then the local stability criterion in [138, 81, 36] remains the same. However, the simulation results in Figure 3.1 show that with more flows, the system becomes more stable as the variation of the normalized queue-length becomes smaller.

As to the global stability, the AQMs studied in [43, 45, 126, 158, 155, 193, 164, 163, 162, 12] are limited to rate-based marking schemes, and not applicable to any other popular queue-based marking schemes such as RED [69]. Still, to the best of our knowl-

edge, there has been no result about the global stability for a queue-based TCP system (e.g. a TCP-newReno/RED system with many flows in our case), because the queue dynamics is really an unfriendly nonlinear factor, which strongly prevents from constructing a Lyapunov function. In addition, the packet marking function for RED is a non-linear (usually piecewise-continuous) function, and this makes any theoretical analysis formidable even for a single flow case. Moreover, as the network evolves, the core router is shared by increasing number of flows. Since each flow has its own window-size dynamics, it becomes more difficult to find an appropriate Lyapunov function for such a nonlinear system with many flows. As a compromise, in most existing works including local stability literature, it has to be assumed that the system operates within a small region on which the marking function becomes linear, in order to make the analysis tractable [138, 137, 36]. Even though those results reveal valuable insights into the TCP/RED system, such simplification makes the results more or less deviate from the realistic RED scheme under certain circumstances.

3.1.3 Motivation and Objectives

With those observations in mind, we consider a TCP-newReno/RED system with many flows and focus on its local and global stability. We first propose a simple, yet accurate model to capture the key system dynamics of TCP-newReno/RED. Our model is motivated by the results in [43] and [187], in which the authors show that the system dynamics can be described by a normalized model with a single flow. Based on our normalized model, we derive the local stability criterion for the system within our regime. After we choose the system parameters such that the system is locally stable, we point out that there are some 'free' parameters we can freely choose without affecting the local stability of the system. Then, using numerical analysis, we show that, by carefully choosing the free parameter of the RED, it is always possible to make the system globally stable. Moreover, our numerical approach shows that the stability region of a TCP/RED system is quite irregular and very sensitive to the RED configuration. This is in accordance with the results in [147], indicating that the performance of TCP/RED is very sensitive to the parameter configuration. In our situation, a relatively small variation of RED configuration may cause a dramatic change of the stability region. Furthermore, our numerical approach is independent on the number of flows, which greatly reduce the simulation complexity for a large number of flows. For example, it is impossible for general simulator (say, *ns*-2 in [3]) to simulate the core router,

which is shared by over $10,000$ flows [72] due to the state explosion. However, our approach can easily handle such case, because our model capture a virtually single flow dynamics. More importantly, as we shall see, the more flows make our results more accurate, rather than suffer the scale complexity. Finally, we extend our model to the general Additive-Increase-Multiple -Decrease (AIMD (a, b)) case [2] and the heterogeneous RTTs' case, which suggest that our model can be applied to a more general and realistic settings.

Based on the results achieved, we propose three possible future work, which will further sharpen our understanding on stability issues: (i) Construct a suitable Lyapunov function to rigorously prove the global stability under our regime; (ii) Examine the impact of time-varying capacity on the stability.

3.2 System Model

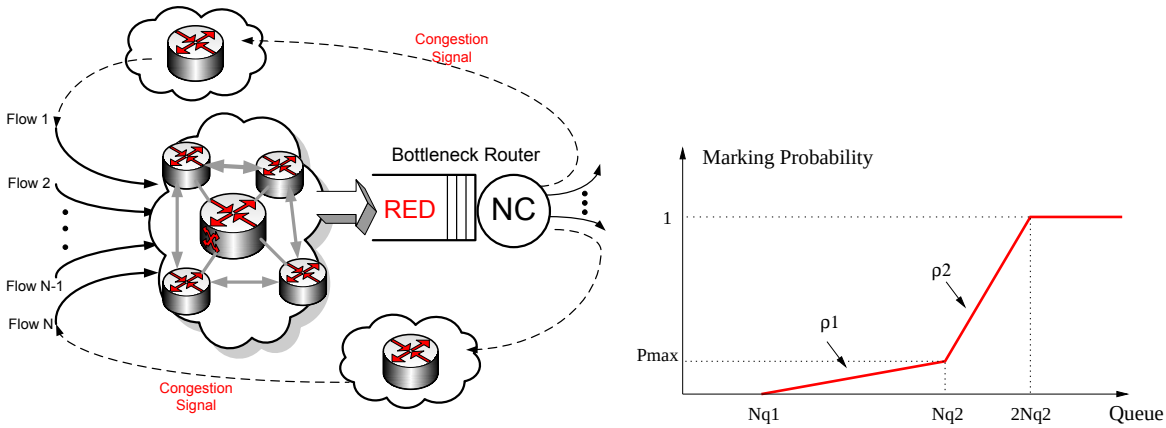
In this section, we describe the problem and introduce a normalized model that essentially captures the average behavior of the queue-length and window sizes. Then, we justify our model by comparing it with the existing one in [187, 184, 185, 188].

3.2.1 Problem Description

We first consider a system with identical TCP flows with the same round-trip-time (RTT). Later in this paper, we will also consider the case of heterogeneous RTTs in Section 3.6. Suppose that there are N long-lived TCP flows sharing one common bottleneck link with capacity Nc packets/sec, where c is a given constant. Let the two-way propagation delay be T sec and $Q^N(t)$ be the queue-length at time t. We set the length of each time slot to be one RTT, which is equal to the two-way propagation delay plus the queuing delay, i.e., $RTT = T + Q^N(t)/Nc$. In this way, the target window size per flow becomes $C = c \times RTT = cT + Q^N(t)/N$. We plot the system model in Figure 3.3-(a).

RED is implemented in the bottleneck router. According to the standard of RED specified in [30], the making/dropping probability is determined only by the queue-length. If the queue-length is smaller than $N \times q_1$, then none of the incoming packets is marked. (See Figure 3.3-(b).) If the queue-length falls between $N \times q_1$ and $N \times q_2$, the marking

[2] where the system will increase by a and decrease by b. For example, $(a = 1, b = 2)$ is the standard TCP AIMD algorithm.



(a) System model. (b) RED algorithm

Figure 3.3: System model and RED algorithm.

probability increases slowly with slope ρ_1. Once the queue-length falls between $N \times q_2$ and $2N \times q_2$, the marking probability increases sharply with slope ρ_2 ($\rho_2 > \rho_1$). Finally, if the queue-length is larger than $2N \times q_2$, all the incoming packets are marked.

For the TCP/RED system with many flows, we want to find the RED parameters as in Figure 3.3-(b) such that the system is globally stable. We see that some parameters in Figure 3.3-(b) are in fact redundant. For example, if we know q_1, q_2 and ρ_1, then we can immediately calculate the others.

3.2.2 Normalized Model

In this section, we present our normalized model based on the Markovian model in [187, 184, 185, 188], which is referred as the "original model" in the rest of the paper. As mentioned in the introduction, the original model capture the window size evolution of every flow and the queue-length dynamics (thus, $N+1$-dimension, e.g., see (2) and (5) in [187]), and this certainly makes it very difficult to analyze the system stability directly. Nevertheless, we will show that our model is able to capture the essence of the original model and also allows us to analyze the stability of the system.

Let $Q^N(t)$ be the real queue-length at the beginning of time slot t when there are N flows with link capacity NC. We then define the normalized queue-length $Q(t)$ by $Q(t) := Q^N(t)/N$. Let $W_i^N(t)$ be the window size of flow i at time slot t and again we

define the average (over N flows) window size $W(t)$ by $W(t) := \sum_{i=1}^{N} W_i^N(t)/N$. Further, we denote by $f^N(Q^N(t))$ the marking probability as a function of $Q^N(t)$ and assume that $f^N(Q^N(t)) = f^N(Q(t) \times N) = f(Q(t))$ for some function f. With these definitions, the system can be represented as follows:

Normalized Queue Dynamics:

$$Q(t+1) = [Q(t) + W(t+1) - C]^+, \tag{3.1}$$

where $[x]^+ = \max\{x, 0\}$. Note that (3.1) is equivalent to

$$Q^N(t+1) = \left[Q^N(t) + \sum_{i=1}^{N} W_i^N(t) - NC \right]^+,$$

which is just the Lindley recursion for the actual queue-length $Q^N(t)$.

Base Marking Function:

The base marking probability $f(x)$ is given by

$$f(x) = \begin{cases} 0 & \text{if } x \le q_1, \\ \rho_1(x - q_1) & \text{if } q_1 < x \le q_2, \\ P_{max} + \rho_2(x - q_2) & \text{if } q_2 < x \le 2q_2, \\ 1 & \text{if } 2q_2 < x, \end{cases} \tag{3.2}$$

where $\rho_1 = \frac{P_{max}}{q_2 - q_1}$ and $\rho_2 \frac{1 - P_{max}}{2q_2 - q_2}$. Then, the probability that at least one packet among those $W(t)$ packets is marked becomes

$$p(t) = 1 - (1 - f(Q(t)))^{W(t)}. \tag{3.3}$$

Average Window Dynamics: According to TCP Additive-Increase-Multiple-Decrease (AIMD) algorithm, the window size increases by 1 with probability $1 - p(t)$ and halves with probability $p(t)$. Thus, on average, the average window dynamics $W(t)$ can be described as in [138]

$$W(t+1) = (W(t) + 1)(1 - p(t)) + \frac{W(t)}{2} p(t). \tag{3.4}$$

Remark: Our approach here is to use the normalized queue-length and the average window size to capture the system dynamics. The merits of using the normalized model are (i) the system behavior gets greatly simplified; (ii) by appealing to the law of large numbers, (3.4) becomes more accurate for larger number of flows. We next provide numerical results to verify that our model is indeed accurate as long as the number of flows is large enough.

3.2.3 Model Comparison

We compare our normalized model in (3.1)–(3.4) with the original model. Table 3.1 lists parameters used in the numerical simulation for model comparison.

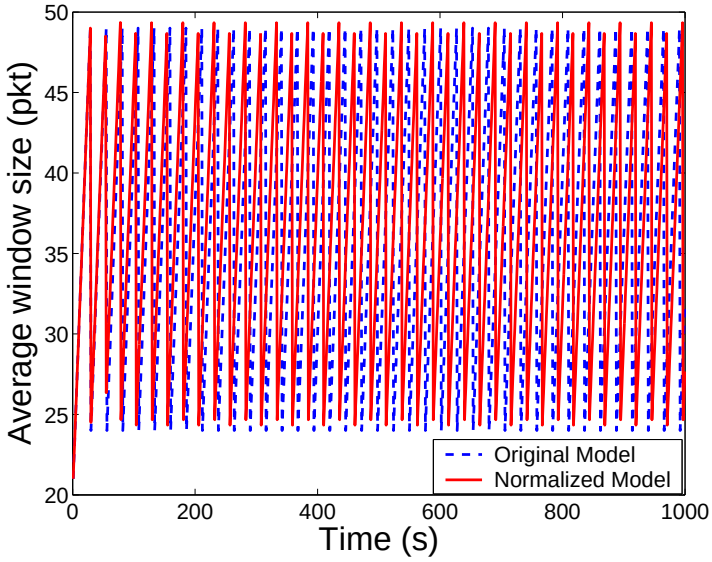
Table 3.1: Parameters for model comparison

Parameter	Value	Parameter	Value
# of flows (N)	1000	q_1	40
# of iterations (t)	1000	q_2	50
initial window size	20	P_{max}	1.0
initial queue-length	20	ρ_1	0.1
C	40	W_{max}	100

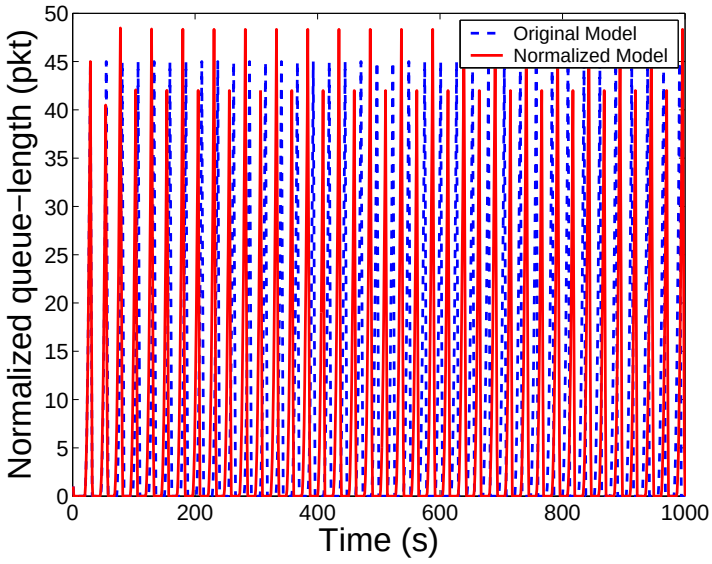
Figure 3.4 shows the comparison of the original model and the normalized model, when the system is *unstable*. (In other words, the normalized queue-length and the average window size suffer wild fluctuations.) Figure 3.4-(a) displays the average window size dynamics of the original model and our normalized model. We observe that the normalized model yields almost the same window dynamics as the original model. Similarly, Figure 3.4-(b) shows the comparison of queue dynamics of two models. The normalized model can effectively capture the wild fluctuation of the unstable queue.

In Figure 3.5, we compare two models when the system is *stable* (In other words, the fluctuations in the normalized queue-length and in the average window size are small.) Here, we replace the parameters ρ_1 and q_2 in Table 3.1 with $\rho_1 = 0.001$ and $q_2 = 1040$ such that the system is stable. Figures 3.5-(a) and (b) show the average window size and the normalized queue-length, respectively, drawn on the same scale as in Figure 3.4. We see that both the normalized queue-length and the average window size stay around some fixed values (equilibrium points) without much fluctuation.

Figure 3.6 displays the variance of the average window size and the normalized queue-length in the original model when the number of flows N increases. Clearly, the variance decreases sharply with the number of flows. In other words, when the number of flows is large enough, our normalized model is almost indistinguishable from the original model.



(a) Average window size

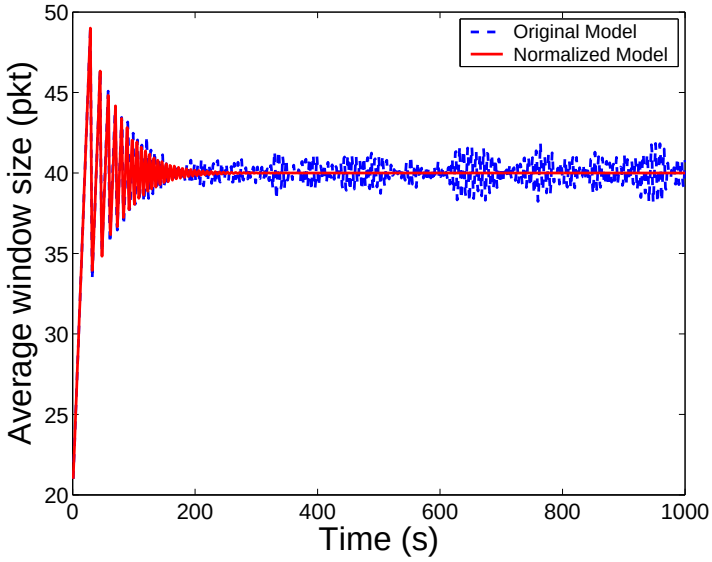


(b) Normalized queue-length

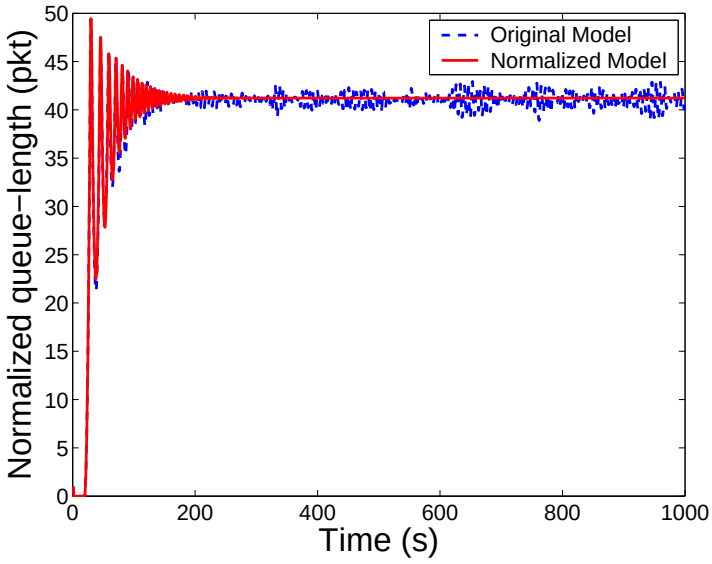
Figure 3.4: Model comparison when the system is unstable.

3.3 Local Stability Criterion

Based on the normalized model, we now study the local and the global stability of the system. In this section, we first configure the system to ensure the local stability.



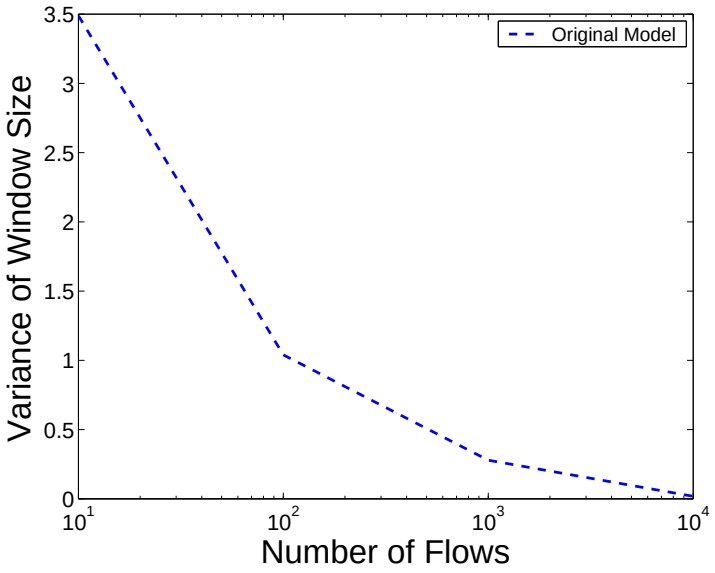
(a) Average window size



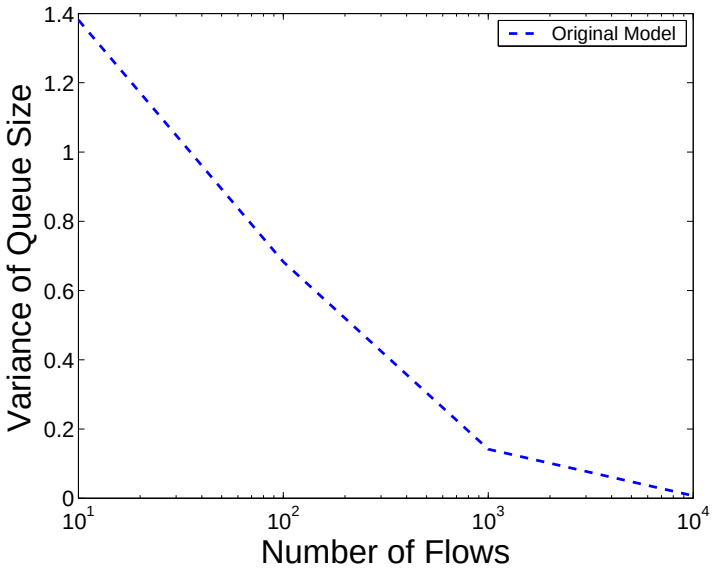
(b) Normalized queue-length

Figure 3.5: Model comparison when the system is stable.

Then, we show in Section 3.4 that we can always enlarge the stability region to entire space (i.e., global stability) by configuring other "free" system parameters.



(a) Average window size



(b) Normalized queue-length

Figure 3.6: Model comparison when the system is stable.

3.3.1 Linearizing System around Equilibrium Point

After the normalization, the system is still nonlinear and non-differentiable. In order to analyze it, we need to make some simplification, while still keep the system key

characteristics. Such methods are widely used in the literature [45, 138, 36].

Our approach is based on the following assumptions:

- **(A1)** $q_1 \leq Q(t) \leq q_2$.
- **(A2)** $C \gg 2$.

Since we are interested in the local stability of the system, it is reasonable to assume that the equilibrium queue-length is located between q_1 and q_2 [138, 137, 36]. Assumption (A2) states that the target window size for each long-lived TCP flow is far greater than 2, which is generally the case in practice. In fact, (A2) is just for simplicity and we can go without (A2) at the cost of more complicated computation.

To simplify the notation, we define

$$\tilde{Q}(t) = Q(t) - q_1. \tag{3.5}$$

Lemma 1 *If the system in (3.1)–(3.4) converges, then it must converge to the following unique equilibrium point:*

$$\lim_{t\to\infty} W(t) = C, \tag{3.6}$$

$$\lim_{t\to\infty} \tilde{Q}(t) = \frac{1}{\rho_1 C(1+\frac{C}{2})}. \tag{3.7}$$

Proof: As $t \to \infty$, $W(t+1) = W(t) = W(\infty)$, and $Q(t+1) = Q(t) = Q(\infty)$. By solving (3.1)–(3.4), we have $W(\infty) = C$. Then, using (3.4), we obtain the equilibrium marking probability $p(\infty) = 1/(1+\frac{C}{2})$. From (3.3) and (3.5), we have

$$\tilde{Q}(\infty) = \frac{1}{\rho_1}\left[1 - \left(1 - \frac{1}{1+C/2}\right)^{\frac{1}{C}}\right] \approx \left[\rho_1 C(1+\frac{C}{2})\right]^{-1},$$

where the approximation is due to $1-(1-x)^\alpha \approx \alpha x$ for small x and α and $C \gg 2$ from (A2). ■

Note that, from (3.2), the equilibrium marking probability f^* becomes

$$f^* = 1 - (1 - \frac{1}{1+\frac{C}{2}})^{\frac{1}{C}} \approx \left[C(1+\frac{C}{2})\right]^{-1}. \tag{3.8}$$

Since the queue is always positive under (A1), we can simplify the queue dynamics in (3.1) as

$$\tilde{Q}(t+1) = \tilde{Q}(t) + W(t+1) - C.$$

Near the equilibrium point, we have $p(t) \approx f(\tilde{Q}(t))W(t)$ again from $1-(1-x)^{\alpha} \approx \alpha x$. We can thus simplify the marking probability in (3.3) as $p(t) = \rho_1 C\tilde{Q}(t)$. Summarizing the above result, we obtain the following simplified model around the equilibrium point:

$$W(t+1) = W(t)\big(1 - \frac{\rho_1 C\tilde{Q}(t)}{2}\big) + (1 - \rho_1 C\tilde{Q}(t)) = F_1(W(t), \tilde{Q}(t)), \tag{3.9}$$

$$\tilde{Q}(t+1) = \tilde{Q}(t)\big(1 - \rho_1 C - \frac{\rho_1 CW(t)}{2}\big) + W(t) + 1 - C = F_2(W(t), \tilde{Q}(t)), \tag{3.10}$$

where $F_1, F_2 : \mathbb{R}^2 \to \mathbb{R}$ are appropriate functions.

3.3.2 Local Stability Criterion

Based on the above simplified model, we obtain the following criterion for the local stability.

Theorem 1 *The system described by (3.9) and (3.10) converges to the equilibrium point, if both* $|\lambda_1| < 1$ *and* $|\lambda_2| < 1$*, where*

$$\lambda_{1,2} = 1 - \frac{1}{2}\left[\rho_1(\frac{C^2}{2} + C) + \frac{1}{2+C} \pm \sqrt{[\rho_1(\frac{C^2}{2} + C) + \frac{1}{2+C}]^2 - 4\rho_1(\frac{C^2}{2} + C)}\right]. \tag{3.11}$$

Proof: From (3.9) and (3.10), we know that it is a discrete nonlinear system. Since we already know the equilibrium point of the system, we linearize the system near the equilibrium point to get

$$\begin{pmatrix} \Delta W(t+1) \\ \Delta\tilde{Q}(t+1) \end{pmatrix} = A \begin{pmatrix} \Delta W(t) \\ \Delta\tilde{Q}(t) \end{pmatrix},$$

where

$$\begin{aligned} \Delta W(t) &= W(t) - C, & \Delta W(t+1) &= W(t+1) - C, \\ \Delta\tilde{Q}(t) &= \tilde{Q}(t) - \tfrac{1}{\rho_1 C(1+\frac{C}{2})}, & \Delta\tilde{Q}(t+1) &= \tilde{Q}(t+1) - \frac{1}{\rho_1 C(1+\frac{C}{2})}, \end{aligned}$$

and matrix A is the Jacobian Matrix given by

$$A = \left(\begin{array}{cc} \frac{\partial F_1}{\partial W} & \frac{\partial F_1}{\partial \tilde{Q}} \\ \frac{\partial F_2}{\partial W} & \frac{\partial F_2}{\partial \tilde{Q}} \end{array} \right) \Bigg|_{W=W(\infty),Q=Q(\infty)}.$$

Substituting $W(t)$ and $\tilde{Q}(t)$ by their equilibrium values respectively, gives

$$A = \left(\begin{array}{cc} 1 - \frac{1}{2+C} & -\rho_1 C(\frac{C}{2}+1) \\ 1 - \frac{1}{2+C} & 1 - \rho_1 C(\frac{C}{2}+1) \end{array} \right).$$

The eigenvalue of matrix A is obtained by solving $\det|\lambda I - A| = 0$, i.e.,

$$\lambda^2 - \left(2 - \rho_1(\frac{C^2}{2} + C) - \frac{1}{2+C}\right)\lambda + 1 - \frac{1}{2+C} = 0.$$

The roots of the above equation are obtained in (3.11). Then, we know that if $|\lambda_1|$ and $|\lambda_2|$ are smaller than 1, then the system in (3.9) and (3.10) is stable. Specifically,

$$\left(\begin{array}{c} \Delta W(t) \\ \Delta \tilde{Q}(t) \end{array} \right) = \Xi \left(\begin{array}{cc} (\lambda_1)^t & 0 \\ 0 & (\lambda_2)^t \end{array} \right) \Xi^{-1} \left(\begin{array}{c} \Delta W(0) \\ \Delta \tilde{Q}(0) \end{array} \right),$$

where Ξ is the transformation matrix of A. Since $||\lambda_1|| < 1$ and $||\lambda_2|| < 1$, we have

$$\lim_{t\to\infty} (\lambda_1)^t = 0 \quad \text{and} \quad \lim_{t\to\infty} (\lambda_2)^t = 0,$$

and

$$\lim_{t\to\infty} \Delta W(t) = 0 \quad \text{and} \quad \lim_{t\to\infty} \Delta \tilde{Q}(t) = 0.$$

Therefore, we have

$$\lim_{t\to\infty} W(t) - C = 0, \quad \text{and} \quad \lim_{t\to\infty} \tilde{Q}(t) - \frac{1}{\rho_1 C(1+\frac{C}{2})} = 0.$$

In other words, the average window size converges to C and the normalized queue-length converges to $\frac{1}{\rho_1 C(1+\frac{C}{2})}$. This is accordance with the results in Lemma 1 and thus the system is locally stable. ■

Note that both λ_1 and λ_2 are smaller than 1 if

$$0 < \rho_1(C/2+1) + 1/(2+C) < 1. \tag{3.12}$$

This suggests that we could always find ρ_1 (possibly very small) such that the system is locally stable. However, one needs to be careful here because C and ρ_1 are related with

each other, which is ignored in the literature. Recall that C is the target window size of each flow, i.e., $C = c(T + Q^N(t)/Nc) = cT + Q(t)$, where cT is constant and $Q(t)$ is the normalized queue-length, which is a function of ρ_1. In general, if we decrease ρ_1, the equilibrium queue-length will be increased to get 'right' amount of marking to maintain equilibrium. This in turn makes C larger, which may compromise the effect of reduction in ρ_1 (see (3.12)). Therefore, we need to know the relationship between ρ_1 and C to precisely determine the local stability.

If the system is stable, it will converge to the equilibrium point eventually. Then at the equilibrium point, $C = cT + Q(\infty)$, and by (3.7), $Q(\infty) = \left[\rho_1 C(1 + \frac{C}{2})\right]^{-1} + q_1$. So, we have $C - cT = \left[\rho_1 C(1 + \frac{C}{2})\right]^{-1} + q_1$. That is,

$$\rho_1 C(\frac{C}{2} + 1)(C - cT - q_1) = 1. \tag{3.13}$$

There is no technical difficulty to solve (3.13) to obtain the roots in terms of ρ_1. We can obtain three roots for (3.13), but only one root is meaningful and the other two are either complex numbers or negative numbers.

3.4 From Local Stability to Global Stability

To the best of our knowledge, there has been no result on how to extend the local stability to the global stability for the system under our consideration (cf. Figure 3.3-(a)).

Theorem 1 shows that a suitable choice of ρ_1 always ensures the local stability of the system. Recall that we have defined the stability region as the set of points, starting from which the system converges to its equilibrium. In this section, we will show that, by properly choosing other "free" parameters that have nothing to do with the local stability, we can always configure the system such that it converges to its unique equilibrium point starting any initial points outside of the local stability region, i.e., the system is globally stable. We first consider a box region and show that all the points outside the box as initial states will eventually move into the inside of the box. Then, we enumerate the points inside the box region as initial states to check whether or not the system converges to its equilibrium.

3.4.1 Convergence From Outside a Box

Lemma 2 *Consider a box with vertices at* $(0,0)$, $(0,W_{max})$, $(2q_2,W_{max})$, *and* $(2q_2,0)$. *If the system (defined by (3.1)–(3.4)) starts from any points outside the box, it will eventually fall into the inside of the box.*

Proof: Without loss of generality, we can assume that there exists $n < \infty$ such that $2^{n-1}C \leq W_{max} < 2^nC$. Let us consider region displayed in Figure 3.7.

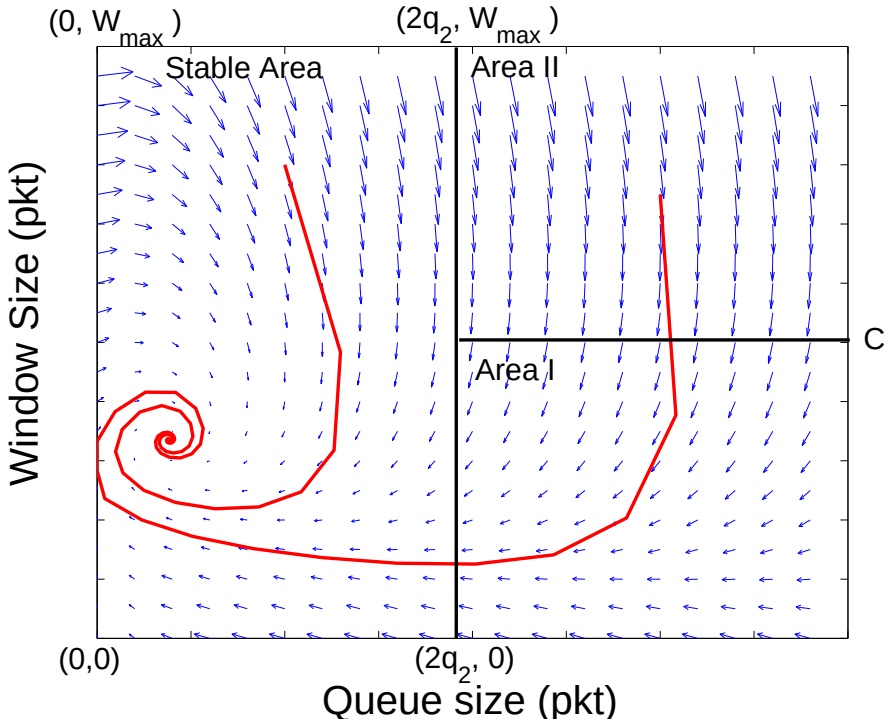


Figure 3.7: The Bound of Q(t) and W(t).

- Case I: Consider a point $(W(t),Q(t))$ in area (I), i.e., $W(t) \leq C$ and $Q(t) > 2q_2$. Since $f(Q(t)) = p(t) = 1$, the window size will halve at $t+1$, i.e., $W(t+1) = W(t)/2 < W(t) < C$ and $Q(t+1) = Q(t) + W(t+1) - C < Q(t)$. In other words, $(W(t+1),Q(t+1))$ moves toward the box and it will eventually fall back into the box.

- Case II: If the point $(W(t), Q(t))$ is in area (II) at time t, it will eventually go back to area (I). Since $f(Q(t) = p(t) = 1$, we have $W(t+1) = W(t)/2$. If $W(t+1) \leq C$, then we are done. When $W(t+1) >> C$, note that it takes at most n iteration for $(W(t), Q(t))$ to fall back into the area (I) as $W(t+n) = W(t)/2^n < C$.

Thus, all the points in the area (I) and (II) will eventually move into the inside of the box and this completes the proof. ■

3.4.2 Choosing Free Parameters to Ensure Global Stability

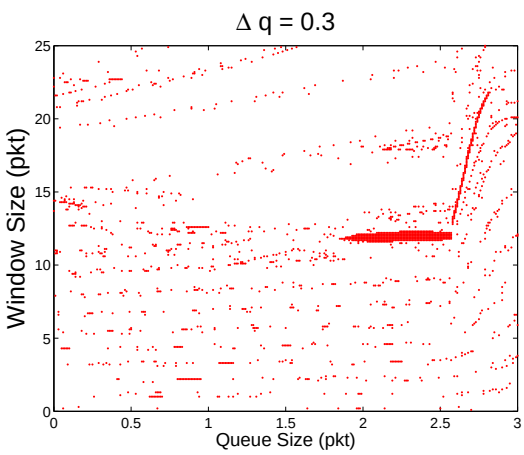
In this section, we will show that q_2 is the"free" parameter we mentioned earlier. With properly chosen q_2, starting from any point within the box region defined in Lemma 2, the system converges to the equilibrium point.

Specifically, note that in Theorem 1 the local stability depends only on ρ_1 and C. So, we are free to choose any q_2 without affecting the local stability of the system. Let $q_2 = Q(\infty) + \Delta q$, where $Q(\infty)$ is the equilibrium queue-length as in (3.5) and (3.7), and Δq is the deviation from the equilibrium point. From $\Delta q = 0$, we increase Δq until the system converges starting from any points inside the box region. Then, by Lemma 2, the system will be globally stable.

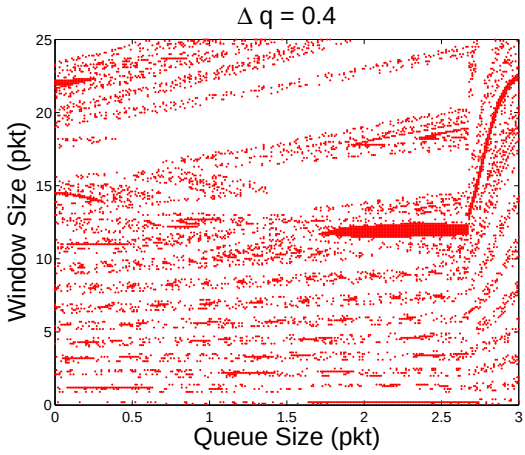
Now, we explain how to proceed numerical analysis based on our normalized model. For every given $\Delta q > 0$, we consider points inside the box with vertices at $(0, 0)$, $(0, W_{max})$, $(2q_2, W_{max})$, $(2q_2, 0)$, and set those points as initial point of $(Q(0), W(0))$. Then, by iterating (3.1) – (3.4), we can check whether or not the system starting from those points converges to the equilibrium point. If Δq is too small, not all the points inside the box will converge. So we gradually increase Δq, until all the points inside the box as initial points will converge to the equilibrium point. In addition, Lemma 2 asserts that the system starting from outside of the box will fall back into inside of the box. In this way, we can configure the system to be globally stable as long as all the points inside the box converges.

Example I: We set $C = 12$ and $\rho_1 = 0.01$. By (3.11), we have $\lambda_{1,2} = 0.5443 \pm i0.7952$ Since $||\lambda_{1,2}|| = 0.9636 < 1$, we obtain $Q(\infty) = 1.2764$, $W(\infty) = 12$ by (3.7). Without loss of generality, we assume $W_{max} = 20$. We plot 3 figures with different small Δq in Figures 3.8-(a), (b) and (c). The points falling in the dark region will converge to the equilibrium point. On the contrary, the points within the blank region will not converge

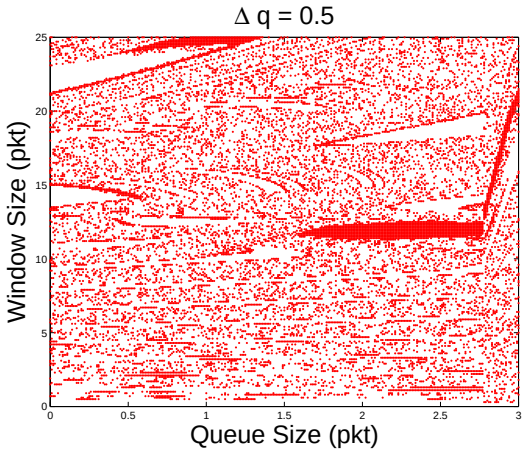
to the equilibrium point, which means the system will oscillate for ever. We can obtain Figure 3.8-(a) with $\Delta q = 0.3$, Figure 3.8-(b) with $\Delta q = 0.4$, and Figure 3.8-(c) with $\Delta q = 0.6$. When $\Delta q = 0.6$, the whole box area is covered by convergence points.



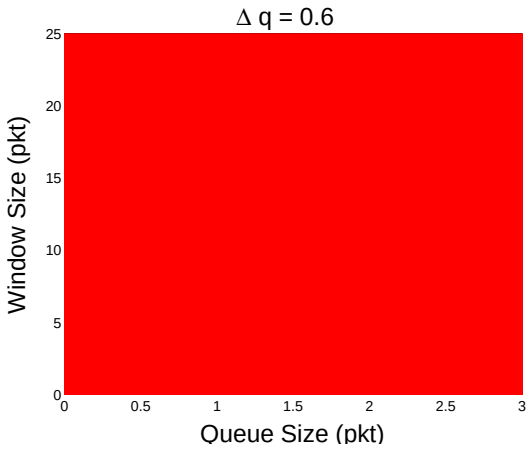
(a) $\Delta q = 0.3$.



(b) $\Delta q = 0.4$.



(c)$\Delta q = 0.5$.

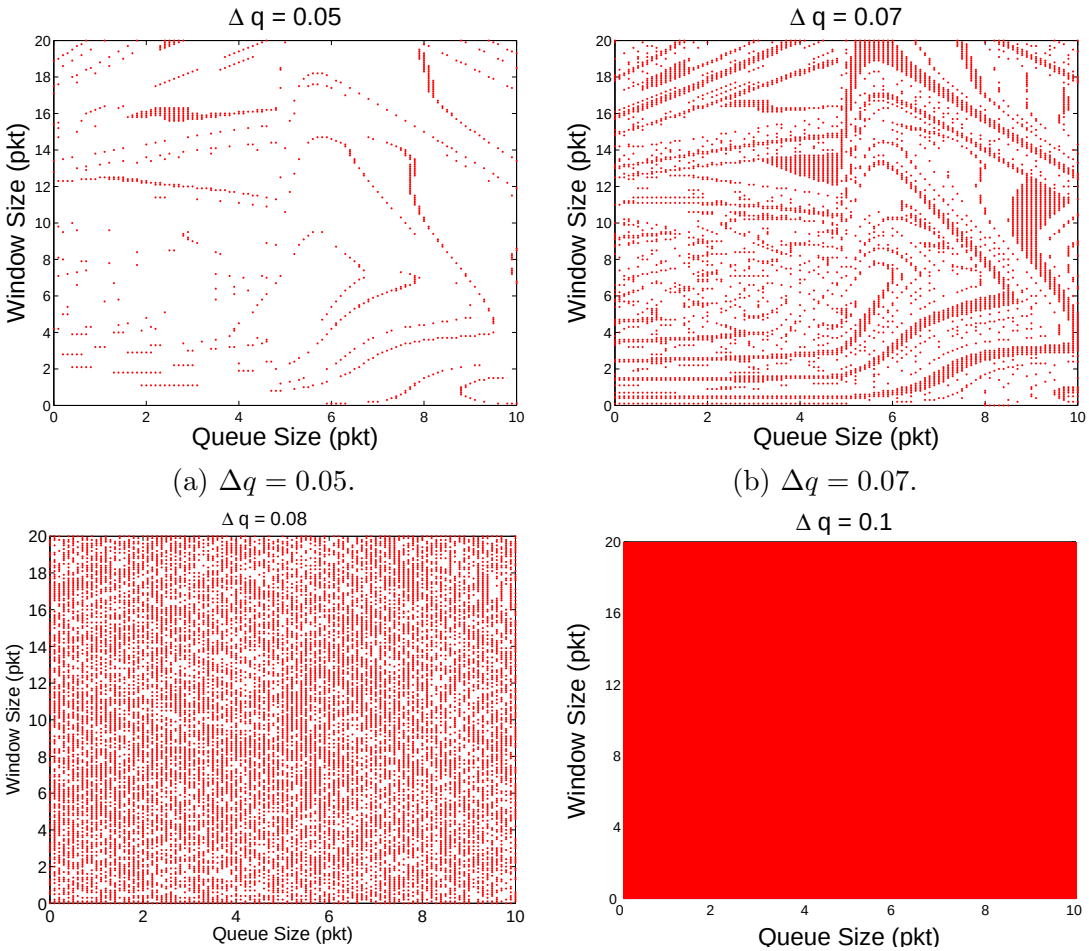


(b) $\Delta q = 0.6$.

Figure 3.8: Example I: the stable region of Q(0) and W(0) with $Q(\infty) = 1.2764$, $W(\infty) = 12$ for $\Delta q = 0.3, 0.4, 0.5, 0.6$.

Example II: We set parameters of (3.1) and (3.4) as follows. $C = 6.70$, $\rho_1 = 0.01$. By simple calculation of (3.11), we have $\lambda_{1,2} = 0.7968 \pm i0.5002$, and we can verify that $||\lambda_{1,2}|| < 1$. Then the system is locally stable with the unique equilibrium point of $(Q(\infty) = 3.8237, W(\infty) = 6.70)$ by (3.7). Without loss of generality, we assume $W_{max} = 20$. We plot 2 figures with different small Δq in Figure 3.9. The points falling in the red region will converge to the equilibrium point. On the contrary, the points within the blank region

will not converge to the equilibrium point, which means the system will oscillate for ever. Figures 3.9-(a) and (b) display the stable points with $\Delta q = 0.05$ and $\Delta q = 0.07$ respectively. When $\Delta q = 0.1$, all the points within the box area: $(0, 0)$, $(0, 20)$, $(10, 20)$, $(10, 0)$ will converge to the equilibrium point $(3.8237, 6.70)$.



(a) $\Delta q = 0.05$.

(b) $\Delta q = 0.07$.

(b) $\Delta q = 0.08$.

(d) $\Delta q = 0.1$.

Figure 3.9: Example II: the stable region of Q(0) and W(0) with $Q(\infty) = 3.8237$, $W(\infty) = 6.70$ for $\Delta q = 0.05, 0.07, 0.08, 0.1$.

From Figures 3.9-(a) and (b), we have the following observations: (i) The profile of stable points are quite irregular. It is hard to obtain a close form expression for the profile. (ii) With a very small variation of Δq (say, from $\Delta q = 0.05$ to $\Delta q = 0.07$), stable region changes dramatically. (iii) A relative small Δq (for example, $\Delta q = 0.1$) is enough to obtain global stability.

3.4.3 Discussion

From the numerical results, we can see that a quite small Δq is sufficient to make the whole box region stale (say $\Delta q = 0.15$). In other words, the whole box region can be quickly filled as Δq increases. An intuitive interpretation of our results is that if the queue falls within the region with slope of ρ_1 (i.e., between q_1 and q_2), by local stability the queue will converge to equilibrium point eventually; If the queue falls outside the region with slope of ρ_1, (say, the region with the slope of ρ_2, i.e., between q_2 and $2q_2$), the queue will fall back to the region with slope of ρ_1. This is because most packets will be marked and most of TCP flows will reduce their window size by half. Then the number of packets arrival to the router will be smaller and the queue-length will decrease to the range of ρ_1.

Our results show that a relative small Δq is enough for the system to be globally stable. In reality, the propose RED suggests enough margin is necessary in order to accommodate other traffic patterns such as burst traffic arrival, webmice traffic and UDP traffic. Hence, in reality, the sufficient margin (i.e, Δq) is available. In most cases, if a system is configured to be locally stable, it is also globally stable.

More importantly, our numerical analysis is independent of the number of flows, which is based on the iteration of (3.1)–(3.4). This is especially important, when we study a system with a huge number of flows. According to [72], the core router is shared by over 10,000 flows. The simulator (say *ns*-2 [3]) cannot simulate such a system because of the state explosion and hence it is infeasible to verify global stability by the simulator. However, our numerical approach can easily be proceed and finish within a very short time (around several minutes), which means our approach has good scalability. Meanwhile, with more flows our results become more accurate.

3.5 General AIMD (a, b)

In this section, we apply our model to the general AIMD (a, b) case. Specifically, if there is no packets marked within one RTT, the window size will increase by a; if there is at least one packets marked, the window size will drop by b. In particular, $(a = 1, b = 2)$ is what we already analyzed in the previous sections.

The queue dynamics and marking function will keep the same as (3.1), (3.2) and (3.3), but the window dynamics becomes more general given by

$$\begin{aligned} W(t+1) &= (W(t)+a)[1-p(t)] + \frac{W(t)}{b}p(t) \\ &\doteq W(t)[1-(1-\frac{1}{b})\rho CQ(t)] + a(1-\rho CQ(t)) \end{aligned}$$

Here, we apply the same approximation approach as we did for $(a = 1, b = 2)$ case.

Lemma 3 *If the system converge, then it must converge to the following* **unique** *equilibrium point.*

$$\lim_{t\to\infty} W(t) = C \tag{3.14}$$

$$\lim_{t\to\infty} \tilde{Q}(t) = \frac{a}{\rho_1 C(C-\frac{C}{b}+a)} \tag{3.15}$$

$$\lim_{t\to\infty} p(t) = \frac{a}{C-\frac{C}{b}+a} \tag{3.16}$$

Proof: As $t \to \infty$, $W(t+1) = W(t) = W(\infty)$ and $Q(t+1) = Q(t) = Q(\infty)$. By solving model given by (3.1) – (3.4), we have $W(\infty) = C$. Then, using (3.4), we obtain the equilibrium marking probability

$$p(\infty) = \frac{a}{C-\frac{C}{b}+a}.$$

By (3.3) and assumption (A2), we have

$$Q(\infty) = \frac{a}{\rho_1 C(C-\frac{C}{b}+a)} + q_1$$

Then $\tilde{Q}(\infty) = \frac{a}{\rho_1 C(C-\frac{C}{b}+a)}$ by (3.5).

■

$$\begin{aligned} W(t+1) &= W(t)[1-(1-\frac{1}{b})\rho CQ(t)] + a(1-\rho CQ(t)) \\ &= F_1(W(t), \tilde{Q}(t)) \\ \tilde{Q}(t+1) &= \tilde{Q}(t)[1-\rho_1 C - \frac{\rho_1 CW(t)}{2}] + W(t) + 1 - C \\ &= F_2(W(t), \tilde{Q}(t)), \end{aligned}$$

3.5.1 Local Stability Criterion

Our results suggest that from the stability point of view, we need to increase the value of "b", which may lead to fast convergence. The convergence rate is not sensitive to the value of "a".

Theorem 2 *The system described by (3.17) and (3.17) converges to the equilibrium point, if both*

$|\lambda_1| < 1$ *and* $||\lambda_2|| < 1$, *where*

$$\lambda_{1,2} = \frac{1}{2}\left[(\xi_{11} + \xi_{22}) \pm \sqrt{\xi_{11}^2 + \xi_{22}^2 - 2\xi_{11}\xi_{22} + 4\xi_{12}\xi_{21}}\right] \tag{3.17}$$

and $\xi_{11}, \xi_{12}, \xi_{21}, \xi_{22}$ *are defined in (3.18).*

Proof of Theorem 2: Let A be the Jacobian Matrix given by

$$A = \begin{pmatrix} \xi_{11} & \xi_{12} \\ \xi_{21} & \xi_{22} \end{pmatrix} = \begin{pmatrix} \frac{\partial F_1}{\partial W} & \frac{\partial F_1}{\partial Q} \\ \frac{\partial F_2}{\partial W} & \frac{\partial F_2}{\partial Q} \end{pmatrix} \tag{3.18}$$

where

$$\begin{aligned}
\frac{\partial F_1}{\partial W} &= 1 - (1 - \frac{1}{b})\rho C Q(t) \\
&= 1 - (1 - \frac{1}{b})\frac{a}{C - \frac{C}{b} + a} \\
\frac{\partial F_1}{\partial Q} &= -(1 - \frac{1}{b})\rho C^2 - a\rho C \\
&= -\rho C[(1 - \frac{1}{b})C + a] \\
\frac{\partial F_2}{\partial W} &= \frac{\partial F_1}{\partial W} \\
&= 1 - (1 - \frac{1}{b})\frac{a}{C - \frac{C}{b} + a} \\
\frac{\partial F_2}{\partial Q} &= 1 + \frac{\partial F_1}{\partial Q} \\
&= 1 - \rho C[(1 - \frac{1}{b})C + a]
\end{aligned}$$

From (3.17) and (3.17), we know that it is a discrete nonlinear system. Since we already know the equilibrium point of the system, we linearize the system near the

equilibrium point.

$$\begin{pmatrix} \Delta W(t+1) \\ \Delta \tilde{Q}(t+1) \end{pmatrix} = A \begin{pmatrix} \Delta W(t) \\ \Delta \tilde{Q}(t) \end{pmatrix}$$

where

$$\begin{aligned} \Delta W(t) &= W(t) - C \\ \Delta \tilde{Q}(t) &= \tilde{Q}(t) - \frac{1}{\rho_1 C(1+\frac{C}{2})} \\ \Delta W(t+1) &= W(t+1) - C \\ \Delta \tilde{Q}(t+1) &= \tilde{Q}(t+1) - \frac{1}{\rho_1 C(1+\frac{C}{2})} \end{aligned}$$

and matrix A is the Jacobian Matrix given in (3.18).

Then the eigenvalues of matrix A can be obtained by setting $\det|\lambda I - A| = 0$, that is

$$\lambda^2 - (\xi_{11} + \xi_{22})\lambda + \xi_{11}\xi_{22} - \xi_{12}\xi_{21} = 0$$

We have the eigenvalues as in (3.17).

Then, we know that if $||\lambda_1||$ and $||\lambda_2||$ are smaller than 1, then the system in (3.17) and (3.17) is stable. Specifically,

$$\begin{pmatrix} \Delta W(t) \\ \Delta \tilde{Q}(t) \end{pmatrix} = \Xi \begin{pmatrix} (\lambda_1)^t & 0 \\ 0 & (\lambda_2)^t \end{pmatrix} \Xi^{-1} \begin{pmatrix} \Delta W(0) \\ \Delta \tilde{Q}(0) \end{pmatrix}$$

where matrix Ξ is the transformation matrix of matrix A.

Since $|\lambda_1| < 1$ and $|\lambda_2| < 1$, we have

$$\lim_{t\to\infty} (\lambda_1)^t = 0 \quad \text{and} \quad \lim_{t\to\infty} (\lambda_2)^t = 0.$$

Thus,

$$\lim_{t\to\infty} \Delta W(t) = 0 \quad \text{and} \quad \lim_{t\to\infty} \Delta \tilde{Q}(t) = 0.$$

Therefore, we have

$$\begin{aligned} \lim_{t\to\infty} W(t) - C &= 0 \\ \lim_{t\to\infty} \tilde{Q}(t) - \frac{a}{\rho_1 C(C - \frac{C}{b} + a)} &= 0. \end{aligned} \tag{3.19}$$

■

3.5.2 Discussion

In this section, we discuss the impact of parameter (a, b) to the system stability.

Figure 3.10 shows that with the increase of b, the norms of the eigenvalues (two values overlap) decrease sharply. The smaller eigenvalues, the faster the system converges. Similar observation is obtained for the increase of a. This result suggests that we can increase the values of a and b to improve the system stability. Intuitively, the relatively larger of a and b will make the system response to the congestion signal quickly.

However, if either a or b is too large, the system will become unstable again. Figure 3.11 shows that under the same configuration as Figure 3.10, if we keep increasing b or a, after certain point the norms of the two eigenvalues bifurcate and one of the eigenvalues will be greater than 1, which means the system becomes unstable. This result indicates that we cannot arbitrarily increase the value of a and b. Even though the larger value of a and b will increase the system response, the excessively large a and b will make the system over reaction to the congestion signal.

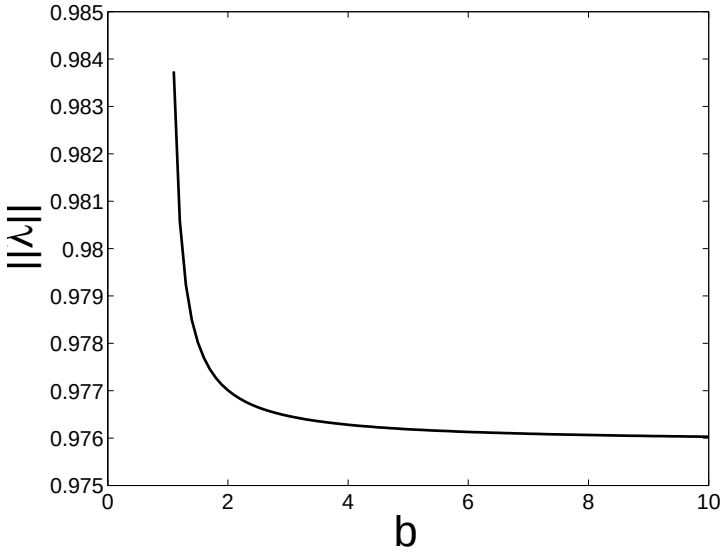
3.6 Heterogeneous RTTs

In reality, each flow goes through its own access network and then arrive at the bottleneck router. Thus, the round trip time (RTT) will be different for each flow. In order to capture RTT heterogeneity, we denote $g(x)$ as the probability density function (pdf) of RTTs and discretize RTTs into m levels [3]. For example, if we set the granularity of RTTs to be 20 ms, then ψ_1 is the percentage of RTT in $[0, 20]$ ms, ψ_2 is the percentage of RTT in $[20, 40]$ ms, and ψ_5 is for $[80, 100]$ ms, etc. If there are no flows with RTTs in $[60, 80]$ ms, then $\psi_4 = 0$. By this way, we can classify RTTs into m groups indexed by i, and let ψ_i be the percentage of RTTs in i^{th} level. We can obtain $\psi_i, i = 1, 2, \cdots, m$ as follows.

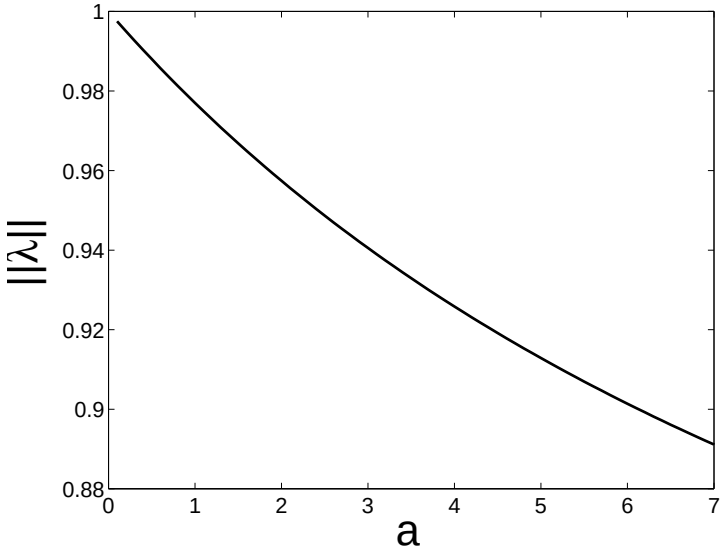
$$\psi_i = \int_{i-1}^{i} g(x)\, dx, \quad \text{where} \quad \psi_1 + \psi_2 + \cdots + \psi_m = 1.$$

Let N_i $(i = 1, 2, \cdots, m)$ be the number of flows of i^{th} level, N be the total number of flows, i.e., $N = \sum_{i=1}^{m} N_i$. It is clear that $\psi_i = N_i/N$. Assume that N_i is also sufficient large, as

[3] The value of 'm' can be determined according to the system requirement. The larger 'm' leads to fine RTT granularity at the expense of more cumbersome description for system dynamics.



(a) Impact of b

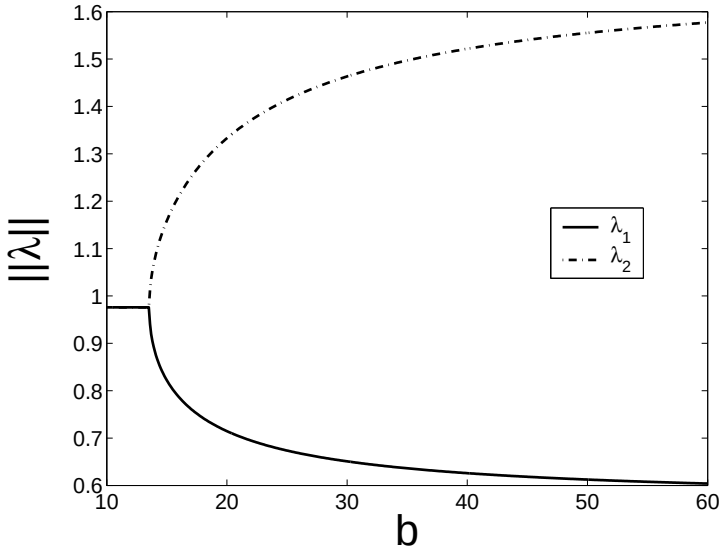


(b) Impact of a

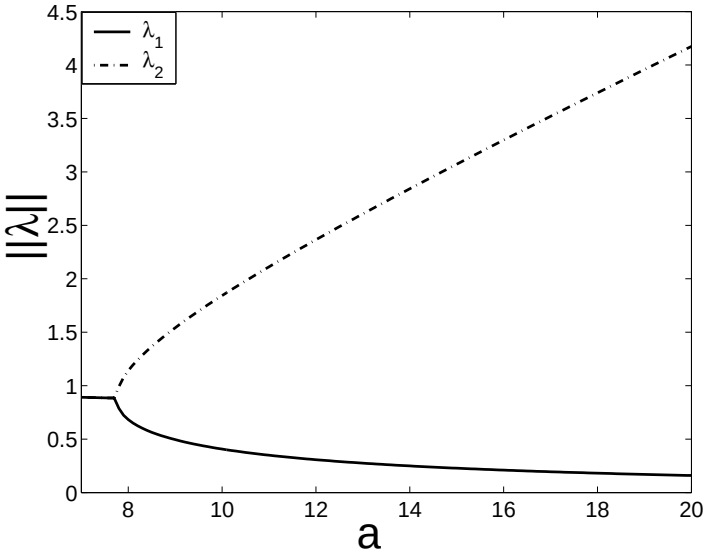
Figure 3.10: The impact of b and a to the system stability with $C = 20$, $\rho_1 = 0.01$.

we shall see later that the influence of the small amount of flows will be averaged out by the large amount of others flows and become negligible.

Let $Q^N(t)$ be the real queue length at the beginning of time slot t. The time slot of 1^{st} group of flows is set to be 1 and the time slot of 2^{nd} group of flows is set to be 2 and



(a) Impact of b



(b) Impact of a

Figure 3.11: The impact of b and a to the system stability with $C = 20$, $\rho_1 = 0.01$.

so on. We define the normalized queue length $Q(t)$ (over N flows) by $Q(t) := Q^N(t)/N$. Let $W(t)$ be the normalized window size over the total number of flows N at time slot t. We denote $W_i^N(t)$ as the window size of flow i at time slot t and again we define the normalized window size (over N_i flows) for the i^{th} group $W_i(t) := \sum_{i=1}^{N_i} W_i^N(t)/N_i$. Further,

we denote $f^N(Q^N(t))$ as the marking probability according to the real queue length and the normalized marking probability (over N flows) $f(Q(t)) := f^N(NQ(t)) = f^N(Q^N(t))$.

3.6.1 Normalized Model for Heterogeneous RTTs

In this section, we generalize our normalized model for homogeneous RTTs to capture the system dynamics for heterogeneous RTTs. According to the specification of TCP-newReno [68], the multiple marks/drops within one RTT will incur only one reduction in window size. Then, the marking probability of the i^{th} group of flows, $p_i(t)$, is given by

$$p_i(t) = 1 - (1 - f(Q(t)))^{W_i(t)}. \tag{3.20}$$

Normalized Window Dynamics

Since $W(t)$ be the normalized window size over the total number of flows N at time slot t, it can be written as

$$W(t) = \sum_{i=1}^{m} \frac{\psi_i W_i(t)}{i}, \tag{3.21}$$

where the normalized window size for each group, $W_i(t)$ ($i = 1, \cdots, m$), is being updated at every i^{th} RTT. For i^{th} RTT group, during the first $i-1$ time slots, the window size remains unchanged because it takes i time slots for the congestion signal to arrive at the TCP senders. At the i^{th} time slot, the window evolves according to AIMD: some flows in the i^{th} group will increase their window sizes with probability $1 - p_i(t)$ and some flows will decrease their window sizes by half with probability $p_i(t)$. In order to describe such dynamics, we use the expression $\lceil \frac{t \mod i}{i-1} \rceil$ to generate $\{0, 0, \ldots, 1, 0, \ldots, 1, 0, \ldots\}$ sequence, where all the elements are zero except every i^{th} position being one. This representation corresponds to the indicator function used in [185, 188].

Hence, the window dynamics follows

$$\begin{aligned} W_i(t+1) = \Bigg\{ \left[1 - \lceil \frac{t \mod i}{i-1} \rceil \right] W_i(t) + \lceil \frac{t \mod i}{i-1} \rceil \\ \times \left[(W_i(t)+1)(1-p_i(t)) + \frac{W_i(t)}{2} p_i(t) \right] \Bigg\}, \end{aligned} \tag{3.22}$$

where $p_i(t)$ given in (3.20).

Next, by substituting (3.22) into (3.21), we have the normalized window size $W(t)$

dynamics with respect to $W_i(t)$ given by

$$W(t+1) = \sum_{i}^{m} \frac{\psi_i}{i} \left\{ \left[1 - \lceil \frac{t \mod i}{i-1} \rceil \right] W_i(t) + \lceil \frac{t \mod i}{i-1} \rceil \right. \\ \left. \times \left[(W_i(t)+1)(1-p_i(t)) + \frac{W_i(t)}{i} p_i(t) \right] \right\}.$$

Note that if we define $\eta_i(t) = W_i(t)/W(t)$ for $i = 1, 2, \cdots, m$, from (3.21), we can rewrite the above in terms of $W(t)$ and $\eta_i(t)$ as follows.

$$W(t+1) = \sum_{i=1}^{m} \frac{\psi_i}{i} \left\{ \left[1 - \lceil \frac{t \mod i}{i-1} \rceil \right] \eta_i(t) W(t) + \lceil \frac{t \mod i}{i-1} \rceil \right. \\ \left. \times \left[(\eta_i(t)W(t)+1)(1-p_i(t)) + \frac{\eta_i(t)W(t)}{i} p_i(t) \right] \right\}. \tag{3.23}$$

3.6.2 Equilibrium Point for Heterogeneous RTTs

Lemma 4 *If the system in (3.1), (3.20)–(3.22) converges, it must converge to the following unique equilibrium point:*

$$\lim_{t\to\infty} W(t) = C, \qquad \lim_{t\to\infty} W_i(t) = \tilde{C}, \qquad \lim_{t\to\infty} Q(t) = \frac{1}{\rho_1 \tilde{C}(1 + \frac{\tilde{C}}{2})},$$

where $\tilde{C} = C \left[\psi_1 + \frac{1}{2}\psi_2 + \cdots + \frac{1}{m}\psi_m \right]^{-1}$.

Proof: As $t \to \infty$, $W(t+1) = W(t)$ and $Q(t+1) = Q(t)$. By solving equations in (3.1), (3.20)–(3.22), we have $W(\infty) = C$. Since $Q(t)$ goes to $Q(\infty)$, the marking probability seen by each TCP flow remains unchanged and hence the average window size is the same for all the flows, i.e., $W_1(\infty) = W_2(\infty) = \cdots = W_m(\infty)$ and $p_1(\infty) = p_2(\infty) = \cdots = p_m(\infty)$ by (3.20). Then, from (3.21), we have

$$W_i(\infty)\frac{1}{1}\psi_1 + W_i(\infty)\frac{1}{2}\psi_2 + \cdots + W_i(\infty)\frac{1}{m}\psi_m = C.$$

This gives $W_i(\infty) = C \left[\psi_1 + \frac{1}{2}\psi_2 + \cdots + \frac{1}{m}\psi_m \right]^{-1} = \tilde{C}$. Given $W_i(\infty)$, by solving (3.22), we obtain $p_i(\infty) = \frac{1}{1+\frac{\tilde{C}}{2}}$. Finally, similarly as in Theorem 1, by using a linearized version of (3.20) when $\tilde{C} = W_i(\infty) \gg 2$, we obtain $Q(\infty)$, and the result follows. ■

Lemma 4 suggests that when the system is stable, the normalized window sizes for different groups of flows $W_i(t)$ are the same regardless of different RTTs. Intuitively speaking, the coordination among all the flows is because they share the common queue. Since the queue length converges to constant if stable, the marking probability within one RTT is the same for different groups. This results in the same normalized window size for different groups.

3.6.3 Stability Condition for Heterogeneous RTTs

Based on Lemma 4, we now show a criterion for the local stability.

Theorem 3 *The system described by (3.1)–(3.21) is locally stable, if both* $||\lambda_1|| < 1$ *and* $||\lambda_2|| < 1$*, where*

$$\lambda_{1,2} = \frac{1}{2}\left[\sigma_{11} + \sigma_{22} \pm \sqrt{\sigma_{11}^2 + \sigma_{22}^2 - 2\sigma_{11}\sigma_{22} + 4\sigma_{12}\sigma_{21}}\right],$$

and $\sigma_{11}, \sigma_{12}, \sigma_{21}, \sigma_{22}$ *are given by (3.24).*

Since near the equilibrium point $f(Q(t))$ is quite small in general, we can approximate $p_i(t)$ by $p_i(t) = f(Q(t))W_i(t)$.

Proof of Theorem 3: For simplicity, we denote (3.23) and (3.1) by $F_1^H(W(t), Q(t))$ and $F_2^H(W(t), Q(t))$, respectively. We know that near the equilibrium point, $\eta_i(t) \to W_i(\infty)/W(\infty) = \left[\psi_1 + \frac{1}{2}\psi_2 + \cdots + \frac{1}{m}\psi_m\right]^{-1}$ for all $i = 1, \cdots, m$. Set $\eta = \eta_i(\infty)$. Since $W(t)$ evolves according to $t \pmod m$, the convergence rate is also different for each $t \pmod$

m) as follows.

$$\begin{aligned}
\frac{\partial F_1^H(t)}{\partial W} &= \psi_1[\eta - \eta^2\rho_1 Q(t)W(t) - \eta\rho_1 Q(t)] + (\frac{\psi_2}{2} + \frac{\psi_3}{3} + \cdots + \frac{\psi_m}{m})\eta, \\
\frac{\partial F_1^H(t)}{\partial Q} &= \psi_1[-\frac{\eta^2}{2}\rho_1 W^2(t) - \eta\rho_1 W(t)], \\
\frac{\partial F_1^H(t+1)}{\partial W} &= (\psi_1 + \frac{\psi_2}{2})[\eta - \eta^2\rho_1 Q(t)W(t) - \eta\rho_1 Q(t)] + (\frac{\psi_3}{3} + \cdots + \frac{\psi_m}{m})\eta, \\
\frac{\partial F_1^H(t+1)}{\partial Q} &= (\psi_1 + \frac{\psi_2}{2})[-\frac{\eta^2}{2}\rho_1 W^2(t) - \eta\rho_1 W(t)], \\
\frac{\partial F_1^H(t+m-1)}{\partial W} &= (\psi_1 + \frac{\psi_2}{2} + \frac{\psi_3}{3} + \cdots + \frac{\psi_m}{m}) \times [\eta - \eta^2\rho_1 Q(t)W(t) - \eta\rho_1 Q(t)], \\
\frac{\partial F_1^H(t+m-1)}{\partial Q} &= (\psi_1 + \frac{\psi_2}{2} + \frac{\psi_3}{3} + \cdots + \frac{\psi_m}{m}) \times [-\frac{\eta^2}{2}\rho_1 W^2(t) - \eta\rho_1 W(t)].
\end{aligned}$$

Let

$$B_k = \begin{pmatrix} \frac{\partial F_1^H(t+k)}{\partial W} & \frac{\partial F_1^H(t+k)}{\partial \tilde{Q}} \\ \frac{\partial F_2^H(t+k)}{\partial W} & \frac{\partial F_2^H(t+k)}{\partial \tilde{Q}} \end{pmatrix},$$

where $k = 0, \cdots, m-1$. The total Jacobian Matrix over m time slot $B(m)$ is then given by

$$B(m) = B_m \times B_{m-1} \times \cdots \times B_1 = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}. \tag{3.24}$$

From (3.24), by repeating the same argument as in the proof of Theorem 1, we are done. ■

3.6.4 Decomposition of Stability Condition for Heterogeneous RTTs

Since the system consists of many groups of flows with different RTTs, it is worth asking what are the roles of different groups of flows with regard to the whole system stability. In order to further investigate the impact of each group of flows on the local stability, we look at a group of flows of our interest with the assumption that other groups of flows are at their equilibrium points. Then, we have the following.

Theorem 4 *The characteristic matrix of i^{th} group of flow A_i is given by (3.27). The i^{th} group of flows is locally stable, if the corresponding eigenvalues for the matrix in (3.27) are less than one in magnitude.*

Proof of Theorem 4: We look at the system dynamics for the i^{th} group of flows while keeping all the others at their equilibrium points. As before, during the first $i-1$ time slot, the window size of i^{th} group remains the same. Hence, for $j = 0, 1, \ldots, i-1$, the Jacobian matrix for the i^{th} group $(W_i(t), Q(t))$ at j^{th} time slot is given by

$$A_{i,j} = \begin{pmatrix} \frac{\partial W_i(t+j)}{\partial W_i} & \frac{\partial W_i(t+j)}{\partial Q} \\ \frac{\partial Q(t+j)}{\partial W_i} & \frac{\partial Q(t+j)}{\partial Q} \end{pmatrix}.$$

At the i^{th} time slot (i.e., $j = i$), the TCP sender will receive the congestion signal and update the window size according to AIMD rule. Then, we have $W_i(t+i) = (W_i(t+i-1)+1)(1-p_i(t+i-1)) + \frac{W_i(t+i-1)}{2} p_i(t+i-1)$. Applying this relationship to Jacobian Matrix, we get

$$\begin{aligned} \frac{\partial W_i(t+i)}{\partial W_i} &= 1 - \frac{1}{1+\frac{\tilde{C}}{2}} - \frac{1}{\tilde{C}(1+\frac{\tilde{C}}{2})} = \xi_{11}, \qquad (3.25) \\ \frac{\partial W_i(t+i)}{\partial Q} &= -\rho_1 \tilde{C}(1+\frac{\tilde{C}}{2}) = \xi_{12}, \qquad (3.26) \\ \frac{\partial Q(t+i)}{\partial W_i} &= \xi_{11}\frac{\psi_i}{i}, \\ \frac{\partial Q(t+i)}{\partial Q} &= 1 + \xi_{12}\frac{\psi_i}{i}. \end{aligned}$$

Then the Jacobian Matrix at i^{th} time slot $A_{i,i}$ is given by

$$A_{i,i} = \begin{pmatrix} \xi_{11} & \xi_{12} \\ \xi_{11}\frac{\psi_i}{i} & 1+\xi_{12}\frac{\psi_i}{i} \end{pmatrix}.$$

Since the RTT for the i^{th} group of flows is i time slots, within the first $i-1$ time slots, the TCP sender does not receive any congestion signal and the window size for the i^{th} group flows $W_i(t)$ remains unchanged. Then, we have $W_i(t+1) = W_i(t)$ for $t = 0, 1, \ldots, i-1$, and by applying this to the Jacobian Matrix, we get

$$A_{i,j} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \text{ for } j = 1, 2, \cdots, i-1.$$

The total Jacobian Matrix over i time slots A_i is given by

$$A_i = A_{i,i} \times A_{i,i-1} \times \cdots \times A_{i,1} = \begin{pmatrix} \xi_{11} + (i-1)\xi_{12} & \xi_{12} \\ \xi_{11}\frac{\psi_i}{i} + (i-1)(1+\xi_{12}\frac{\psi_i}{i}) & 1+\xi_{12}\frac{\psi_i}{i} \end{pmatrix}. \qquad (3.27)$$

where ξ_{11} and ξ_{12} are from (3.25) and (3.26). Then, the local stability follows by enforcing all the solutions of $\det|\lambda_i I - A_i| = 0$ to be inside a unit circle. ■

3.6.5 Discussion

Lemma 4 shows that the equilibrium point is determined by the RTT and the associated weight ψ_i for group i. This indicates that a single flow or a group of flows with small weight (ψ_i) does not contribute much to the equilibrium point and plays a minor role when there exist a large amount of other flows. The same observation can be applied to Theorem 3. The stability of the whole system will be largely unaffected by a flow with long RTT, and this is in contrast to Theorem 2 in [138], where the RTT of each flow may affect the maximum RTT $\overline{\tau}$ or minimum RTT $\underline{\tau}$ and hence affect the stability of the system considered in [138]. Note that the model in [138] is more suitable for a small system with a small number of flows, while our results suggest that from a whole system point of view we apply different stability criterion for a large system with many flows taking into account the proportion of flows with similar RTT.

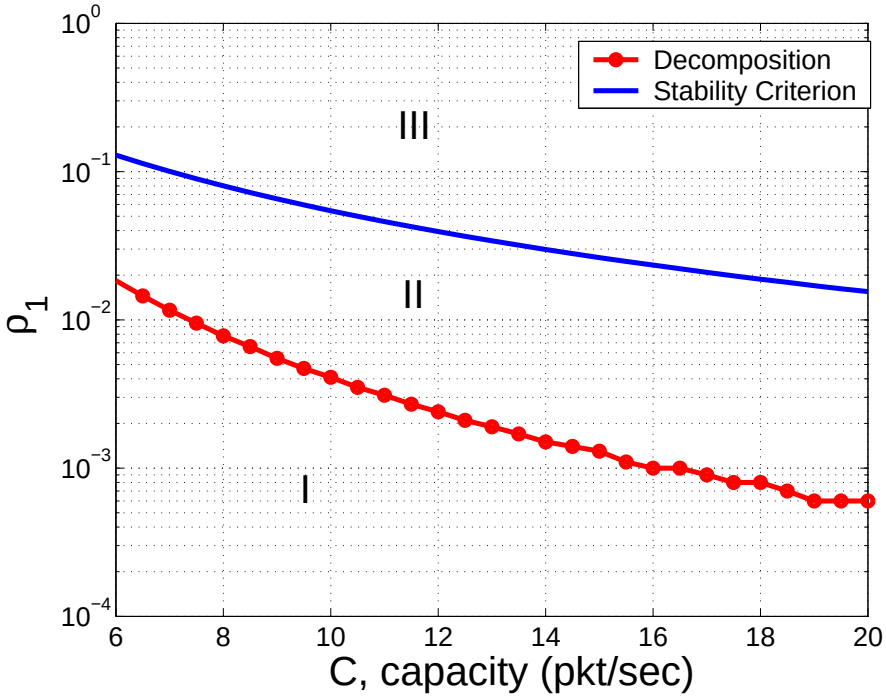


Figure 3.12: The comparison of stability region and decomposition for Heterogenous RTTs.

Theorem 4 allows us to investigate how much each group of flows contributes to the whole system stability. Figure 3.12 shows a numerical example of Theorem 3 and Theorem 4, where the system consists of two groups of flows with $\psi_1 = 0.9$ and $\psi_2 = 0.1$.

The RTTs of flows in group 2 are set to be twice larger than those in group 1. From Figure 3.12, we see that the stability criterion in Theorem 3 (see the line without markers) separates the whole area into two regions. The region above the stability criterion (i.e., area III) means that for a given system with capacity C, if ρ_1 is above this line, the system is locally unstable. If ρ_1 is below the line (i.e., area I and II), the system is locally stable.

Theorem 4 (see the line with marker) further divides the area below the stability criterion into two parts. For area I, not only the whole system is stable, but each group of flows is stable. For area II, even though the whole system is stable, the group 2 flows with longer RTT are unstable. In other words, for a given capacity C, if ρ_1 falls into the area II, flows in group 2 will not converge when all the flows in group 1 are replaced by their equilibrium values (constants). But, if we replace flows in group 2 with their equilibrium points, then flows in group 1 are guaranteed to converge and thus stable. When ρ_1 is large and belongs to area III, the entire system is unstable, i.e., none of the flow converges. In the next section we provide simulation results to verify the observations made here.

3.7 Simulation Results

In this section, we present *ns*-2 [3] simulation results to show that (i) for a TCP-newReno/RED system with many flows, if it is stable, then it will converge to the unique equilibrium point given by (3.6) and (3.7); (ii) if the system is locally stable and Δq is not too small, it is then also globally stable.

Table 3.2: Simulation Parameters

Parameter	Value	Parameter	Value
# of Flows	1000	mean_pktsize_	1500 Bytes
Router Capacity	100M bps	thres_	1000 pkt
Buffer	200K pkt	maxthres_	9000 pkt
Local Speed	10 M bps	linterm_	12.5
Bottleneck Delay	100 ms	q_weight_	0.002
Local Delay	100 ms	setbit_	True
Max Window Size	100	gentle_	True

3.7.1 Homogeneous RTTs

We simulate the system as in Figure 3.3-(a) with parameters listed in Table 3.2. Figures 3.13 (a) and (b) show the normalized queue-length and the average window size, respectively, where the number of flows ranges from 10 to 1000. The lower graphs display magnified versions of the upper graphs over $[400, 500]$ (fine time scale).

By solving (3.13) with the parameters given in Table 3.2, we have $C \approx 6.70$. Then, from (3.5)–(3.7), we obtain $W(\infty) = 6.70$ and $Q(\infty) = 4.96$. Figure 3.13 shows that $W(t)$ converges to 6.71 and normalized queue-length converges to 4.94, which are in good agreement with the analytical results. Figure 3.13 also shows that as the number of flows increases, the variation becomes smaller, indicating that our model can effectively capture the system behavior especially when the number of flows in the system is large.

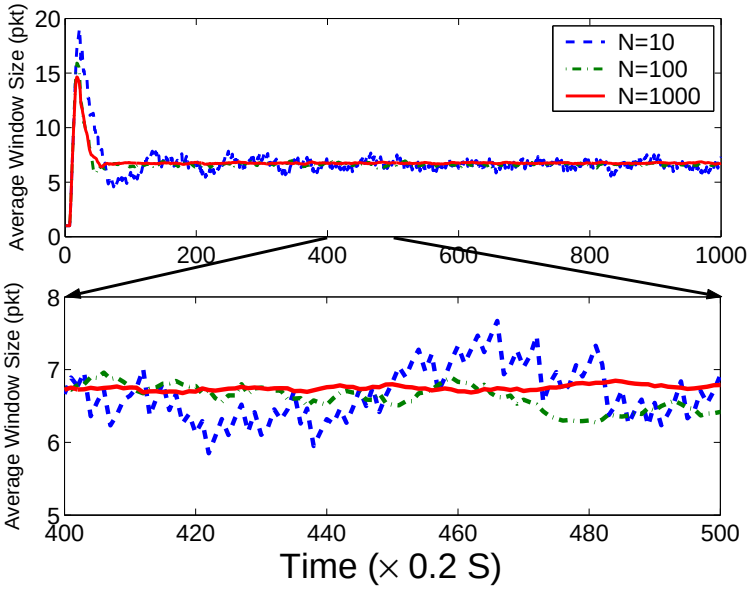
Next, we provide experimental results for global stability. We use the same simulation parameters in Table 3.2, but different initial window size $W(0)$ and normalized queue size $Q(0)$. It is easy to verify that the configuration is locally stable by Theorem 1 and $\Delta q = q_2 - Q(\infty) = 4.06$. Figure 3.14 shows that the average window size and the normalized queue-length all converges to their equilibrium points, regardless of their initial window sizes tested in our simulations. We have also run many other simulations using different initial queue-length and obtain similar results.

3.7.2 Heterogeneous RTTs

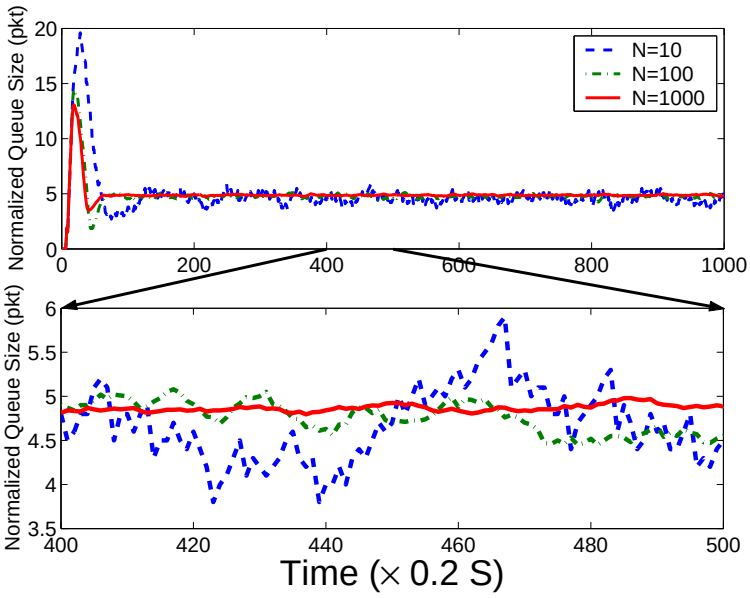
We here consider two groups of flows. The first group consists of 20 TCP flows with RTT 320 ms and the second group has 200 TCP flows with RTT 40 ms. They all pass through a bottleneck router with capacity 200 kbps $\times(20 + 200) = 44$ Mbps.

Figure 3.15-(a) shows that the mixed two groups of flows yield the stable queue dynamics. From Theorem 4, we know that the first group is unstable and the second group is stable. Thus, we conduct the follow two sets of simulations: (i) only the first group of flows passes through the bottleneck router with 'scaled-down' capacity 200 kbps $\times 20 = 4$ Mbps; (ii) only the second group of flows pass through the bottleneck router with capacity proportional to the number of flows, 200 kbps $\times 200 = 40$ Mbps. [4] We plot the corresponding queue dynamics of each group in Figures 3.15-(b) and (c), respectively. We can see that

[4] The use of 'scaled-down' capacity is equivalent to replacing the other group of flows by their average window sizes (constant) under the original capacity 200 kbps $\times$220=44 Mbps.



(a) Average Window Size.



(b) Normalized queue-length.

Figure 3.13: The average window size and normalized queue-length under different number of flows N.

the first group displays wild fluctuation (not stable), and the second group shows much less fluctuation (stable). We then increase the number of flows in the second group from 200 to 1000 while the number of flows in the first group remains the same. Consequently, ψ_1

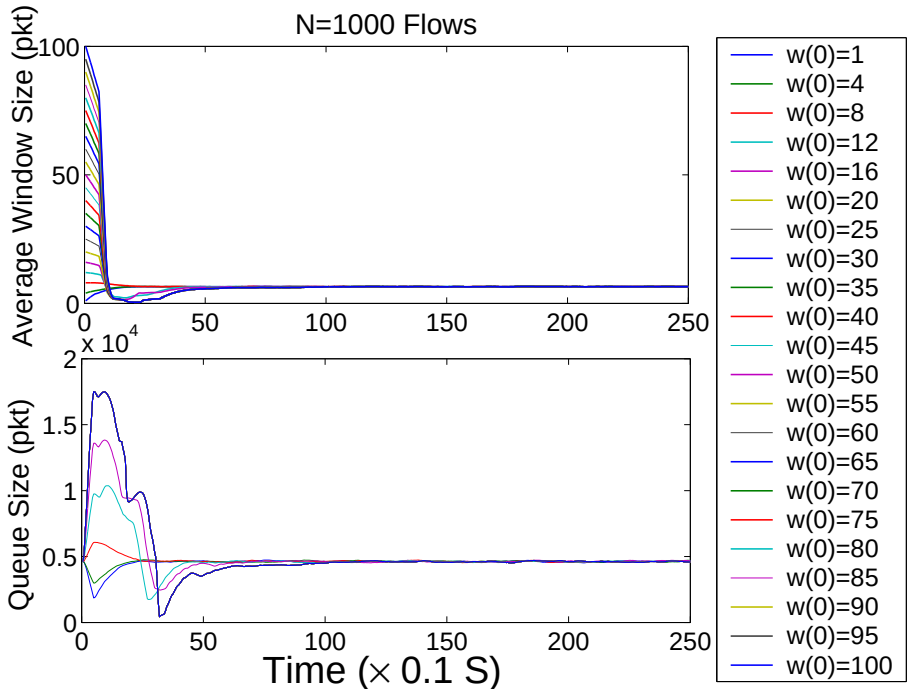
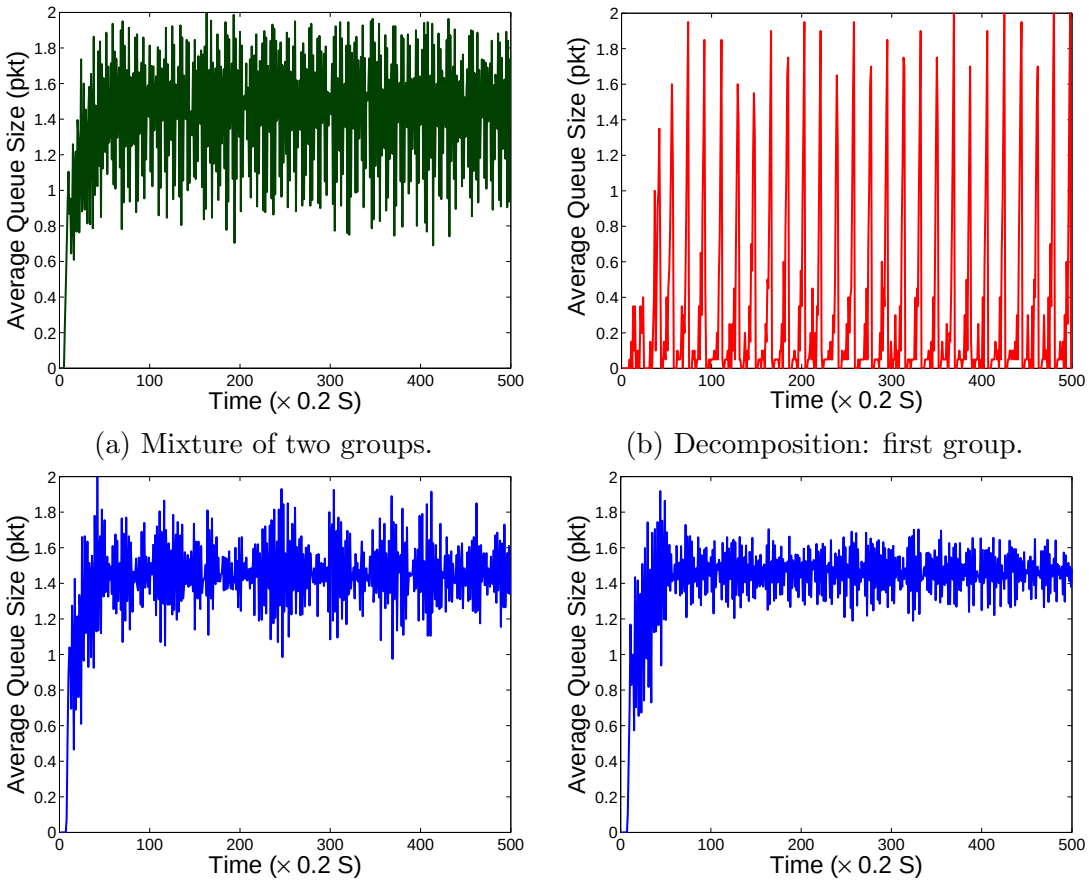


Figure 3.14: The average window size and instantaneous queue-length $Q^N(t)$ with different initial window size $W(0)$ and under fixed $Q(0) = 5$.

decreases and ψ_2 increases, which means the influence of the first group is diluted by the second group and the second group (stable) dominates the system stability in such a setting as shown in Figure 3.15-(d).

3.8 Conclusion

We investigate the local and global stability of a TCP-newReno system with RED queue under many flows' regime. The existing local stability literature is designed for TCP-Reno not for TCP-newReno. As to the global stability, traditional approaches rely on constructing a suitable Lyapunov function. However, for a typical TCP-newReno/RED system with many flows, it is generally infeasible to find the Lyapunov function as the entire system dynamics are too complicated to be tracked. In this paper, we present a normalized model for the TCP-newReno/RED with many flows. This normalized model is able to capture the essential system dynamics, and at the same time simple enough to allow fast numerical analysis. Based on the normalized model, we derive the local stability criterion



(a) Mixture of two groups.

(b) Decomposition: first group.

(c) Decomposition: second group.

(d) Mixture of two groups: with the second group of 1000 flows.

Figure 3.15: The decomposition for heterogenous RTTs.

and show how to make a locally stable TCP-newReno/RED system also globally stable by properly choosing some of its parameters. Our extensive numerical analysis also suggests that in most cases, any locally stable TCP/RED system is also globally stable, and this trend becomes clearer as the number of flows increases. Finally, we extend our model to the case of heterogeneous RTTs.

Chapter 4

TCP Performance under Aggressive Scale

In this chapter, we investigate TCP performance under aggressive marking scale. We note that this work joints with my advisor Dr. Do Young Eun, and for the purpose of completeness, we include some contributions of Dr. Eun in this chapter, and for the detailed information, please refer to the corresponding technical report [50].

4.1 Preliminaries

4.1.1 Scaling Buffers and AQMs inside a Network

In TCP congestion control, one of the long-held rules-of-thumb in network design is that the buffer size at bottleneck links should be proportional to the bandwidth-delay product [189, 180]. In a large network setting where there are N flows with link capacity NC, this rule-of-thumb suggests $O(N)$ for buffer sizing. Similarly, most results available in the literature require the marking function $p^N(x) \in [0, 1]$, or the probability of packet marking when the queue-length is x, be scaled *linearly*, i.e., for all N and x, $p^N(Nx) = p(x)$ for some function p, where the superscript 'N' in the marking function $p^N(x)$ means that

there are N flows with capacity NC. This linear scaling for buffer size as well as any AQM policy has thus been *de facto* scaling in most existing results for large systems [44, 139, 186, 183, 180, 178], as it is necessary to make the system at least linearly stable [46, 178, 139].

Quite recently, however, under the drop-tail AQM policy with many flows, it is shown [17] that one can do much better than this traditional rule-of-thumb for buffer sizing. *Empirical observation* reveals that when N is very large, each user's window size W_i ($i = 1, 2, \ldots, N$) becomes independent. By appealing to the Central Limit Theorem, their sum will behave like a Gaussian random variable with mean equal to the size of the 'pipe' (NC $\times$ RTT) and with standard deviation $O(\sqrt{N})$. Hence, by choosing the buffer size on the order of $\sqrt{N}$ to absorb typical fluctuations, one can still achieve high utilization and thus save huge cost for buffers implemented in high-speed core routers.

This new observation based on the independence among different flows indicates that the long-held rule-of-thumb of $O(N)$ buffer sizing is no longer necessary in large systems. In this regard, an open question is: what is the *ultimate* rule-of-thumb? Can we do even better than $O(\sqrt{N})$, i.e., can we put buffers of size $O(N^\alpha)$ with $\alpha \in (0, 0.5)$ instead, without hurting the performance? To the best of our knowledge there is no result available in the literature as to whether such scaling would work.

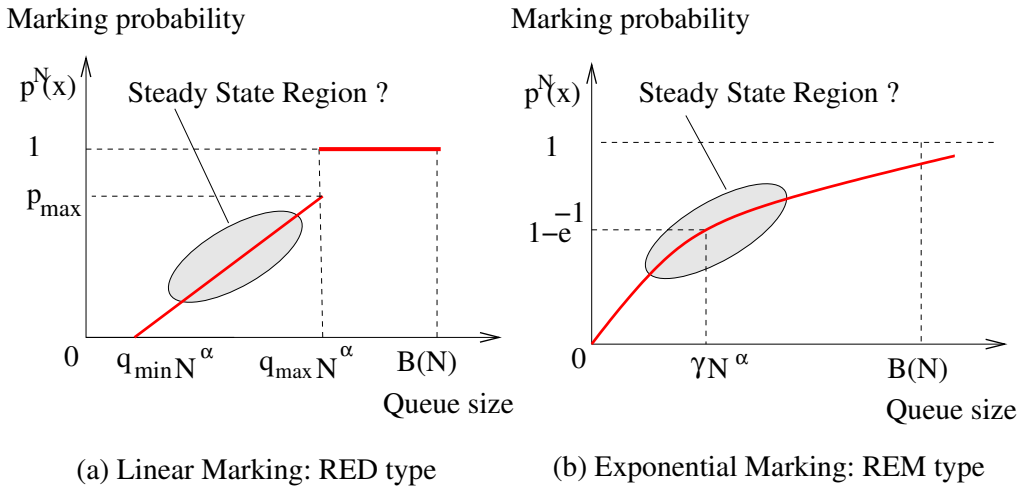
Suppose we adopt such a scaling $O(N^\alpha)$ ($\alpha < 0.5$) for buffer size. Since the natural fluctuation of the sum of N random variables (window sizes of N users) is $O(\sqrt{N})$, a buffer of size of $O(N^\alpha)$ can no longer absorb the $O(\sqrt{N})$ fluctuation once it becomes present. In other words, even if there exists independence among all the flows, the system would collapse and give poor performance under the scale of $O(N^\alpha)$ ($0 < \alpha < 0.5$).

4.1.2 Aggressive Packet Marking

In this paper we explore any possibility of using buffer sizes and AQM schemes with $O(N^\alpha)$ scaling, where $\alpha < 0.5$. By the scale of N^α, we mean that the buffer size is on the order of N^α ($B(N) = O(N^\alpha)$) and there exists a function p such that $p^N(N^\alpha x) = p(x)$ for all N and $x \geq 0$ (see Figure 4.1). Note that this can be interpreted as a more *aggressive* marking than the case of $\alpha = 1$ (linear scale) since the marking probability is much higher for the same queue-length.[1] We note that difference in marking scales simply means different parameter set-up at the routers, which can be easily configured as desired. Moreover, the

[1] Our scale is even more aggressive than the case of $\alpha = 0.5$.

scaling of marking functions, or AQM schemes in general, is directly related to the issues of network design over a long time scale. For example, if the number of connections (or customers) were to increase by 10 times, then a natural question to ask is: what percentages of additional capacities or buffer sizes are needed at routers to provide the same level of performance to each user? Under the linear scale ($\alpha = 1$), we would have to increase by 10 times the buffer size as well as all the parameters associated with the AQM. If our aggressive scale is to be used, the buffer size and the AQM parameters should be increased very little (e.g., $10^{0.2} = 1.6$ when $\alpha = 0.2$).



(a) Linear Marking: RED type (b) Exponential Marking: REM type

Figure 4.1: Examples of marking functions with aggressive marking scale of N^α.

Figure 4.1 shows some examples of marking functions with the scale of N^α. Note that all the thresholds and the buffer size are on the order of N^α. For the example in Figure 4.1 (a), if the queue-length becomes larger than the upper threshold $q_{max}N^\alpha$, then all the incoming packets will be marked and accordingly, all the flows will reduce their window sizes. This makes the queue-length decrease in the next RTT. Similarly, if the queue-length becomes smaller than the lower threshold $q_{min}N^\alpha$, then no packet is marked and all the flows increase the window sizes, making the queue-length larger later on. Thus, we see that *if there exists any 'steady-state'* [2] *for the queue-length, it must be between these two thresholds, both of which are* $O(N^\alpha)$. Suppose this conjecture is true for any $0 < \alpha < 0.5$ for large N. Then, clearly, the queue-length still increases as N becomes larger. In other

[2]Here, the steady-state should be interpreted in a distributional sense, not as a constant queue-length. In such a steady-state, the queue-length will be $O(N^\alpha)$ with high probability.

words, the queue will never be empty and we will have almost 100% link utilization. On the other hand, the packet delay in the queue ($Q^N(t)/NC$) will be $O(N^{\alpha-1})$, which decreases to zero for large N for any choice of $0 < \alpha < 0.5$. Further, we also expect that, if the buffer size is chosen slightly larger than $O(N^\alpha)$, the packet loss will be kept very small. Since $0 < \alpha < 0.5$, this would imply that we can do much better than $O(\sqrt{N})$ buffer sizing [17] while maintaining all the good performance measures (high link utilization, negligible delay, and low loss).

However, since the traditional stability theory based on the fluid modeling usually focuses on the 'normalized' system with averaged parameters over RTTs, it is not surprising to see that they all predict that this scaling with $O(N^\alpha)$ ($\alpha < 0.5$) will make the system always unstable. For example, in case of TCP/RED with ECN marking and when there are N flows with link capacity NC, it turns out that the criterion for linear stability (obtained from the Nyquist criterion) becomes $O(N^{1-\alpha}) < \zeta$, where ζ is a system-independent constant [139]. Thus, when N is large, the system is always linearly *unstable* under the fluid-based modeling, and therefore *not desirable* as it may increase jitters in source rate and delay, and may lead to under-utilization of link capacities if queues jump between empty and full [139].

4.1.3 A Closer Look at the Queue Dynamics

In this section we present an example with *ns*-2 simulation results showing that the system remains largely intact even under the aggressive scales. Figure 4.2 shows the queue-length fluctuation of a TCP/RED system with 1000 flows under different scalings for AQM as well as for the buffer size. We set the two-way propagation delays for 1000 flows to be *i.i.d.* and uniformly distributed over $[120, 280]$ ms with mean 200 ms. In Figure 4.2(a), we set the lower threshold as 1000 and the upper threshold 101×1000 as in [186]. This corresponds to a linear scale and the system is stable according to the criterion in [139]. In Figure 4.2(b), we set both the thresholds very small (4 and 24), which corresponds to the aggressive packet marking. Also, we choose the buffer size to be 25, much smaller than any other existing practice. (We note that the guideline in [17] using $O(\sqrt{N})$ scaling is about 150 for buffer size.) Obviously, the system becomes unstable and displays 'wild' fluctuation as shown in Figure 4.2(b). However, if we plot these two figures on a same scale, the results reveal the opposite. As in Figure 4.2(c), the queue-length for the aggressive marking is close

to zero and looks almost constant, while we can still see some 'noise' in the queue-length for the linear marking. In particular, although the system in Figure 4.2(b) is unstable and displays fast fluctuation in queue-length, that fluctuation is also 'controlled' and 'trapped' in the sense that the queue-length never goes above 24 and rarely hits empty, while the queue-length stays around 3500 for the stable system as in Figure 4.2(a).

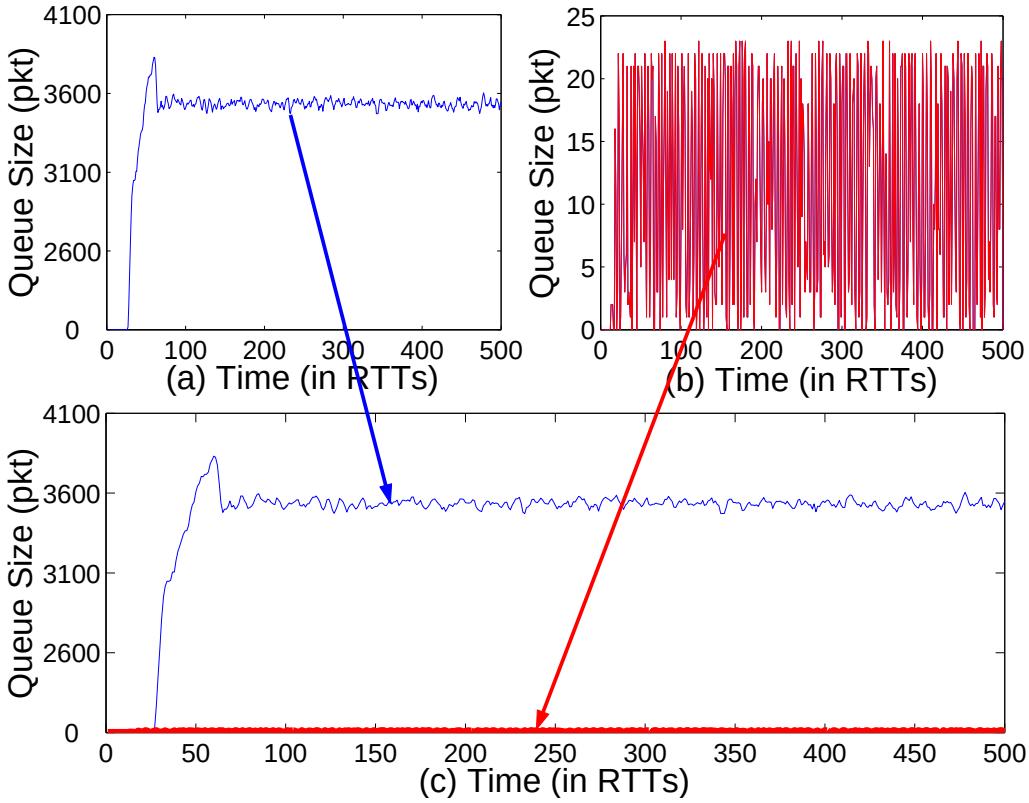


Figure 4.2: Queue-length fluctuation using *ns*-2: (a) stable queue; (b) unstable queue with aggressive marking; (c) comparison of (a) and (b) over the same y-axis.

Table 4.1: Performance Comparison of the queues in Figure 3.5.

Queue	Avg. delay	Jitter	Util.	Loss Ratio
Stable (a)	154 ms	1.94 ms	1.0000	0.0000
Unstable (b)	0.46 ms	0.37 ms	0.9742	0.0233

Table 4.1 shows the four performance metrics for the systems in Figure 4.2. We observe that the average delay and delay-jitter (standard deviation) for the case (b) in Figure 4.2 are much smaller than for the case (a) at the cost of slight reductions in the link

utilization and the packet loss ratio. But, these are still far from the 'poor performance' as predicted from the traditional stability theory. In fact, our extensive simulation results in Section 3.7 show that this reduction in performance metrics becomes negligible (approaching to zero) as the system size N increases. We see that any traditional stability theory or even the arguments based on the Central Limit Theorem cannot explain the observed behavior. We are thus led to believe that there must be some other ways to explain the 'good performance' under the aggressive marking scale.

To address this issue, in this paper, we take the packet-level dynamics into account for the congestion control in large-bandwidth networks. In what follows, we will show that there exist two different sources of randomness by closely looking at the system dynamics at packet level over fine time scales, and we will develop a doubly-stochastic model to effectively capture such dynamics.

4.2 Main Results

By CLT argument of Stanford scheme [17],

$$\sum_{i=1}^{N} W_i = N\mathbb{E}\{W_i\} + o(\sqrt{N})$$

Further, our results show that even for the queue size of $o(N^\alpha), 0 < \alpha < 0.5$, similar performance as that in [17] is obtained. Intuitively, we can invoke the same as CLT argument as displayed in Figure 4.4.

If the queue-length is $o(\sqrt{N})$, the queue would have high utilization with small loss ratio. And if the queue size is $o(N^\alpha), \alpha < 0.5$, the packet loss ratio will be almost 50%, which means most the flows will subject to half reduction of window size and the resulting link utilization will be poor. However, this does not happen in real world. Our observation is that even for the queue-size of $o(N^\alpha), \alpha < 0.5$, the queue still keeps the high link utilization, small loss, small queueing delay and jitter.

The reason is that there exist randomness in packet arrival instance, which help queue to get the earlier information to the

We want to show that under the aggressive scale, without arrival randomness, the performance is poor; and with arrival randomness, the performance is good enough.

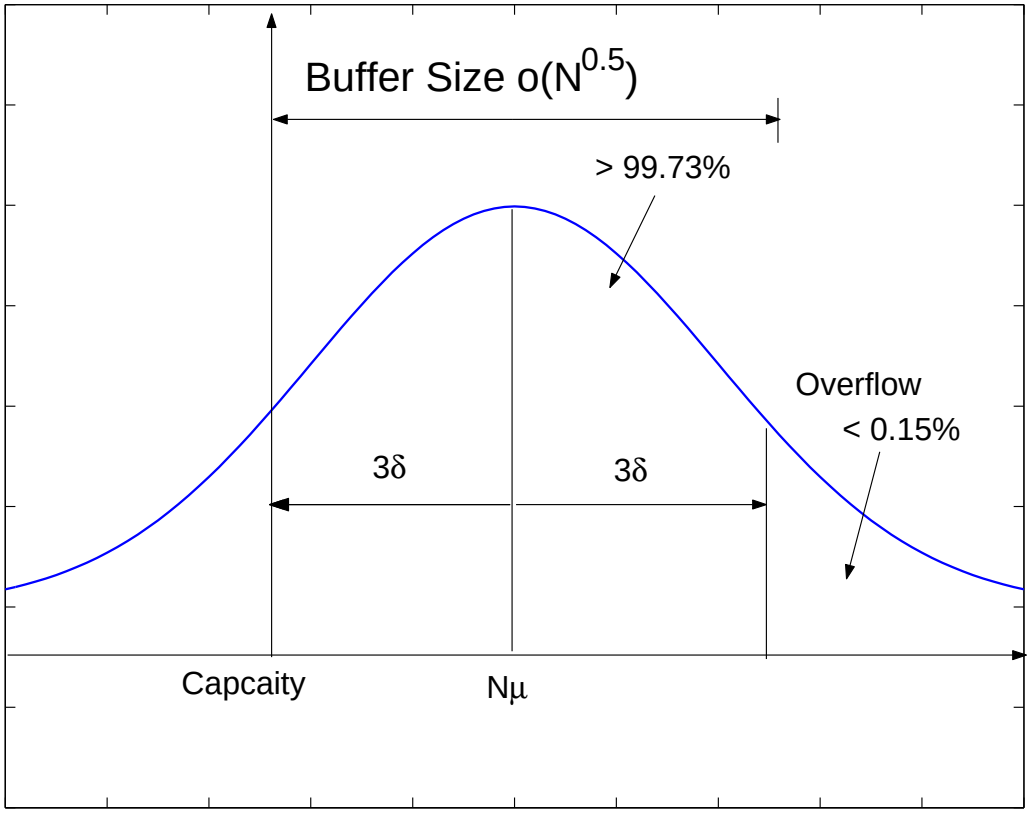


Figure 4.3: The CLT argument in [17] for $o(N^{0.5})$.

Thus, we can reach the conclusion that the randomness in packet arrival helps the system performance in certain situations especially for aggressive scale.

We note that under linear scale $o(N)$, deterministic packet arrival explain the system dynamics well by using fluid model [187, 138]. However, under the aggressive scale $o(N^{\alpha}), 0 < \alpha \leq 0.5$, the randomness in packet arrival cannot be ignored and plays an important role in help router regulate the queue dynamics nicely in the sense that the link-utilization is high; the packet loss ratio is small; the queuing delay and jitter are small.

4.3 Numerical Results

In this section, we provide numerical results using *ns*-2 [3] to validate our results in Section 4.2 under a variety of network configurations. We perform a set of simulations under

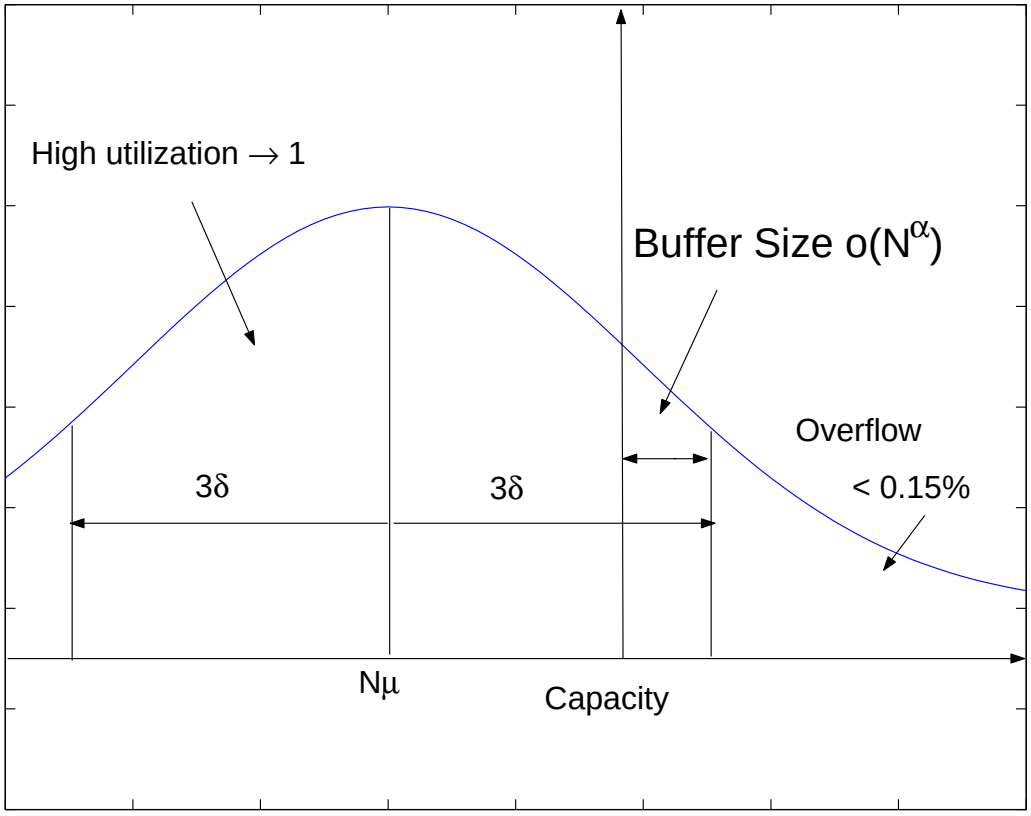


Figure 4.4: The CLT argument for our scheme in [51] for $o(N^{\alpha}), 0 < \alpha < 0.5$.

the following cases: (i) long-lived flows with homogeneous RTTs, (ii) long-lived flows with heterogeneous RTTs, (iii) a mixture of long-lived and short-lived flows with heterogeneous RTTs, and (iv) long-lived flows with multiple links.

Although our theoretical results assume a purely queue-based AQM scheme, we show that our results can also be applied to other popular AQM schemes such as REM [18], PI [36], and even Drop-Tail (DT). Throughout the simulations, we consider the following four performance metrics:

- Average queueing delay: we measure the queueing delay of all the packets arriving to the queue of our interest over a long time and take the average.
- Delay jitter: we calculate the standard deviation from all the measured queueing delays.

- Link utilization: this is the ratio between the total amount of data transmitted and the link capacity.
- Packet loss ratio: this is given by the total number of dropped packets at the queue over the total number of packets coming into the queue.

4.3.1 AQM Configurations

We consider 5 different AQM schemes for our simulation. Table 4.2 summarizes the parameters of each AQM scheme used in our simulation.

Table 4.2: AQM parameters for *ns*-2.

AQM	Scaled Parameters
RED	$min_{th} = 2N^{\alpha}$, $max_{th} = 10N^{\alpha}$ $P_{max} = 0.2$, Buffer Size = $12N^{\alpha}$
EXP	$\gamma = -10/\ln(1 - P_{max})$, Buffer Size = $12N^{\alpha}$
REM	pbo_ = $2N^{\alpha}$, Buffer Size = $12N^{\alpha}$
PI	$q_{ref} = 2N^{\alpha}$, Buffer Size = $12N^{\alpha}$
DT	Buffer Size = $12N^{\alpha}$

For RED, the minimum and the maximum thresholds ($q_{min}N^{\alpha}$ and $q_{max}N^{\alpha}$ in Figure 4.1(a)) are $2 \times N^{\alpha}$ and $10 \times N^{\alpha}$, respectively. The maximum dropping probability, P_{max}, is set to 0.2 and the queue averaging factor, q_weight_, is set to 1. (We have also run simulations with other queue averaging factors, e.g., q_weight_=0.002, and obtained similar results.) As to EXP, we use a queue-based AQM with an exponential marking profile as in Figure 4.1(b). In order to make a fair comparison with RED, we choose $\gamma = -\frac{10}{\ln(1-P_{max})}$ (as in Figure 4.1(b)) such that $p^N(max_{th}) = p^N(10N^{\alpha}) = P_{max}$. For REM and PI, we set the target queue size ("pbo_" of REM and "q_{ref}" of PI) as $2 \times N^{\alpha}$. For other parameters of REM and PI, we use the default values as specified in *ns*-2. In any case, the physical buffer size is given by $B(N) = 12 \times N^{\alpha}$.

4.3.2 Homogeneous RTTs

We first consider a simple topology with a single bottleneck link shared by N long-lived TCP flows. we set the sum of two-way propagation delays to be 200 ms for all N

flows. This gives $T = 200$ ms if the queueing delay is negligible. Each packet size is fixed to 500 bytes and the link capacity is $N \times 100$ kbps. Thus, we have $C = 25$ packets/sec, and the size of the 'pipe' per flow becomes $CT = 25 \times 0.2 = 5$ packets.

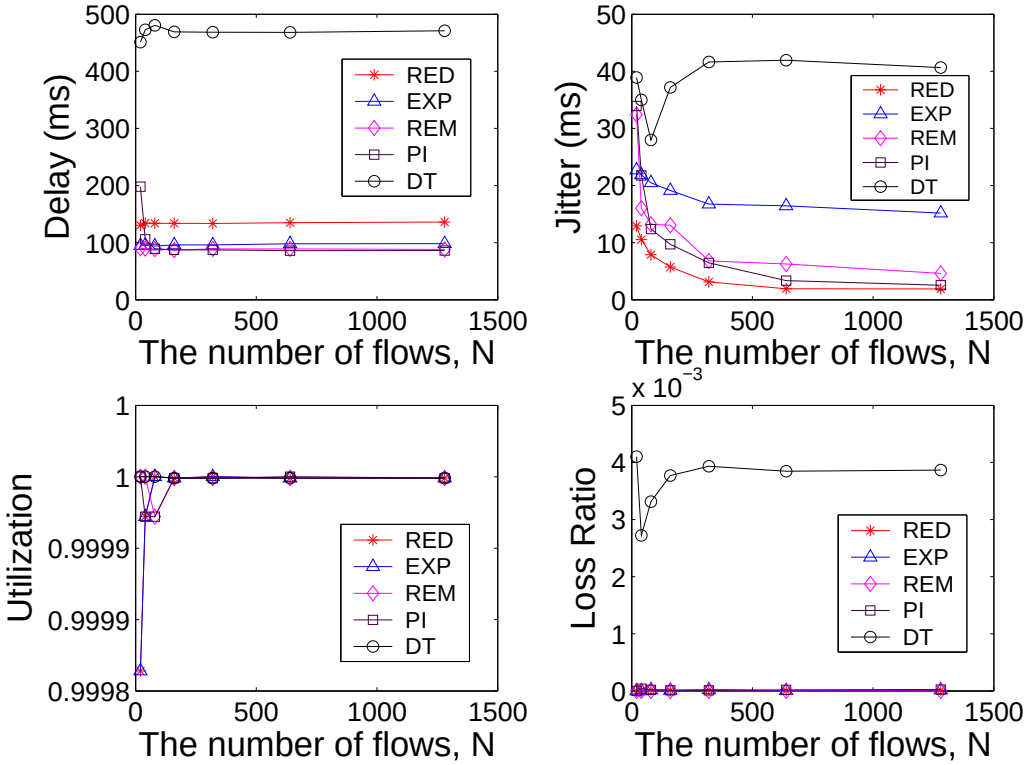


Figure 4.5: Homogeneous RTTs: Performance metrics with increasing number of flows (N) for fixed $\alpha = 1$.

Figure 4.5 shows the four performance metrics as the number of long-lived flows (N) increases under the linear scale for AQM schemes, i.e., $\alpha = 1$. As expected, the queueing delay is almost constant for each AQM scheme as the queue is scaled up linearly. Note that, as N grows, the delay jitters of all AQM schemes except DT quickly decrease and becomes almost constant, since packet arrivals to the queue tends to be stationary and well-behaved with more flows. Clearly, the drop-tail policy gives the worst performance among those AQMs considered. We also see that all the AQM schemes yield almost 100% link utilization and very low packet loss ratio as is the case for any 'stable' TCP/AQM system. However, this comes at a cost: large buffer size and queueing delay. From our choice of buffer size $12N^{\alpha}$ with $\alpha = 1$, the required buffer size for $N = 1000$ becomes 12000 packets, which is larger than the bandwidth-delay product NCT (= 5000 packets).

Also, the queueing delay is by no means negligible and almost comparable to the sum of propagation delays (200 ms).

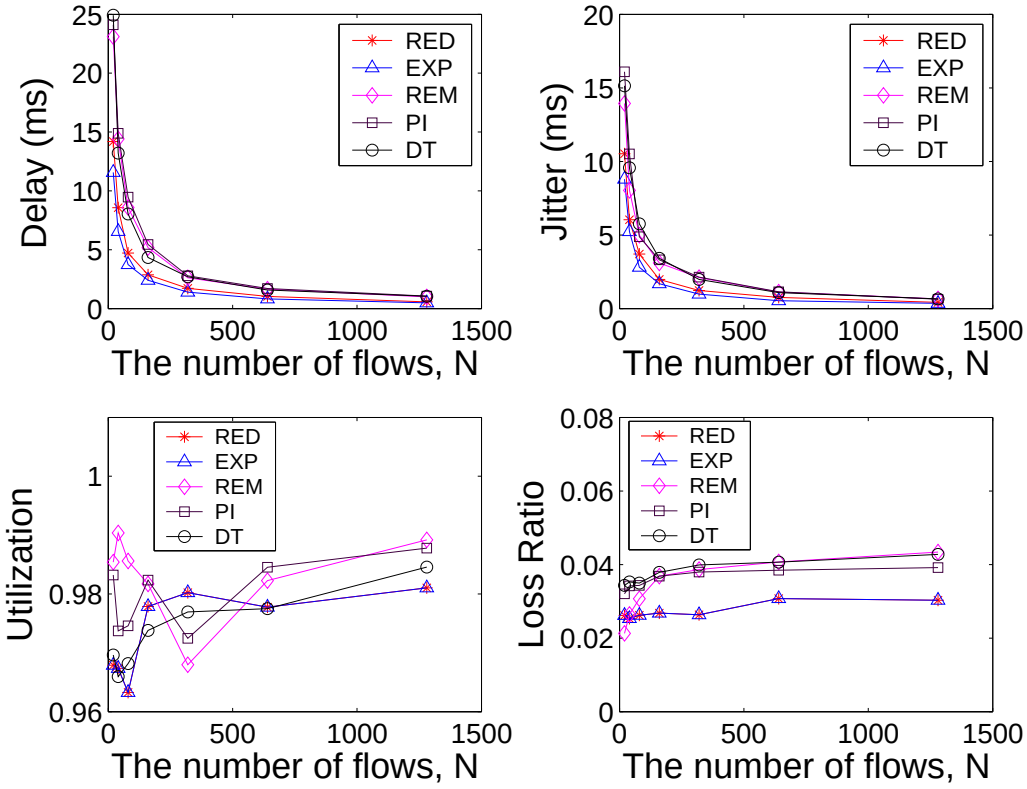


Figure 4.6: Homogeneous RTTs: Performance metrics with increasing number of flows (N) for fixed $\alpha = 0.2$.

Figure 4.6 shows the performance metrics as the number of flows N increases, now with the aggressive scale, $\alpha = 0.2$. All the other parameters are the same as in Figure 4.5. Note that for all the AQM schemes, the queuing delay and delay-jitter decrease sharply to zero with the increase of N. This is in stark contrast to the case of $\alpha = 1$, where we have almost constant delay and jitter for each AQM scheme. Further, we observe that the aggressive scale for packet marking eliminates the difference in delays and jitters for different AQM schemes, and makes them behave virtually identical. For all AQM schemes, the link utilization tends to increase in N and becomes at least 98% in the worst case when N is large enough.

As to the packet loss ratio, RED and EXP yield slightly better performance than the other three schemes, and all the AQM schemes do not seem to display strongly decreasing

trend as expected. This may be because the convergence rate for packet loss ratio is very slow (compared to delay and delay jitter). Another reason could be that, with homogeneous RTTs and a single link, the packet arrivals are not as much random as required in our theory. In fact, in this particular case where we have a single link shared by N identical flows, the only source of randomness for *arrival instants*[3] is the time-varying queue-length (or queueing delay), which makes the actual RTTs also time-varying. However, as we see in Figure 4.6, the queueing delay itself becomes very small (almost zero) for large N, and thus there is not much randomness in packet arrivals (in time) to the queue for large N.

It should be pointed out that the case of $\alpha = 0.2$ requires very small amount of buffer space. Indeed, the buffer size for all the AQM schemes in Figure 4.6 is $B(N) = 12N^{\alpha}$, and for $N = 1000$, this gives only $12 \times 1000^{0.2} \approx 48$ packets, which is several orders of magnitude smaller than 12000 packets for $\alpha = 1$. Note that it is even smaller than the $O(\sqrt{N})$ buffer sizing [17], where the authors proposed $B(N) \approx NCT/\sqrt{N} = 5000/\sqrt{1000} \approx 158$ packets for high link utilization. Since $\alpha \in (0, 1/2)$ can be chosen arbitrary, the buffer size can be much smaller under our aggressive scale than $O(\sqrt{N})$ as long as there are many flows in the system, and this difference becomes huge for larger values of N (i.e., $B(N) \ll O(\sqrt{N})$).

4.3.3 Heterogeneous RTTs

In this section, we consider the same topology as in Section 4.3.2, but now the sum of two-way propagation delays for N flows are *i.i.d.* and uniformly distributed over $[120, 280]$ ms (with mean 200 ms). The simulation results for the linear scale ($\alpha = 1$) generate similar trends as those of homogeneous RTTs in Figure 4.5. Due to the space limit, we do not include them here.

Figure 4.8 shows that under the aggressive scale $\alpha = 0.2$, the delay and delay-jitter still sharply decrease as the number of flows increases. In addition, the utilization and the packet loss ratio now show clear trend with the number of flows N, as predicted by Theorems. This is because heterogeneous RTTs make the packet arrivals to the queue more random even for a single link case, which helps our theory kick in faster.

We observe that there is some tradeoffs between RED, EXP, and other three AQM schemes. RED and EXP have the lowest packet loss ratio with the lowest link utilization.

[3]Note that the *amount* of packet arrivals is still random due to the random packet marking.

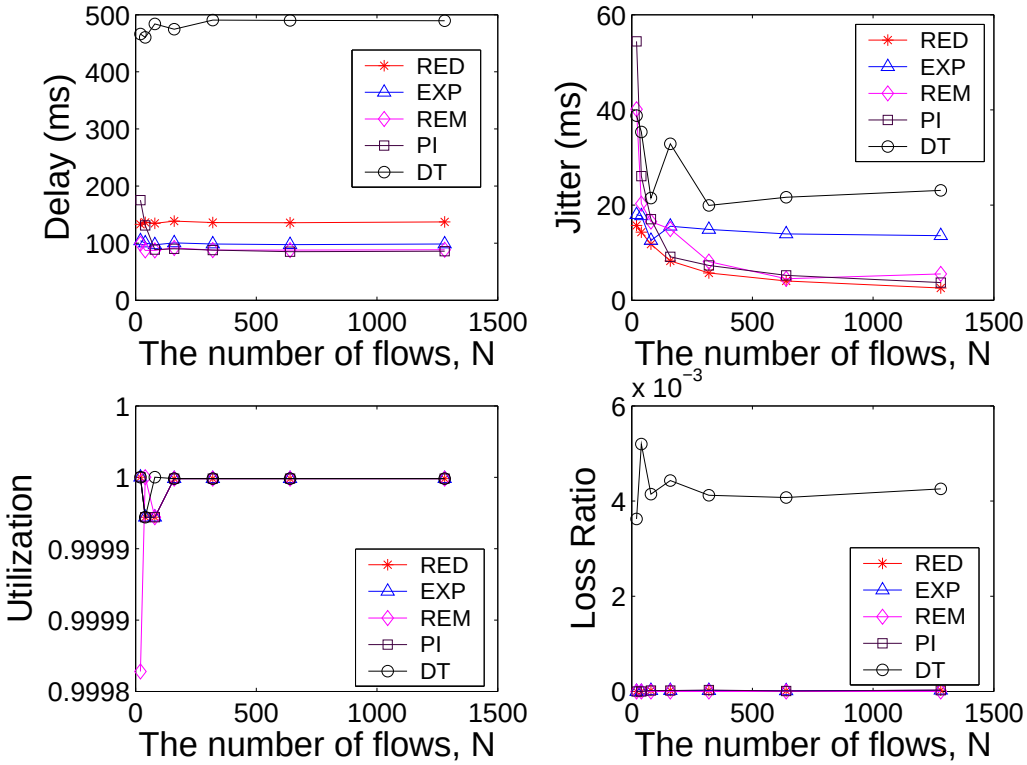


Figure 4.7: Heterogeneous RTT: Performance metrics with increasing number of flows, N for fixed $\alpha = 1$.

Using different configurations for RED/EXP, it is possible to improve the link utilization at the expense of higher loss ratio. We can see that the difference between RED and EXP (also the difference among REM, PI, and DT) becomes indistinguishable under the aggressive scale, which implies that the scale kills the difference among AQM schemes in some sense and becomes the dominant factor in determining the performances.

4.3.4 Mixture of Long-lived and Short-lived Flows

It is well-known that file transfers account for the majority of the traffic over the Internet. Typical file sizes follow Pareto distribution as reported in [42, 109], which means the majority of the file transfers has a short life-time and few have a very long life-time. In general, short-lived flows finish transmission within the slow-start phase and never reach the congestion control phase, thus do not react to the congestion signal as the long-lived flows do.

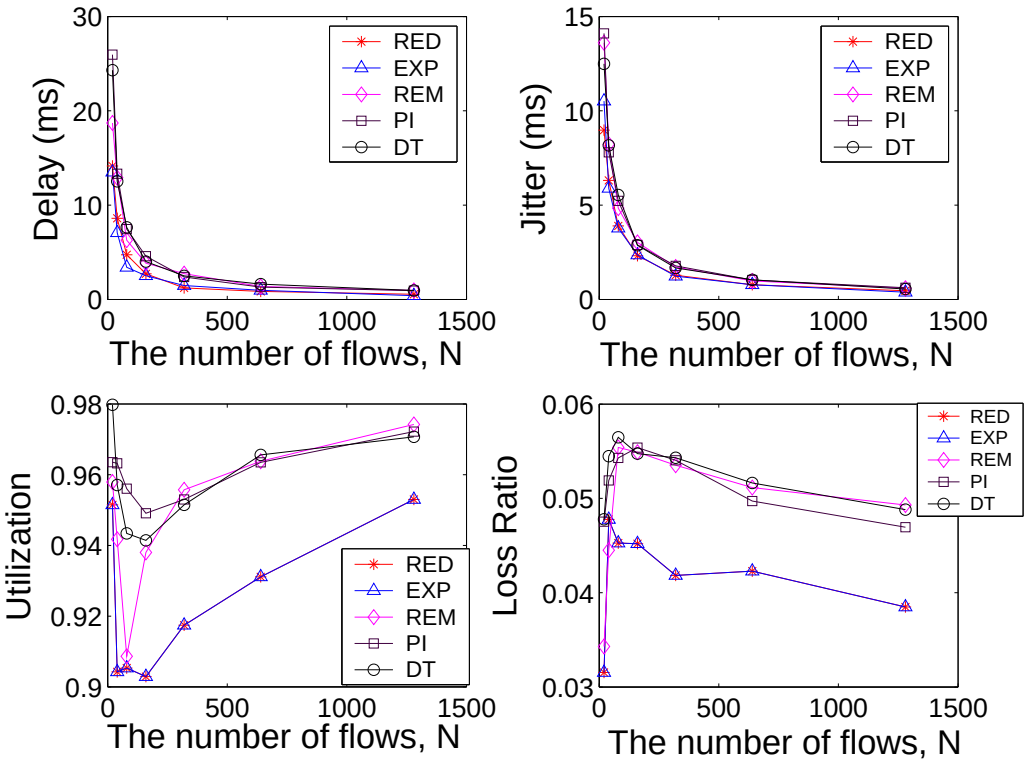


Figure 4.8: Heterogeneous RTTs: Performance metrics with increasing number of flows (N) for fixed $\alpha = 0.2$.

In this section, we simulate TCP/AQM systems with a mixture of long-lived and short-lived flows under heterogeneous RTTs. We generate the traffic as follows. For each persistence TCP flow, we add a number of flows to share the link capacity. The file size for each added flow is characterized by Pareto distribution with shape parameter 1.5 [4] with the minimum file size $x_{min} = 1.667$ kbytes. So, the distribution of the file size X of each added flow is given by

$$\mathbb{P}\{X > x\} = \left(\frac{x}{x_{min}}\right)^{-1.5}, \quad x \geq x_{min}. \tag{4.1}$$

This gives the mean file size of 5 kbytes (= 10 packets) and a median of 2.645 kbytes, which implies that most of the flows end within 10 packets ($\mathbb{P}\{X \leq 5\text{k}\} = 0.81$) and nearly half of them end within 6 packets. During the simulation, for each persistent TCP flow, we add about 300 random flows whose file sizes are distributed according to (4.1) (Most of them are short-lived ones.), and those random flows as a whole consume about 50% of the total

[4] The shape parameter is between 1 and 2 [42].

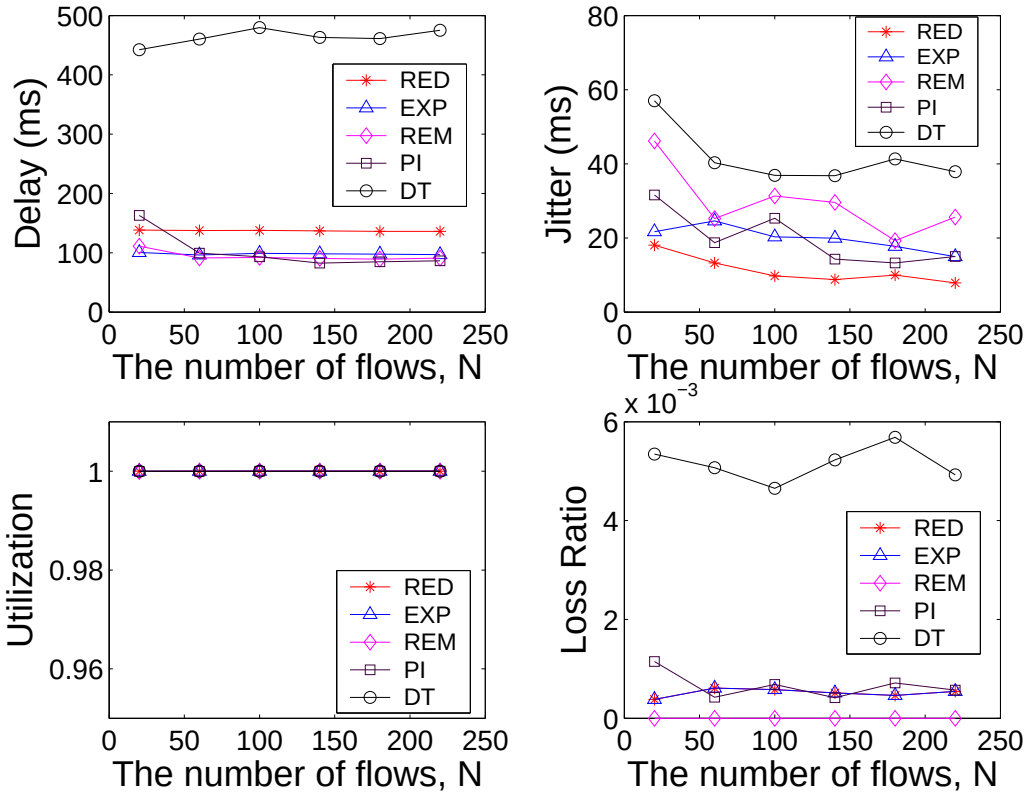


Figure 4.9: Webmice traffic: Performance metrics with increasing number of persistent flows, N for fixed $\alpha = 1$.

bandwidth of the link. We have also examined other situation where the random flows account for 20% of the total bandwidth, and obtained similar results.

Figure 4.10 plots the performance metrics under the aggressive scale $\alpha = 0.2$. Due to the space limit, we omit results with the linear scale ($\alpha = 1$), where we obtained similar observations. In Figure 4.10, both delay and jitter decrease as the number of persistent flows increases. For the link utilization and the packet loss ratio, there is no definite trend in any of AQM schemes considered. This is mainly because the number of long-lived flows is not large enough (around 250 at most) and the system is still in 'transient' stage, compared to the system in Figure 4.8 where we have more than 1000 persistent flows and the system is in the steady-state. Note that with the existence of short-lived flows, we were unable to simulate 1000 persistent flows, since in this case we would have to add about several hundreds of thousands of random flows and this was not feasible in our current simulator setup. However, note also that the link utilization is still around 95% and the loss ratio is

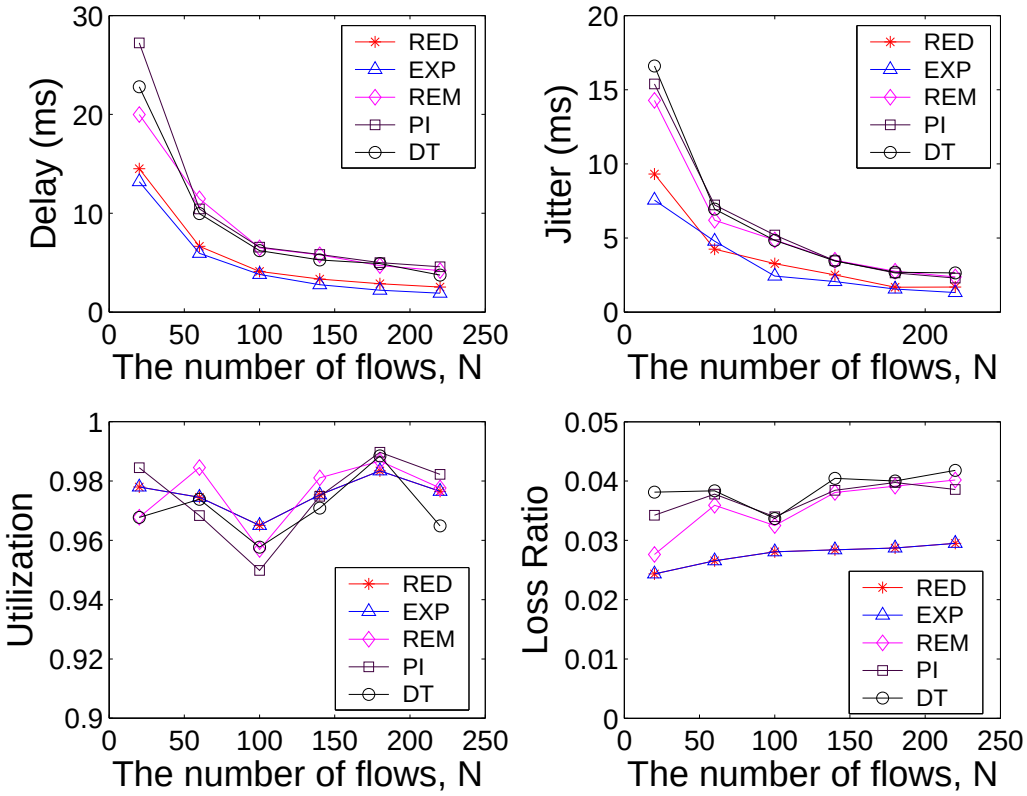


Figure 4.10: A mixture of short- and long-lived flows: Performance metrics for $\alpha = 0.2$ as the number of persistent flows (N) increases.

less than 4% even in the worst case.

4.3.5 Multiple Links with Heterogeneous RTTs

Up until now, we have simulated various network scenarios with a single link. In this section, we present simulations results using multiple links in the network.

Figure 4.11 depicts a network topology with multiple links of our consideration. In this scenario, we have three routers and two groups of flows, where group 1 ($S1-1$ to $S1-N$) traverses routers 1–3 and group 2 ($S2-1$ to $S2-N$) goes through routers 2 and 3. We use drop-tail for routers 1 and 3 with $O(N)$ buffer sizes, and for router 2, we use different AQM schemes as before with the scale of N^{α}. The capacities of routers 1 and 2 are $N \times 100$ and $2N \times 100$ kbps, respectively. The two-way propagation delays of group 1 flows are uniformly distributed over $[110, 190]$ ms with mean 150 ms, and over $[60, 140]$

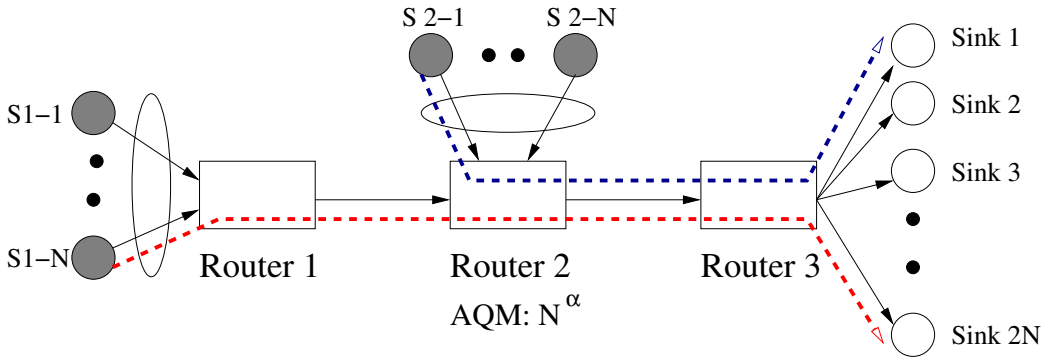


Figure 4.11: Simulation topology: multiple bottleneck links with heterogeneous RTTs.

with mean 100 ms for group 2.

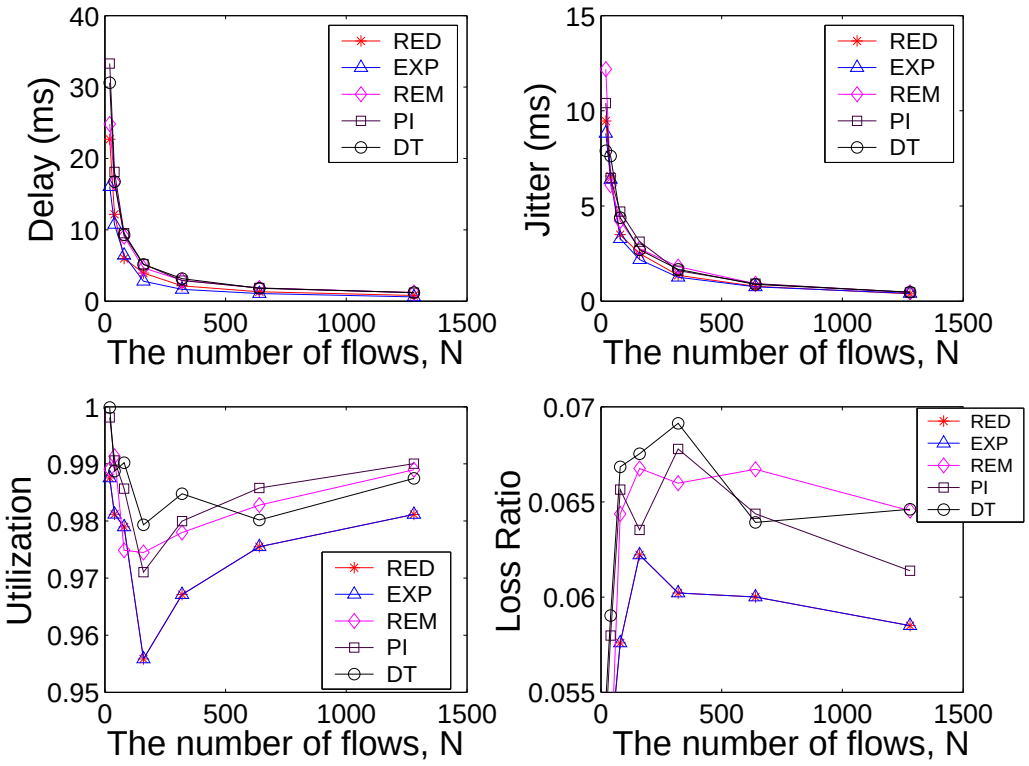


Figure 4.12: Multi-hop Topology: Performance metrics with increasing number of flows (N) for fixed $\alpha = 0.2$.

Figure 4.12 shows the similar trends for all the performance metrics measured at router 2, which demonstrate that our results can be applied to a more general network

environment. Regarding the link utilization and packet loss ratio, we see some zigzags when the number of flows is small (up to around 200). For such small values of N, it is still away from the domain of convergence, and the performance is mainly governed by other factors. For example, the buffer size in Table 4.2 is $12 \times N^{\alpha}$, and if N is not large enough, the resulting queue size is determined by the preceding factor, 12, rather than by N^{α}.

4.4 Conclusions

In this paper, we investigate the performance and the stability of TCP/AQM systems under aggressive packet marking. We demonstrate that traditional approaches in the literature fail to capture the system dynamics and may wrongfully predict the system performance if a more aggressive scale ($\alpha < 0.5$) for packet marking is used. By observing the two sources of randomness in packet arrivals to a TCP/AQM system, we have developed a doubly stochastic model to capture all the system dynamics under our scale. Based on our model, we prove that the system still yields good performance in terms of high utilization and low packet loss ratio, while making the queueing delay negligible and saving huge buffer space. We also provide an extensive set of numerical results using *ns*-2 under a variety of network configurations with different AQM schemes, and further validate our results in more realistic settings. We point out that the TCP/AQM system becomes unstable under our scale, but still preserves all the good performance as the number of flows and the size of the link capacity become large.

Chapter 5

Control TCP over Wireless Networks: A Resource Allocation Point of View

In this chapter, we will investigate TCP performance over wireless networks from the resource allocation point of view, specifically from call admission control (CAC) perspective. We first layout a two-level framework to allow us investigate CAC over the future packet-switched wireless networks. Based on the framework, we then propose a TCP-AIMD aware CAC scheme that takes the impact of AIMD algorithm into consideration. The idea is that the system will intentionally control the packet loss rate of TCP calls according the system status. In particular, if the system is under lightly loaded (i.e., the number of calls in the system is small), the system will allow smaller packet loss probability of TCP calls implying higher average TCP throughput (as we will show later on, the TCP throughput is reverse proportional to the packet loss probability). On the contrary, if the system is under the heavy load (i.e., the number of calls in the system is large and the system capacity is nearly full-utilized), the system will impose higher packet loss probability to reduce the throughput of the ongoing TCP calls and this allows the system to accommodate more call

requests to reduce the blocking probability. Our results show that by dynamically controlling the packet loss probability of the TCP calls, we can significantly improve the system performance metrics of both call level (e.g., smaller call blocking probability and higher call level throughput (call/min)), and packet level (e.g., higher link utilization).

This chapter is organized as follows. In Section 5.1, we describe TCP AIMD algorithm. We present system model in Section 5.2. TCP-AIMD aware CAC scheme is presented in Section 5.3. Packet level performance of our CAC scheme is analyzed in Section 5.4 and call level performance is investigated in Section 5.5. We provide extensive simulation results in Section 5.6 and this paper is concluded in Section 5.7.

5.1 Preliminary: TCP AIMD Algorithm

The basic idea of TCP congestion control is to explore the available network capacity from the TCP source point of view. If TCP source detects that there is no congestion (i.e., no packet loss) in the network, it keeps increasing the sending rate. If congestion happens (i.e, packet loss occurs), TCP source decreases the rate injecting to the network.

This feature of TCP congestion control algorithm is characterized by TCP AIMD [91]. Specifically, assume the window size of a TCP source in the current round trip time (RTT) is W. If all the W packets are transmitted successfully to the destination [1], the TCP source will inject $W + 1$ packets in the next RTT. If one of the W packets losses, the TCP source will reduce the window size to $W/2$. An example of TCP AIMD algorithm under a constant loss rate is plotted in Figure 5.1.

The impact of TCP AIMD algorithm on the packet-switched wireless access networks is that in addition to the packet loss caused by the unreliable wireless link [2], the packet loss due to a base station (BS) buffer overflow is inevitable, because packet loss is the congestion signal for TCP sources. Without congestion signal, the TCP sources will keep increasing the rate injecting to the network until packet loss occurs. In other words, the capacity of a BS will be fulfilled sooner or later and those packets beyond the capacity will be dropped by the BS and become the congestion signal for the TCP sources to reduce the sending rate.

[1] The TCP source detects the successful transmission by "ACK" packets sent back from the destination.

[2] In the rest of this paper, we assume the packet loss due to the wireless link is handled by MAC layer and transparent to the higher layer.

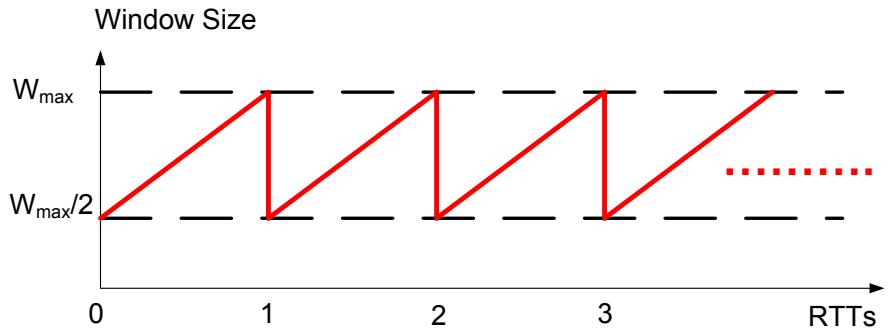


Figure 5.1: The TCP AIMD algorithm under a constant loss rate.

In this situation, the average throughput of a TCP call can be expressed by the well-known formula [154, 145] (cf. Equation (20) in [154])

$$\mathbb{E}\{TCP\} = \frac{1}{RTT}\sqrt{\frac{3}{2p(t)}}, \tag{5.1}$$

where $p(t)$ is the packet loss probability.

Moreover, for a TCP call, suppose the minimum window size is denoted by W. The path of window size evolution is $W, W+1, W+2, ..., W+W$. Then the window size drops by half and becomes W again. The average window size W^* can be obtained by $W^* = \frac{3W}{2}$ and we also know that $W^* = \sqrt{3/2p(t)}/RTT$ by (5.1). Therefore, the standard deviation of the window size for a single TCP call is

$$\delta_{TCP} = \sqrt{\frac{1}{W}\sum_{i=W}^{2W}(i - W^*)^2}. \tag{5.2}$$

In the rest of this paper, we will investigate how to control the TCP loss rate to reduce the impact to other types of services and improve the system performance such as throughput, link utilization, call blocking probability etc.

5.2 Framework: A Two-Level Model

We study a packet switch wireless networks supporting multi-class of services, which is the feature for the future wireless networks [6, 130, 75, 52, 89]. Unlike traditional

CAC scheme for wireless networks, in which only call level performance metrics are taken into account, we look at a system displayed in Figure 5.2 with two-level considerations: call level performance, packet level performance and their interactions.

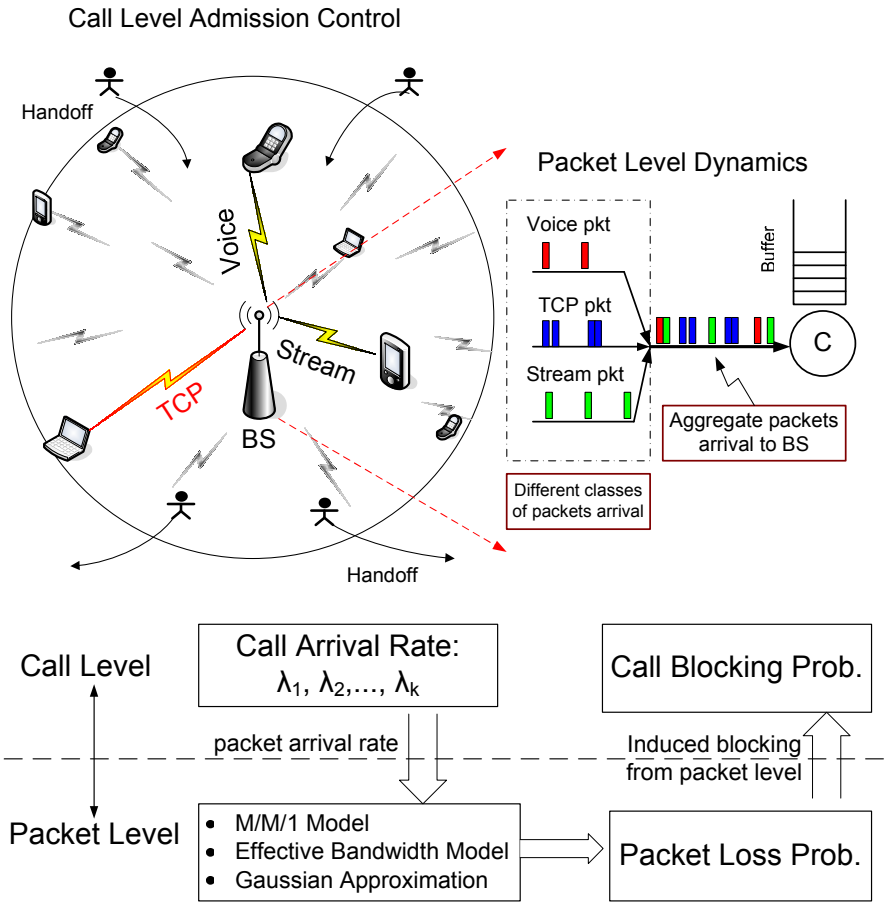


Figure 5.2: The system model and the interaction between call level and packet level.

At call level, the BS will allocate radio resource to the arrival call request: either new call or handoff call request [3]. Since the system provide service to multiple classes of services, each class of request is characterized by their own features: call arrival rate, and service holding time etc. At packet level, the feature is characterized by the QoS profile

[3]We note that for the packet-switch wireless network, even though a call is admitted to the system, the system does not reserve the resource for the specific call like circuit-switch network, rather the resource is shared by all the admitted calls, which is quite similar to GPRS wireless system. The general belief is that the packet-switch networks yields higher resource utilization than the circuit-switch networks.

that describes the packet arrival rate, packet service rate and packet loss requirement. As to different classes of service, the packet loss requirements may be different. For example, the voice traffic requires much smaller loss ratio than data traffic. In general, the real time traffic has more stringent loss requirement than the non-realtime traffic.

Intuitively, the interaction between call level and packet level is the following. if the BS accommodates more call requests into the system and each request has certain amount of data to be transmitted, the overall system load in terms of packets (average packets arrival rate) will be increased, which becomes the impact of call level on packet level. Hence, at the packet level, the system given fixed capacity (bits/sec) becomes more and more likely to be overwhelmed with the increase of the number of calls. In other words, the probability of packets dropped due to buffer overflow is increased. Since each type of service has certain QoS constraints (e.g. packet loss probability), in order not to violate the QoS agreement with the ongoing users, the system would control the admission of the arrival request, which is the reaction of packet level to call level. Such interaction of the two levels is illustrated in the lower graph of Figure 5.2.

In order to quantify the system behavior, we summary the parameters used for the later context in Table 5.1.

Table 5.1: System parameters: call level and packet level.

Call Level	
K	number of classes, $k = 1, 2, ..., K$
Λ_k	call arrival rate of class k (call/min)
Ψ_k	call service rate of class k (call/min)
P_b^k	call blocking probability of class k
Packet Level	
C	system capacity (bits/sec)
$\tilde{C}$	QoS guaranteed capacity
$\overline{p}$	QoS guaranteed packet loss prob.
p_T^u	upper bound of TCP packet loss prob.
p_T^l	lower bound of TCP packet loss prob.
$p(t)$	packet loss prob. of TCP call
B	buffer size (pkt)
λ_k	packet arrival rate of class k (pkt/sec)
μ_k	packet service rate of class k (pkt/sec)

We note that p_T^u is the upper bound of packet loss probability for TCP flows. In the later context, we assume the the packet loss only happens at the access link, i.e., at the

base station. In the middle of the link, there is no packet loss for the TCP flows. While, our scheme can be readying extended to incorporate the case, where the packet loss happens in the middle.

5.3 TCP-AIMD Aware CAC Scheme

In this section, we present the TCP-AIMD aware CAC scheme, describe how packet level dynamics affects the admission control at call level, and explain the impact of TCP application on the packet level behavior.

The idea is that in order to guarantee packet level QoS (say, packet loss probability), the system will control the total packet arrival rate (the summation of the arrival rate for each call) within certain threshold. In other words, if the total packet arrival rate is beyond the threshold, the packet loss probability will be greater than the predefined QoS requirement. Moreover, we know that in general, the QoS profile for a class of calls is fixed. For example, the packet arrival rate of voice traffic is fixed and the packet loss probability of voice traffic should be less than a certain fixed threshold. If the system serves only for those classes of calls with fixed QoS profile, the system does not have any flexibility to deal with (either accept it with guaranteed QoS, or reject it). However, for a TCP call, the QoS requirement is not fixed, but falls into a wide range, because TCP is a self-adaptation protocol. If there is more bandwidth, TCP will increase the transmission rate. If there is no bandwidth and congestion happens, TCP will reduce the transmission rate. Thus, TCP itself is capable to cope with a wide range of situations, which is a design feature of TCP protocol [91] and this feature allows the BS to have the flexibility to control the TCP calls. For example, if there is a new call request and the system has no enough resource (i.e., the total packet arrival rate is near the threshold) to admit this call directly, instead of rejecting the call immediately, the system will try to reduce the transmission rate of TCP call by controlling the $p(t)$ of TCP calls to make some room for the new call request. This thus increases the chance to accommodate the new call request and decreases the call blocking probability. Next, we will illustrate how our TCP-AIMD aware CAC works.

When there is a call request arriving at the system, if it is a handoff call, the BS will accept the call regardless of what class it belongs to. The system will not suffer the handoff dropping problem at call level at the expense of increasing likelihood of packet

Algorithm 1 TCP-AIMD Aware CAC: New Call Admission.

Require: Base Station (BS) status

A new call request 'm' arrives
if BS has enough resource **then**
    Accept 'm'
else
    "TCP Call Adjustment" /* adjust TCP calls */
    if BS has enough resource after TCP adjustment **then**
        Accept 'm'
    else
        Reject 'm'
    end if
end if

Algorithm 2 TCP-AIMD Aware CAC: TCP Call Adjustment.

Require: $p(t)$, p_T^U, $\tilde{C}$ and QoS profile of call 'm'

$n_1 \leftarrow$ the number of TCP call in the system

$\lambda \leftarrow$ the total packet arrival rate

for i = 1 to n_1 **do**

  /* reduce TCP call's transmission rate */

  $\Delta\lambda \leftarrow \frac{1}{RTT}\sqrt{\frac{3}{2p(t)}} - \frac{1}{RTT}\sqrt{\frac{3}{2p_T^U}}$

  $\lambda \leftarrow \lambda - \Delta\lambda$

  if $\lambda + E(m) \leq \tilde{C}$ **then**

    Accept 'm'

    Return "True"

  end if

end for

Reject 'm'

Return "False"

loss at packet level [4]. If the call request is a new call, the BS will examine whether there are enough resource or not. That is, the total packet arrival rate is smaller than the QoS guaranteed capacity $\tilde{C}$ or not ($\tilde{C}$ is obtained in Section 5.4). If true, the BS will accept the request; If not, the BS will proceed TCP call adjustment (cf. Algorithm 2). After TCP call adjustment, if the system has enough resource, the call request will be accepted, otherwise, the call will be rejected. This is the algorithm for new call admission displayed in Algorithm 1. We note that Algorithm 1 mainly deals with call level decision, and the packet level consideration is reflected in Algorithm 2: TCP call adjustment. The BS will enumerate all the ongoing TCP calls and reduce the packet level transmission rate of TCP calls. Hence, the total transmission will be decreased until there is enough room to accommodate the new request.

5.3.1 Discussion on the Packet Loss in the Middle Links of Networks

We note that p_T^u is the upper bound of packet loss probability for TCP flows. We first assume the the packet loss only happens at the access link, i.e., at the base station. In the middle of the link, there is no packet loss for the TCP flows. Second, our scheme can be readily extended to incorporate the case, where the packet losses occur in the middle links as well. Assume that the packet loss probabilities at the base station and in the middle link are p_T^U and p_m, respectively. Then, the overall packet loss probability is $1-(1-p_T^U)(1-p_m) \approx p_T^U + p_m$, and such approximation holds when p_T^u and p_m are small. The the average throughput of TCP flow is given by $E\{TCP\} = \frac{1}{RTT}\sqrt{\frac{3}{2(p_T^U+p_m)}} < \frac{1}{RTT}\sqrt{\frac{3}{2p_T^U}}$. This shows that the average end-to-end TCP throughput will be even smaller than throughput that the base station is guaranteed to transmit. In other words, the base station is willing to deliver more packets for the TCP flow, but the bottleneck along the path limits the throughput to a lower level. From the base station side, it is able to fulfill the agreement with the mobile users and it is the mobile user that does not fully utilize the base station's capacity. As a consequence, the packet losses due to the middle link further reduce the transmission rate of the TCP flows, and hence the base station get more room at the packet level to accommodate the incoming new call requests.

In order to fully utilize the information of packet losses in the middle of the links,

[4] An alternative approach is not to admit the handoff call unconditionally. Rather, the base station will accept the handoff request based on the packet level constraint. There is a tradeoff between handoff dropping probability and packet loss probability.

we define the notion of the effective TCP throughput seen by base station, $T_{TCP}(p(t))$ as the effective average TCP throughput when the base station imposes the packet loss probability $p(t)$ and the flow is subject to some other losses in middle of the networks, which is unknown to the base station. Still, the base station can 'observe' the amount of packets passing through from the particular flow, i.e., $T_{TCP}(p(t))$. Due to the visibility of $T_{TCP}(p(t))$ to the base station in practice, the base station can proceed the TCP call adjustment as well. We therefore revise our TCP adjustment algorithm as displayed in Algorithm 3.

Algorithm 3 TCP Call Adjustment with Packet Loss in the Middle Links.

Require: $p(t)$, p_T^U, $\tilde{C}$ and QoS profile of call 'm'

$n_1 \leftarrow$ the number of TCP call in the system

$\lambda \leftarrow$ the total packet arrival rate

for i = 1 to n_1 **do**

  /* reduce TCP call's transmission rate */

  $\Delta\lambda \leftarrow T_{TCP}(p(t)) - T_{TCP}(p_T^U)$

  $\lambda \leftarrow \lambda - \Delta\lambda$

  if $\lambda + E(m) \leq \tilde{C}$ **then**

    Accept 'm'

    Return "True"

  end if

end for

Reject 'm'

Return "False"

5.3.2 Further Discussions on Implementing Rate Control in Reality

In the our admission and rate control scheme, the TCP rate is adjusted by controlling the packet loss probability $p(t)$, and different TCP flows may have different $p(t)$. A natural question is how the base station can accurately know this $p(t)$ for each TCP flow?

In reality, we know that each TCP flow maintains its own timer and counter to keep track of RTT, the number packet sent out and number packets acknowledged. The base station could obtain this $p(t)$ information from the TCP sender, and verify the TCP sending rate to the base station as well. If the error within the tolerable range (say 1%), then the base station knows the packet loss probability experience by TCP sender is indeed what it observes, i.e., the number of dropped packets divided by the number of packets pass through the base station. If the error is larger than the tolerable range, then the base station will simply choose the largest value of packet loss probability to ensure that the aggregate packets will not overflow the packet level system capacity. We display the such algorithm in Figure 5.3.

Algorithm: Obtain $p_i(t)$
Input: The inquiry period T; the tolerable error δ;
The base station has the initial $p_i(0)$.
Output: The base station obtain the accurate $p_i(t)$.
Method:
01. The base station keeps track of the incoming packet rate for each TCP flow i.
02. Every T seconds, the base station send $p_i(t)$ request to each TCP flow i.
03. Each TCP flow returns the average $p_i'(t)$ to the base station.
04. **IF** $|p_i(t) - p_i'(t)| \leq \delta$
05. The error within the tolerable range.
06. $p(t+1) \leftarrow$ updated packet loss probability desired for the BS.
07. **ELSE**
08. The base station updates its own $p_i(t)$ according to the TCP sender's report.
09. $p(t+1) \leftarrow \max\{p_i'(t), p_i(t)\}$
10. **END**

Figure 5.3: The algorithm for the base station to obtain $p(t)$.

We should note that our rate control algorithm heavily depends on the formula (5.1), which is accurate only when $p(t)$ is small, say 1% or 2%. Generally, it is believed based on the simulation test that when the packet loss probability is less than 6%, the formula

could yield satisfactory accuracy to be used to estimate the average TCP throughput. In the heavy packet loss situations, e.g., multi-hop ad-hoc fashion (the packet loss ratio could be higher than 20%), the formula may not produce the accurate the TCP throughput, because the higher order term of $\frac{1}{\sqrt{p(t)}}$ cannot be ignored. For those situations, it is beyond the scope of our topic. Meanwhile, it is believed that the future wire line and wireless network will maintain much higher reliability and throughput, so it is kind of safe to use the formula (5.1) in the future practice. Once the base station has this $p(t)$ in hand, it can drop the packet based on a stochastic fashion displayed in Figure 5.4.

Moreover, in reality, we know different TCP flows may experience different RTTs, which result in various transmission rates even under the same $p(t)$. Thus, in reality, only with $p(t)$ in hand is not enough for the base station to effectively control TCP's transmission rate, and the base station should obtain the RTT information for each flow as well. Since each TCP flow has an internal timer to measure its own RTT, it could update the RTT to the base station periodically, say every 10 or 20 RTTs. In addition, the base station needs to be aware of TCP fairness issue.

Remark: We admit that by introducing this $p(t)$ into the base station for rate control, we incur some overhead of the computation, design and implementation. Intuitively, this $p(t)$ is unlike power control, error control or signaling protocols, which are essential features embedded in the base station. In order to implement $p(t)$ in a real wireless system, we have to consider several possible issues associated. First, the base station need to keep track of the transmission rate for each TCP flow. Second, it would communicate with each TCP flow to obtain the RTT for each flow periodically. However, close look at both overhead induce reveals that the actual overhead to the system is tangible and within the range of future wireless system capacity. Since the formula (5.1) is very simple, the computation overhead for each flow is not large, and we know that the future wireless cellular system tends to be smaller and smaller in order to increase the wireless bandwidth utilization, e.g., micro cell, pico cell etc. The number of mobile users within one cell is on the level of tens or hundreds compared to the fast increasing computation capacity of DSP. We can expect that the computational overhead will not become an obstacle for the implement this such rate control scheme. On the other hand, since the mobile terminals will keep update this status information to the base station frequently. The $p(t)$ and RTT information could be implemented as piggypack along with the existing exchange information between the base station and the TCP flows.

Algorithm: Packet Dropping Decision based on $p_i(t)$
Input: The average packet loss probability $p_i(t)$;
There is a incoming packet from TCP flow i.
Output: The dropping decision for the incoming packet of TCP flow T_i.
Method:
1. The base station receives a packet from TCP flow i, i.e.,
2. The base station decide the desirable p_i for T_i
3. The base station generates a r.v. X uniformly distributed in [0,1]
4. **IF** $X \leq p_i(t)$
5. Transmit the packet for T_i
6. **ELSE**
7. Dropped the packet for T_i
8. **END**
9. goto 1.

Figure 5.4: The rate control scheme based on $p_i(t)$: stochastic packet dropping.

As to Fast TCP or TCP vegas, which are delay-based TCP variants, the base station cannot control the transmission rate through intensionally dropping the packets. Still, the idea of control the TCP injection rate to the local network can be applied to improve the efficiency and performance of admission control. There are two ways to implement this idea. One is that the base station will not drop the packets from that particular TCP flow, rather the base station only notifies the TCP end to reduce the sending rate. The TCP end correspondingly reduces its transmission rate. The pros for this approach is that the overhead for the base station is small. The cons are that this approach is not transparent to the TCP end, and the information between the base station and the TCP end should be communicated through another mechanics. In order to make is transparent to the TCP end, the base station need to intentionally delay the packets from the TCP sender. That is, the base station puts the incoming packets from that TCP sender, wait for some time and then transmit them out. Those packets experiencing a longer delay will make the TCP sender 'believe' that the congestion happens somewhere in the network and hence reduce the transmission rate. Finally, we should note that for the wireless access networks, the transmission rate is not as high as assumed to be applied by Fast TCP, which is designed for high bandwidth and delay product network, say 10 Gbps or 40 Gbps. Here we just try to justify that our rate control idea is quite general and can be tailored to the

delay based TCPs. Even though our TCP rate control algorithm is designed for TCP with AIMD algorithm, it is ready to be extended to AIAD [196], MIMD [14], BIC [197] or Cubic TCP [167].

Algorithm: Admission Control For a Base Station with a Queue
Input: Acall arrival '**M**'; remaining number of channels '**C**';
the remaining buffer capacity '**B**'.
Output: An admission decision of a AP.
Method:

```
01. A call M arrives at the AP
02. IF C > 0
03.     Accept the call
04.     C ← C − 1
05. ELSE
06.     Notify M that there is no available channel and ask
        M whether it wants to join the queue to wait or not
07.     M will make a decision based on Algorithm 2
08.     IF M wants to join the queue
09.         IF B > 0
10.             put the call into the queue
11.             B ← B − 1
12.         ELSE
13.             Reject the call
14.         END
15.     ELSE
16.     M withdraws the call request and leaves the system
17.     END
18. END
```

Figure 5.5: Stochastic admission control for a system with a queue.

Finally, we point out in reality, the base station may have a queue for the handoff call or incoming call. We then describe how our CAC scheme admit a call request stochastically. When there is a new call request arriving at the BS, if the BS has the available channels, it will allocate one of the channels to the call. If all the channels are occupied by the ongoing calls, the BS will notify the mobile users that currently there is no resource available and the new call could be put into the queue (if the queue is not full yet) to wait until the channel is available. Up to here, the our CAC scheme is indifferent from the existing ones, but the difference is the following. After receive such notification, the mobile user

will decide whether to join the queue or not. If the mobile user decide to join the queue, then the BS will put it into the queue; if not, the mobile user simple leaves the system and the BS discards the request. The pseudocode of the admission control procedures is displayed in Figure 5.5.

The decision of whether a mobile user joins the queue or not is determined by its utility function. The objective of each mobile user is to maximize its utility function. In other words, if join the queue tends to increase the utility, the mobile user will join. If not, the mobile user will avoid joining. Since the system is random in that the arrival is random and service is random as well, the utility function could be defined through the average of certain random variables. Thus, the decision of whether to join the queue or not is not deterministic as well, rather it is a stochastic decision. We therefore denote p as the probability to join the queue. For example, $p = 0.8$ means that on average, the mobile user will join the queue 8 times out of 10 times (with 2 times not join the queue). If $p = 0.2$, the mobile user is more reluctant to join the queue compared with $p = 0.8$. As a special case, for $p = 1$ ($p = 0$), the mobile user will always join (not join) the queue and hence this degenerates to the deterministic case.

Further discussion on this issue becomes the future work of this dissertation.

5.4 Performance Analysis: Packet Level

Since the TCP-AIMD aware CAC scheme is based on a two-level framework, the performance of our scheme consists of two folds: call level and packet level. Moreover, since the call level performance is based on the packet level dynamics, we first present packet-level performance analysis. Define $\omega = (n_1, ..., n_K)$, where n_k is the number of calls of Class k. Assume TCP call is the Class 1 and the average throughput of TCP call $\mathbb{E}\{TCP\}$ can be obtained by (5.1). Denote $\mathbb{E}\{X_k\}$ be the average packet level throughput of Class $k = 2, ..., K$, given by the predefined QoS profile. Let Ω be the state space collecting all the admissible/possible states in the system and $\Omega_k, k = 1, ..., K$ be the sub-space consisting of all the boundary states. We formulate Ω and Ω_k as follows.

$$\Omega = \left\{ \omega \in \mathbb{Z}^{+K} \middle| n_1 \mathbb{E}\{TCP\} + \sum_{k=2}^{K} n_k \mathbb{E}\{X_k\} \leq \tilde{C} \right\}, \tag{5.3}$$

$$\Omega_k = \left\{ \omega \in \Omega \left| n_1 \mathbb{E}\{TCP\} + \sum_{k=2}^{K} (n_k + 1)\mathbb{E}\{X_k\} > \tilde{C} \right. \right\}, \tag{5.4}$$

where $k = 2, ..., K$ and for $k = 1$,

$$\Omega_1 = \left\{ \omega \in \Omega \left| (n_1 + 1)\mathbb{E}\{TCP\} + \sum_{k=2}^{K} n_k \mathbb{E}\{X_k\} > \tilde{C} \right. \right\}, \tag{5.5}$$

Here $\tilde{C}$ is the QoS guaranteed capacity in Table 5.1 and the interpretation of $\tilde{C}$ is the maximum total packet arrival rate to the system with QoS guarantee. In other words, if the total packet arrival rate is greater than $\tilde{C}$, then the system cannot guarantee the predefined QoS requirement such as packet loss probability. $\tilde{C}$ plays the key role to joint call level and packet level. On one hand, the larger $\tilde{C}$ allows the system to accommodate more states implying higher *call level* capacity and lower call blocking probability; On the other hand, $\tilde{C}$ is determined through the packet loss probability at *packet level*.

Next, we investigate how to obtain the packet loss probability and the induced $\tilde{C}$ based on four models, each of which are widely applied to study the stochastic loss issues [112, 171, 114]: (i) $M/M/1$ model; (ii) a priority queue model; (iii) effective bandwidth model; (iv) Gaussian approximation model.

5.4.1 $M/M/1$ Model

In order to obtain the packet loss probability, we start with a $M/M/1$ model that is simplest and widely used queueing model displayed in Figure 5.6. A $M/M/1$ model assumes the total arrival rate of all the classes follows Poisson process with mean λ and the service holding time is exponentially distributed with mean$1/\mu$. In this model, the service rates of different class is assumed to be the same and the arrival rate the is the summation of the arrival rate of each class.

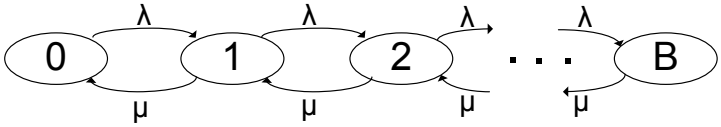


Figure 5.6: $M/M/1$ model of packet arrival/serivce process.

Let $\rho = \frac{\lambda}{\mu}$ be the traffic intensity, and the packet loss probability can be stated as

$$P\{loss\} = \frac{1-\rho}{1-\rho^{B+1}}\rho^B,$$

where B is the base station's buffer size in Table 5.1.

In order to make the packet loss probability satisfy the predefined QoS requirement $\overline{p}$, i.e., $P\{loss\} \leq \overline{p}$, we define Q_λ is the collection of all the arrival rate such that

$$Q_\lambda = \left\{ \lambda \left| \frac{1-\rho}{1-\rho^{B+1}}\rho^B \leq \overline{p} \right. \right\}.$$

For any given $\overline{p}$, we can obtain the corresponding the QoS guaranteed capacity given by

$$\tilde{C} = \sup\left\{Q_\lambda\right\}. \tag{5.6}$$

5.4.2 A Priority Queue Model

In practice, different classes have different priorities in sharing a queue. For example, voice traffic has a higher priority than data traffic. In this section, we present the packet loss rate for a priority queue. To avoid cumbersome presentation, we analyze two classes of packet arrival: voice and TCP, but the idea of our approach is quite general and can be extended to many classes for Diffserv [77] and Traffic Engineering [21].

Consider two classes of traffic with Poisson arrive rates λ_1 and λ_2, exponential service rates μ_1 and μ_2. The priorities of two classes are characterized by two monotone nondecreasing weight function $g_1(x)$ and $g_2(x)$ such that $g_1(x_0) > g_2(x_0), \quad \forall x_0$. This represents that for the same backlog of two classes, class 1 has higher priority than class 2. Such two classes model with priority queue is displayed in Figure 5.7.

Define a triplet $\zeta = (0/1, m_1, m_2)$, where 0/1 is the flag variable with 0 (resp. 1) indicating a packet of class 1 (resp. 2) being transmitted by the BS; m_1 and m_2 are the number of packets of class 1 and 2 in the buffer, respectively. Let the projection mapping functions be $\pi_0(\zeta) = 0/1$, $\pi_1(\zeta) = m_1$, $\pi_2(\zeta) = m_2$. Note that the value of $\pi_0(\zeta)$ is either 0 or 1. Moreover, define if $\pi_0(\zeta) = 0$, then $\pi_0(\zeta) + 0/1 = 0/1$; if $\pi_0(\zeta) = 1$, then $\pi_0(\zeta)+0/1 = 1/0$. Further, define $e_{11} = (0,1,0), e_{12} = (1,1,0), e_{21} = (0,0,1), e_{22} = (1,0,1)$. The steady state balance equations of the priority queue are shown below.

1. If $\pi_1(\zeta) > B$, then it is an impossible state, thus, we have

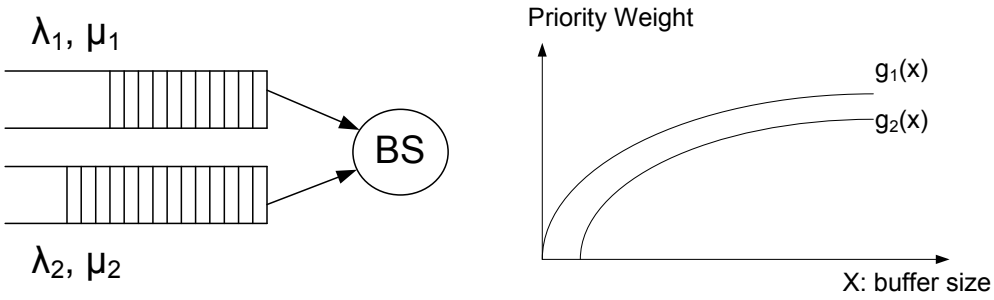


Figure 5.7: The system model with high priority queue and the priority weight function with queue length: $g_1(x)$ and $g_2(x)$ are monotone nondecreasing functions with $g_1(x_0) > g_2(x_0), \ \forall x_0$.

$$P(\zeta) = 0 \ .$$

2. If $\pi_1(\zeta) = B, \pi_2(\zeta) = 0$, then all the arrival will be blocked and there are no incoming class 2 packets from backward states [5], thus, we have

$$P(\zeta)[1_{\{\pi_0(\zeta)=1\}}\mu_1 + 1_{\{\pi_0(\zeta)=2\}}\mu_2] = \sum_{i=1}^{2} P(\zeta - e_{1i})\lambda_1.$$

3. If $\pi_1(\zeta) + \pi_2(\zeta) = B, \pi_1(\zeta) > 0, \pi_2(\zeta) > 0$, then all the arrival will be blocked, thus, we have

$$\begin{aligned} & P(\zeta)[1_{\{\pi_0(\zeta)=1\}}\mu_1 + 1_{\{\pi_0(\zeta)=2\}}\mu_2] \\ &= \sum_{i=1}^{2} P(\zeta - e_{1i})\lambda_1 1_{\{g_1(\pi_1(\zeta)) \geq g_2(\pi_2(\zeta))\}} \\ &\quad + \ \textstyle\sum_{i=1}^{2} P(\zeta - e_{2i})\lambda_2 1_{\{g_1(\pi_1(\zeta)) < g_2(\pi_2(\zeta))\}}. \end{aligned}$$

4. If $\pi_1(\zeta) + \pi_2(\zeta) < B, \pi_1(\zeta) > 0, \pi_2(\zeta) > 0$, then the any arrival will be accepted, thus, we have

$$\begin{aligned} & P(\zeta)[\lambda_1 + \lambda_2 + 1_{\{\pi_0(\zeta)=1\}}\mu_1 + 1_{\{\pi_0(\zeta)=2\}}\mu_2] \\ &= \sum_{i=1}^{2} P(\zeta - e_{1i})\lambda_1 1_{\{g_1(\pi_1(\zeta)) \geq g_2(\pi_2(\zeta))\}} \\ &\quad + \ \textstyle\sum_{i=1}^{2} P(\zeta - e_{2i})\lambda_2 1_{\{g_1(\pi_1(\zeta)) < g_2(\pi_2(\zeta))\}}. \end{aligned}$$

[5] We denote $1_{\{A\}}$ as the indicator function of event A.

5. If $\pi_2(\zeta) = B, \pi_1(\zeta) = 0$, then there are no incoming packets of Class 1 from backward states, thus, we have

$$P(\zeta)[1_{\{\pi_0(\zeta)=1\}}\mu_1 + 1_{\{\pi_0(\zeta)=2\}}\mu_2] = \sum_{i=1}^{2} P(\zeta - e_{2i})\lambda_2.$$

We can obtain the packet loss probability by collecting all the boundary states

$$\begin{aligned} P\{loss\} \quad = \quad & \sum_{\left\{\zeta:\ \substack{\pi_1(\zeta)=B \\ \pi_2(\zeta)=0}\right\}} P(\zeta) + \sum_{\left\{\zeta:\ \substack{\pi_2(\zeta)=B \\ \pi_1(\zeta)=0}\right\}} P(\zeta) \\ & + \sum_{\left\{\zeta:\ \substack{\pi_1(\zeta)+\pi_2(\zeta)=B \\ \pi_1(\zeta)>0,\pi_2(\zeta)>0}\right\}} P(\zeta). \end{aligned}$$

Thus, the QoS guaranteed capacity is given by

$$\tilde{C} = \sup_{\lambda_1,\lambda_2} \Big\{ \lambda_1 + \lambda_2 : P\{loss\} \leq \overline{p} \Big\}. \tag{5.7}$$

The balance equations for the priority queue model can be solved by the numerical approach proposed in [82]. This method is a recursive technique based on the state probability of a subsets of Chapman-Kolmogroff equations. It first identifies some states called "boundary states" within the state space and remaining states are the linear combinations of the boundary states. The idea is to solve the balance equations for those boundary states and then all the states in the space can be derived based on the boundary states. Unlike the traditional approach solving a huge transition matrix inversion, the proposed technique significantly reduces the computational complexity and overhead (cf. more details in [82]).

5.4.3 Effective Bandwidth Model

"Effective bandwidth" is another important approach to obtain the required capacity for a given loss rate under the situation with multiplexing gain [103, 107, 114, 113]. It determines a bandwidth between the maximum rate and the average rate, and this is possible due to the existence of multiplexing gain among the ongoing flows. The idea is that the packet loss rate can be approximated by $e^{-BI(C)}$, where B is the buffer size, C is the packet level capacity. $I(C)$ is some monotone nondecreasing function of the capacity C and the traffic statistics, which is used as a measure of the upper bound of the packet loss

rate. Then, we can get

$$\lim_{B\to\infty} \frac{1}{B} \log P\{Y > B\} \leq I(C),$$

where Y is the random variable of stationary buffer size.

We can obtain $I(C)$ for a ON-OFF type of traffic by [15]

$$I(C) = \beta \exp\left\{ -\frac{B(c_i - \alpha r)}{t(1-\alpha)(r-c_i)c_i} \right\}, \tag{5.8}$$

where r is the peak rate, t is the mean duration time, c_i is the effective bandwidth to guarantee the packet loss probability with upper bound $I(C)$, α is the activity ratio, and β is a intermediate parameter given by

$$\beta = \frac{(c_i - c_i r) + \epsilon\alpha(r - c_i)}{(1-\alpha)c_i}. \tag{5.9}$$

However, to solve c_i directly from (5.8) and (5.9) is time consuming. In order to reduce the computational overhead, an approximation solution is given by

$$c_i = \frac{r\left[\theta - B + \sqrt{(\gamma - B)^2 + 4\theta\alpha B}\right]}{2\theta},$$

where $\theta = \ln(\frac{1}{I(C)})t(1-\alpha)r$.

Then, the QoS guaranteed capacity $\tilde{C}$ can be stated as

$$\tilde{C} = \sum_{k=1}^{K} c_k. \tag{5.10}$$

5.4.4 Gaussian Approximation Model

The Gaussian process is widely applied to the situation, where many independent sources are multiplexed, and validated by Cental Limit Theorem [172, 171]. We therefore have a much simpler way to calculate the packet loss probability.

Let X be the packet arrival process and assume X is Gaussian process with mean λ and variance δ^2. Then, the packet loss probability for a system with capacity C is given by

$$\begin{aligned} P\{loss\} &= P\{X \geq C\} \\ &= \frac{1}{\delta\sqrt{2\pi}} \int_C^\infty \exp\left\{ -\frac{(x-\lambda)^2}{2\delta^2} \right\} dx \\ &= 1 - \phi\Big(\frac{X - C}{\delta}\Big) \\ &= \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{C-\lambda}{\sqrt{2}\delta}\right)\right]. \end{aligned}$$

In Gaussian approximation model, all the flows are assumed to be independent and thus we have

$$\begin{aligned} \lambda &= n_1\mathbb{E}\{TCP\} + \sum_{k=2}^{K} n_k\mathbb{E}\{X_k\}, \\ \delta &= \sqrt{n_1^2\delta_{TCP}^2 + \sum_{k=2}^{K} n_k^2\delta_i^2}, \end{aligned}$$

where $(n_1, ..., n_K) = \omega$ is the state vector, δ_{TCP} is given by (5.2), and the standard deviation of other class of calls, δ_k, $k = 2, ..., K$ can be obtained by predefined QoS profile like $\mathbb{E}\{X_k\}$.

Then, for a given C and δ, we can obtain

$$\tilde{C} = \sup_{\lambda} \left\{ \lambda : P\{loss\} \leq \overline{p} \right\}. \tag{5.11}$$

5.4.5 Discussion

The QoS guaranteed capacity $\tilde{C}$ is obtained in (5.6), (5.7), (5.10), and (5.11) by four models, respectively. The rationales behind these four models are that $M/M/1$ model is the simplest model for a queueing approximation. The priority queue model improves $M/M/1$ by taking different priority queues into account at the expense of more computational overhead. However, both $M/M/1$ and priority queue models require Poisson arrival process, which cannot be applied to the burst traffic. Effective bandwidth model is widely recognized in capturing the effect of burst arrival process, while the shortcoming of effective bandwidth is its inaccuracy when the buffers are small [79]. The Gaussian approximation model based on Central Limit Theorem (CLT) is appealing due to the simplicity and fastness in determining the required capacity. The limitation of CLT, which becomes the limitation of Gaussian model in return, is that the approximation is accurate only for a large number of sources.

Since each model has its own pros and cons, our objective is not to advocate some of them and deemphasize others. Rather, they provide us plenty of tools to calculate the packet level performance under different circumstances and facilitate us to analyze the call level performance.

5.5 Performance Analysis: Call Level

With the packet level results in the previous section, we are ready to analyze the call performance under different scenarios. We consider two cases: (i) Case I: the transmission time is independent of the transmission rate, which is the analogy that the channel holding time is independent of the number of channels assumed by most of the existing work [52, 89, 9, 129]; (ii) Case II: the transmission time is dependent on the transmission rate. This can be widely applied to the data transmission scenarios, where the more transmission rate the call possesses, the shorter transmission period the call experiences. To the best of our knowledged, there is no CAC literature analyzing the performance under Case II.

We start with a general form for call blocking probability, and then we present the analysis for Cases I and II as a special case of the general form, respectively.

Moreover, we point out that the assumption of service distribution in both Case I and Case II, which is widely assumed to be exponential distribution, can be relaxed to the any arbitrary distributions. It is know that the Poisson process for the arrival is a good approximation in any situations, while the exponential service distribution is not realistic and has to be enforced only for the analysis purpose. However, with the recent queueing theory, we show that in our framework, the performance measures of both Case I and Case II are insensitive to the service distribution and is determined only through its mean. In other words, the performance measures obtained under exponential service distribution can be applied to any other service distributions, which greatly expand the applicable situations of our results and are in accordance with the fact that the Erlang types of formula work well in practice.

5.5.1 Call Blocking Probability: A General Form

Assume the call arrival process for class k is non-homogeneous Poisson process with mean $\Lambda_k(\omega)$ and the service holding time is exponential with mean $\Psi_k(\omega)$, which means both the arrival process and the service process are state dependent processes.

Define the $K-$dimensional base vectors as

$$\begin{aligned} e_0 &= \underbrace{(0,0,...,0)}_{K} \\ e_1 &= (1,0,...,0) \\ e_2 &= (0,1,...,0) \\ ... &\quad ... \\ e_K &= (0,0,...,1). \end{aligned}$$

and the function $T_i^j(\omega)$ as $T_i^j(\omega) = \omega - e_i + e_j$.

Consider a given state $\omega \in \Omega$ with a measure $P(\omega)$, the outgoing transition rate of ω, denoted as $\mathcal{A}(\omega, \rightarrow)$, and the incoming transition rate of ω, denoted as $\mathcal{B}(\rightarrow, \omega)$, can be stated as

$$\begin{aligned} \mathcal{A}(\omega, \rightarrow) &= \begin{cases} \Lambda_k(\omega) & \text{if } T_0^k(\omega) \in \Omega \\ n_k \Psi_k(\omega) & \text{if } T_k^0(\omega) \in \Omega \end{cases} \\ \mathcal{B}(\rightarrow, \omega) &= \begin{cases} \Lambda_k(T_k^0(\omega)) & \text{if } T_k^0(\omega) \in \Omega \\ (n_k+1)\Psi_k(T_0^k(\omega)) & \text{if } T_0^k(\omega) \in \Omega \end{cases} \end{aligned}$$

where $k = 1, ..., K$.

Therefore, combining $\mathcal{A}(\omega, \rightarrow)$ and $\mathcal{B}(\rightarrow, \omega)$ with the stationary probability distribution $P(\omega)$, we can obtain the general form of steady-state balance equation expressed by

$$\begin{aligned} &\left[\sum_{k=1}^{K} \Lambda_k(\omega) 1_{\{T_0^k(\omega) \in \Omega\}} + \sum_{k=1}^{K} n_k \Psi_k(\omega) 1_{\{T_k^0(\omega) \in \Omega\}} \right] P(\omega) \\ &= \sum_{k=1}^{K} \Lambda_k(T_k^0(\omega)) P(T_k^0(\omega)) 1_{\{T_k^0(\omega) \in \Omega\}} \\ &\quad + \textstyle\sum_{k=1}^{K} (n_k+1) \Psi_k(T_0^k(\omega)) P(T_0^k(\omega)) 1_{\{T_0^k(\omega) \in \Omega\}}, \end{aligned}$$

where $\Lambda_k(\omega)$ and $\Psi_k(\omega)$ are state dependent variable.

$$\sum_{\omega \in \Omega} P(\omega) = 1, \tag{5.12}$$

where $P(\omega)$ is stationary probability measure of state ω.

Then, the blocking probability of class k is equal to the sum of all the probability of the states, $P(\omega)$, to be blocked, i.e., all the states in the subset, Ω_k, given in (5.4). Hence, the blocking probability of class k given satisfying QoS constraint, P_b^k, can be written as

$$P_b^k = \sum_{\omega \in \Omega_k} P(\omega), \tag{5.13}$$

where Ω and Ω_k are given in (5.3) and (5.4), respectively.

Remark: *The impact of packet level dynamics on the call level performance is reflected by the definition of Ω in (5.3) and Ω_k in (5.4). Different packet level requirements result in a different call level state space Ω and subspace Ω_k. Here, we only layout a framework and further discussion is beyond this work.*

Moreover, the close form solution under such a general condition is not available [6], because $\Lambda_k(\omega)$ and $\Psi_k(\omega)$ are state dependent variables. We can obtain the close form solution of $P(\omega)$ under some specific conditions: Case I and Case II.

5.5.2 Call Blocking Probability for Case I

As the assumption used in[52, 89], we assume the arrival rate is Λ_k and service rate is Ψ_k for class k, both of which are constants and not depends on states. Therefore, for Case I, the steady-state balance equation can be expressed by

$$\begin{aligned}
&\left[\sum_{k=1}^{K} \Lambda_k 1_{\{T_0^k(\omega)\in\Omega\}} + \sum_{k=1}^{K} n_k \Psi_k 1_{\{T_k^0(\omega)\in\Omega\}}\right] P(\omega) \\
&= \sum_{k=1}^{K} \Lambda_k P(T_k^0(\omega)) 1_{\{T_k^0(\omega)\in\Omega\}} \\
&\quad + \textstyle\sum_{k=1}^{K} (n_k+1)\Psi_k P(T_0^k(\omega)) 1_{\{T_0^k(\omega)\in\Omega\}},
\end{aligned}$$

In this simplified situation, the close form of $P(\omega)$ is achievable given by

$$P(\omega) = \frac{1}{\Phi(\Omega)} \prod_{k=1}^{K} \frac{\Lambda_k^{n_k}}{\Psi_k^{n_k} n_k!}, \quad \omega \in \Omega, \tag{5.14}$$

and then the normalized factor $\Phi(\Omega)$ can be solved by appealing (5.12),

$$\Phi(\Omega) = \sum_{\omega \in \Omega} \prod_{k=1}^{K} \frac{\Lambda_k^{n_k}}{\Psi_k^{n_k} n_k!}.$$

[6] We can obtain $P(\omega)$ by numerical methods as in [98, 130].

Then, the blocking probability P_b^k of Case I can be obtained by (5.13), which is verified by our simulations (cf. Figures 5.12 to 5.15).

We note that the exponential service time with mean Ψ_k can be extend to any arbitrary distribution with mean Ψ_k as long as the arrival process is Poisson. This is so called "insensitivity" property [174, 173, 33] and we verify this in our simulations (cf. Figure 5.17).

5.5.3 Call Blocking Probability for Case II

In Case II, we consider the transmission time of TCP calls depends on transmission rate (i.e., Ψ_1 is state dependent) and the transmission time of other classes of calls do not depend on their transmission rate, which are the same as in Case I (i.e., $\Psi_k, k = 2, ..., K$ are state independent) [7]. Moreover, assume the amount of data to be transmitted by TCP call is arbitrarily distributed with mean ξ.

In Case II, since the service time is not exponential distributed, the system is not a Markov process anymore, but a Generalized Semi-Markov Process (GSMP) (cf. [174, 173]). However, if we look at the process at the moment at each arrival, then system is still a Markov process embedded in the original process, which is the "embedded Markov process". This embedded Markov process is irrelevant to the service holding time distribution by "insensitivity" property [174, 173, 33]. In other words, this embedded Markov process is the same for any arbitrary service holding time with the same mean, which is validated in our simulation section (cf. Figure 5.23). The following theorem makes the above observation rigorous.

Theorem 5 *In Case II, the measure for each state $\omega \in \Omega$ is insensitive to the workload distribution and is determined only through its mean ξ.*

Proof: First show that it is indeed partial balanced, and then invoking [177], we have the insensitivity property.

[7] The same technique can be applied to the situation that the service time of all the classes depends on the transmission rate with more cumbersome formulation and presentation.

Therefore, for Case II, the embedded balance equation is given by

$$\begin{aligned}
&\left[\sum_{k=1}^{K}\Lambda_k 1_{\{T_0^k(\omega)\in\Omega\}}+\sum_{k=1}^{K}n_k\Psi_k(\omega)1_{\{T_k^0(\omega)\in\Omega\}}\right]P(\omega)\\
&=\sum_{k=1}^{K}\Lambda_k P(T_k^0(\omega))1_{\{T_k^0(\omega)\in\Omega\}}\\
&\quad+\textstyle\sum_{k=1}^{K}(n_k+1)\Psi_k(T_0^k(\omega))P(T_0^k(\omega))1_{\{T_0^k(\omega)\in\Omega\}},
\end{aligned}$$

The difference of the balance equation between Case II and Case I lies in that $\Psi_1(\omega)$ is state dependent in Case II, while Ψ_1 is state independent in Case I.

Let ν_1 and ν_2 be the maximum and minimum average transmission rates of TCP calls given by

$$\begin{aligned}
\nu_1 &= \sup_{p_T^l\le p(t)\le p_T^u}\left\{E\{TCP\}=\frac{1}{RTT}\sqrt{\frac{3}{2p(t)}}\right\},\\
\nu_2 &= \inf_{p_T^l\le p(t)\le p_T^u}\left\{E\{TCP\}=\frac{1}{RTT}\sqrt{\frac{3}{2p(t)}}\right\},
\end{aligned}$$

where p_T^l and p_T^u are defined in Table 5.1.

Theorem 1 not only shows us the insensitive property of Case II, but provides us the approach to calculate the insensitive blocking probability. We can assume the workload is exponentially distributed, and the system degenerates from GSMP to a Markov process. Thus, we can obtain the stationary probability measure by solving the balance equations, and the obtained results is the guaranteed to be the same for any arbitrary distribution of workload by Theorem 1. However, by this means, the close form solution is rather complicated and has to be solved by numerical approach as in [98, 130].

For the purpose of simplicity, we present an quite simple approximation of $P(\omega)$ in terms of the close form given by

$$P(\omega)=\frac{1}{\Phi(\Omega)}\prod_{k=1}^{K}\frac{\rho_k^{n_k}}{n_k!},\qquad \omega\in\Omega_k, \tag{5.15}$$

where the normalized factor $\Phi(\Omega)$ can be solved by appealing (5.12),

$$\Phi(\Omega)=\sum_{\omega\in\Omega}\prod_{k=1}^{K}\frac{\rho_k^{n_k}}{n_k!}.$$

and $\rho_k = \Lambda_k/\Psi_k, \quad k = 2, ..., K$ be the traffic intensity for each class of flows, and for $k = 1$,

$$\begin{aligned} \rho_1 &= \frac{\Lambda_1\xi}{\nu_1} 1_{\left\{\sum_k \rho_k \le \frac{\tilde{C}}{2}\right\}} + \frac{\Lambda_1\xi}{\nu_2} 1_{\{\sum_k \rho_k \ge \tilde{C}\}} \\ &\quad + \frac{\Lambda_1\xi}{\nu_3} 1_{\left\{\frac{\tilde{C}}{2} < \sum_k \rho_k < \tilde{C}\right\}}, \end{aligned} \tag{5.16}$$

where $\nu_3 = \nu_1 - \frac{\sum_k \rho_k - \tilde{C}/2}{\tilde{C}/2}(\nu_1 - \nu_2)$.

The idea behind such approximation is that when the arrival rate is small, the traffic intensity of class 1 is given by $\rho_1 = \Lambda_1\xi/\nu_1$ and when the arrival rate is large, $\rho_1 = \Lambda_1\xi/\nu_2$. In the middle, we use linear approximation for the corresponding traffic intensity.

Then, the blocking probability P_b^k of Case II can be obtained by (5.13), which is shown to be quite accurate by our simulations (cf. Figures 5.18 to 5.21).

5.5.4 Insensitivity Discussion

In this section, we discuss that the assumptions in our analysis for both cases can be greatly relaxed. It is well known that the Poisson process is a good enough approximation for the arrival process [27]. However, the exponential assumption for the service distribution is not realistic in general and confines the analytical results to a limited application situations. It has to be enforced in most of cases to make the analysis go through and this becomes the reason that the exponential distribution has to appear somewhere in the existing literature [9, 129, 98, 7, 52, 89, 191, 192]. For instance, in our analysis, for Case I, we assume the service holding time is exponentially distributed, and for Case II, we assume the amount of data to be transmitted is exponentially distributed. Under these exponential distribution assumptions, the stochastic process for the system can be formulated as a Markov Chain and the consequence analysis can be carried out.

However, with the advances of queueing theory, the assumption of service distribution in both Case I and Case II, which is exponential distribution, can be relaxed to the any arbitrary distributions with the same mean. This is so called "insensitivity" property in [174, 173, 33, 177, 27, 28]. Specifically, in [177, 27, 28], the author gives the necessary and sufficient condition for a queueing networks, i.e., if the considered networks is partial balanced, then the networks has insensitivity property, which is also named "Whittle Network". Since in both case I and II, the network model we consider is a special case of

Whittle network. Thus, the results we obtained in Sections 5.5.2 and 5.5.3 can be applied to any arbitrary distributions with the same mean.

In particular, for Case I, the exponential service time with mean Ψ_k can be extend to any arbitrary distributions with mean Ψ_k, and we verify this in our simulations (cf. Figure 5.17). Similarly, for Case II, the exponential distribution of the amount of data to be transmitted can be applied to any distributions (cf. Figure 5.23).

In other words, the performance measures of both Case I and Case II are insensitive to the service distribution and is determined only through its mean. The performance measures obtained under exponential service distribution can be applied to any other service distributions, which greatly expand the applicable situations of our results and are in accordance with the fact that the Erlang types of formula work well in practice.

5.6 Simulation Results

In this section, we present extensive the simulation results of our TCP-AIMD aware CAC scheme and compare the performance under different configurations. We simulate Case I and Case II as described in Section 5.5.

The system parameters used in the simulation are as follows. We consider two classes of calls and note that our scheme are quite general and could be effectively applied to the system with more classes. Class 1 is the TCP call with transmission rate adjusted by the packet loss probability $p(t)$ (the value of $p(t)$ is set according to different scenarios) and Class 2 is the voice-type call with average transmission rate of $E\{\text{voice}\} = 2$ (pkt/sec), which could correspond to the voice transmission rate ranging from 2 kbps and 8 kbps by choosing different packet sizes [2]. The call arrival processes for both classes are characterized by Poisson process with mean rate λ_1 and λ_2, respectively, and the transmission time is exponentially distributed with mean $1/\mu_1$ and $1/\mu_2$, respectively.

The performance metrics consist of both call level and packet level. Specifically, for call level, we consider call blocking probability [8] and call level throughput (call/min). As to packet level, link utilization is used as performance measure.

[8] In order not to distract from our focus, we do not consider the handoff call specially.

5.6.1 Case I: Independent Transmission Time

In this section, we present simulation results for case I: independent transmission time. First, we compare the performance without TCP-AIMD aware and that of with TCP-AIMD aware. Without TCP-AIMD aware, the system is not able to tell whether the call is TCP call or not. The TCP call will keep increasing without any control from the BS until the capacity overflow and packet loss occurs. Once the packet loss occurs, the system is treat this signal as the system capacity is under fully utilized and will blocking any incoming call requests. Figure 5.8 shows the call blocking probabilities of Class 1 and Class 2 for without and with TCP-AIMD aware as the increase of arrival rate of Class 1, λ_1 (λ_2 is fixed and equal to 2). The solid lines (resp. the dashed lines) are the blocking probabilities of with TCP-AIMD aware (resp. without TCP-AIMD aware). We can see that the blocking probabilities with TCP-AIMD aware is much smaller than those of without TCP-AIMD aware, respectively. With TCP-AIMD aware, the system is able to regulate the average throughput of TCP calls and reduce the resources allocated to the TCP calls to accommodate the new call requests, which results in a lower blocking probabilities for both classes.

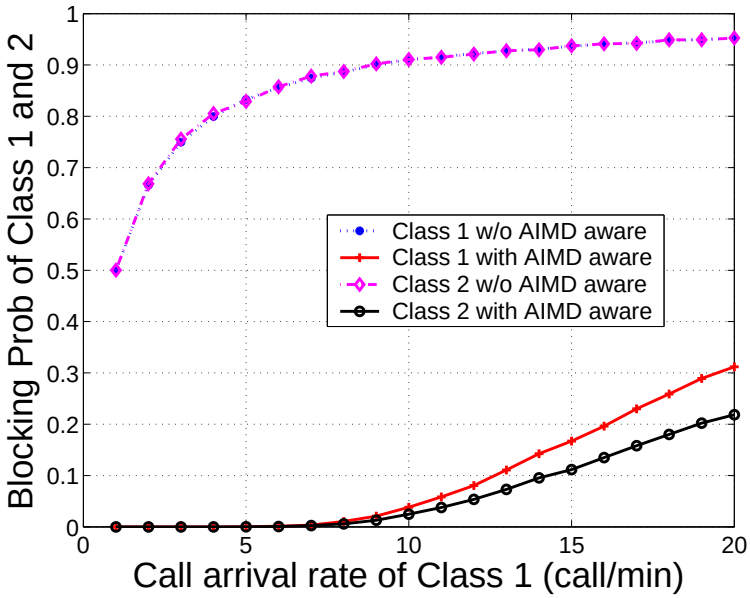


Figure 5.8: The blocking probability comparison for with and without TCP AIMD control.

The call level throughput comparison of both Classes is plotted in Figure 5.9.

As the increase of λ_1, the throughput of with TCP-AIMD aware are greater than that of without TCP-AIMD aware. This is reasonable because the blocking probabilities of with TCP-AIMD aware are smaller than those of without TCP-AIMD aware as shown in Figure 5.8. Close look at Figure 5.9 reveals that the throughput of Class 1 of with TCP-AIMD aware increases, while that of Class 2 decreases slightly. This is because in this set of simulation, we set the arrival rate of Class 2 unchanged over time and only increase the arrival rate of Class 1. Hence, the throughput of Class 1 increases with the increase of arrival rate of Class 1 and the throughput of Class 2 decreases a little bit due to more Class 1 calls in the system competing the resources. In the later simulation, we will show what if we increase both λ_1 and λ_2.

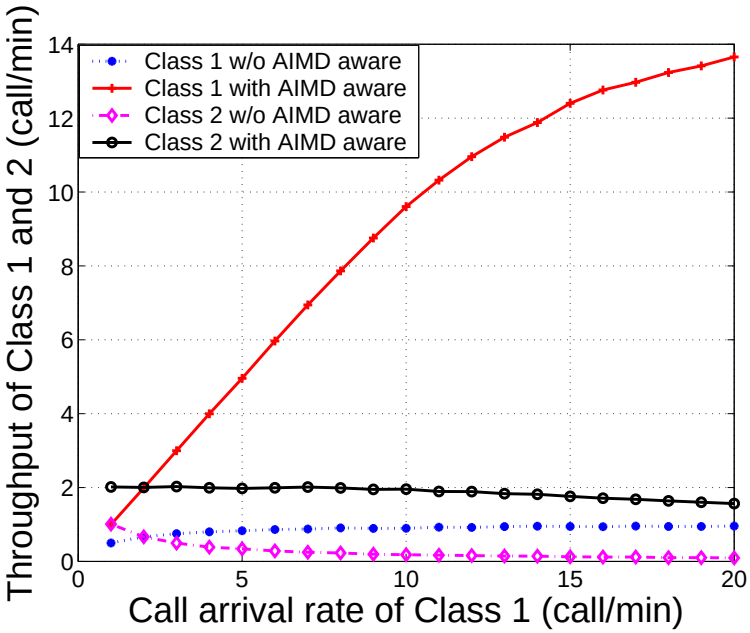


Figure 5.9: The throughput comparison for with and without TCP AIMD control.

Figure 5.10 shows the packet level performance for without and with TCP-AIMD aware, respectively. With TCP-AIMD aware, the system yield much higher link utilization than that of without TCP-AIMD aware, because with TCP-AIMD aware, the system could allow more calls to share the system resource and thus increases the resource utilization. When the arrival rate of TCP call is too small, the number of TCP calls in the system is small. The link utilization of with TCP-AIMD aware is similar to that of without TCP-

AIMD aware. For the purpose of fair comparison, we ignore the lower arrival rate for link utilization.

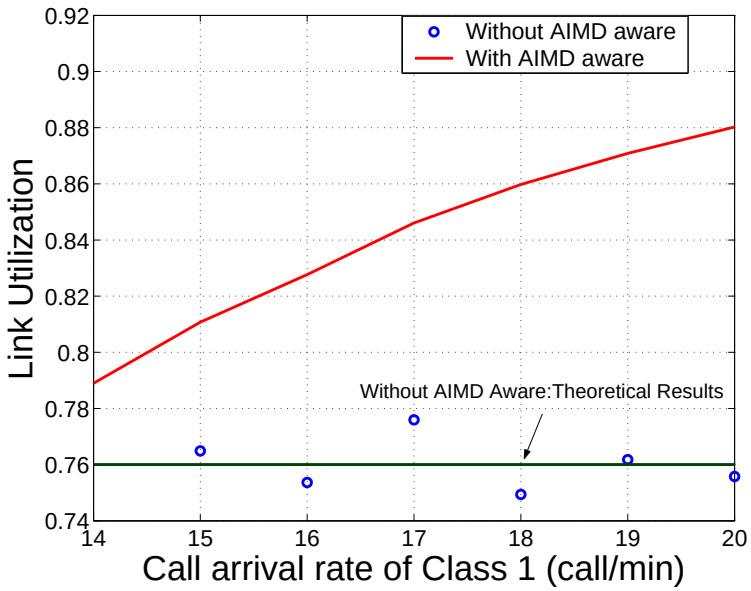
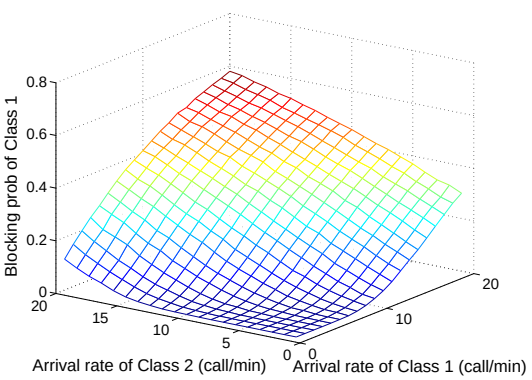


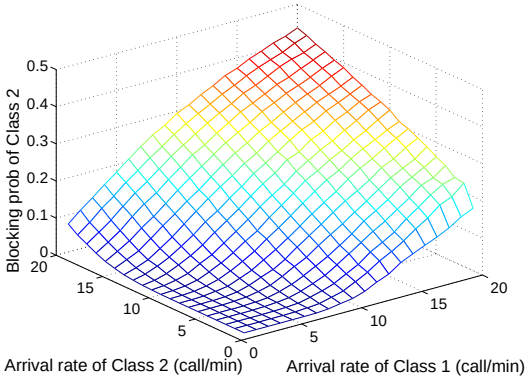
Figure 5.10: The link utilization comparison for with and without TCP AIMD control.

Remark: *Figures 5.8, 5.9 and 5.10 show that the TCP-AIMD aware is necessary for CAC and can significantly improve the system performance.*

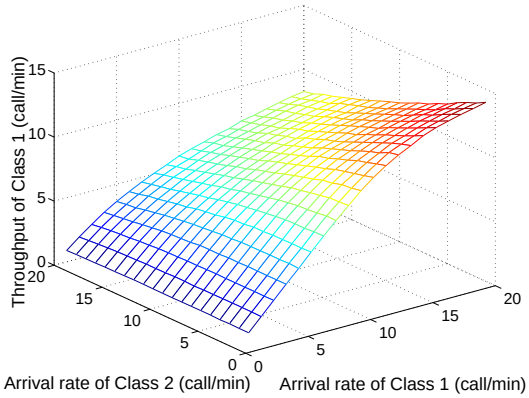
The previous three figures show the performance with only one-dimensional increase of λ_1. Figure 5.11 illustrate the performance of TCP-AIMD aware with two-dimensional increase of both λ_1 and λ_2. The lines of TCP-AIMD aware in Figures 5.8, 5.9 and 5.10 are the special case of Figure 5.11 with $\lambda_2 = 2$, respectively. We can tell the contributions to the performance metrics by each Class. For example, the blocking probabilities of both Classes will increase in the arrival rates of either λ_1 and λ_2. This is reasonable, because the increase of either Class will push the system more close to the saturated status (i.e., capacity is near fully utilized), where the packet loss occurs more frequently. The link utilization shares the same trend as the blocking probabilities, while the throughput yields different trends from the blocking probabilities. The throughput of Class 1 (resp. 2) will increase in the arrival rate of Class 1 (resp. 2), but decrease in the arrival rate of Class 2 (resp. 1), because the more arrivals of Class 1 in the system will increase the chance of Class 1 to be served by the system and thus decrease the likelihood of Class 2 to be served.



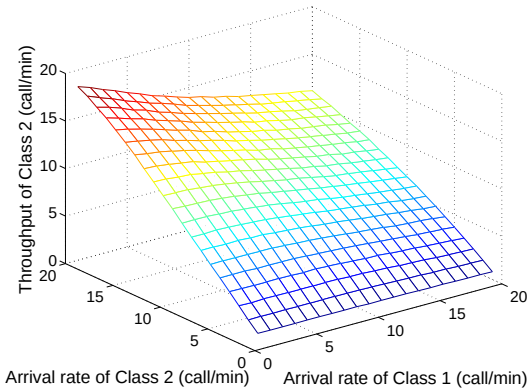
(a) Blocking Prob. of Class 1.



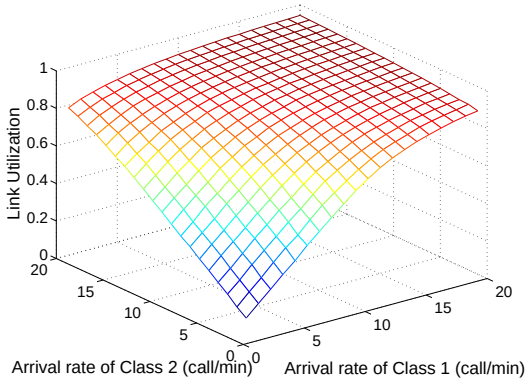
(b) Blocking Prob. of Class 2.



(c) Throughput of Class 1.



(d) Throughput of Class 2.



(e) Link utilization.

Figure 5.11: The performance metrics of TCP-AIMD aware with the two-dimensional increase of both λ_1 and λ_2.

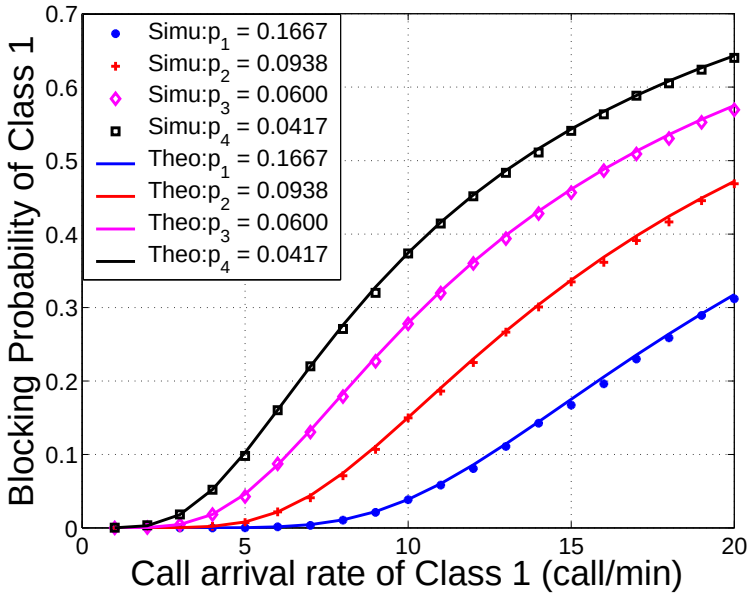


Figure 5.12: The blocking probability of Class 1 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively.

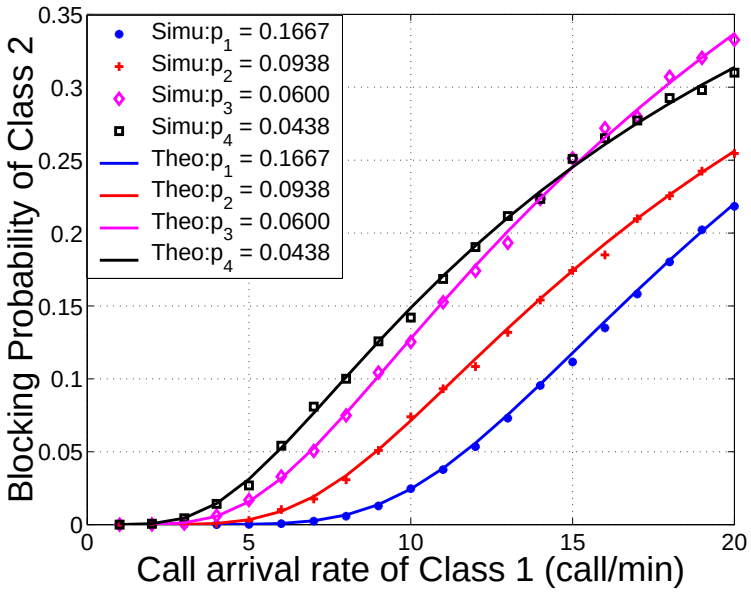


Figure 5.13: The blocking probability of Class 2 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively.

Next, we show the impact of different packet loss probability p on the performance metrics. We know that different p_i corresponds to different average throughput of TCP calls denoted by $E\{\text{TCP}\}_i$. Figures 5.12 and 5.13 display the blocking probabilities with the increase of arrival rate of Class 1, while keep the arrival rate of Class 2 fixed i.e.,$\lambda_2 = 2$. We plot both the simulation results (the legends with markers but no lines) and the theoretical results, i.e., P_b^k derived based on (5.14), (the legends with lines but no markers) for the purpose of comparison. We can see the simulation results are in good agreement with the theoretical results for both Figures. Moreover, Figure 5.12 shows a clear trend that with the decrease of p (i.e., the increase of average TCP throughput), the blocking probability of Class 1 increases. However, this trend is violated in Figure 5.13 when the arrival rate is larger than 15 (call/min) (cf. p_3 and p_4). There is a trade off between p_3 and p_4, i.e., the blocking probability of Class 1 of p_3 is smaller than that of Class 1 of p_4 in Figure 5.12, but the blocking probability of Class 2 of p_3 is larger than that of Class 2 of p_4 in Figure 5.13. Even though the underlining reason is not quite clear yet, the results indicate that there may exist tradeoff among different Classes under some situations. However the overall trend is that the blocking probabilities increase with the decrease of p.

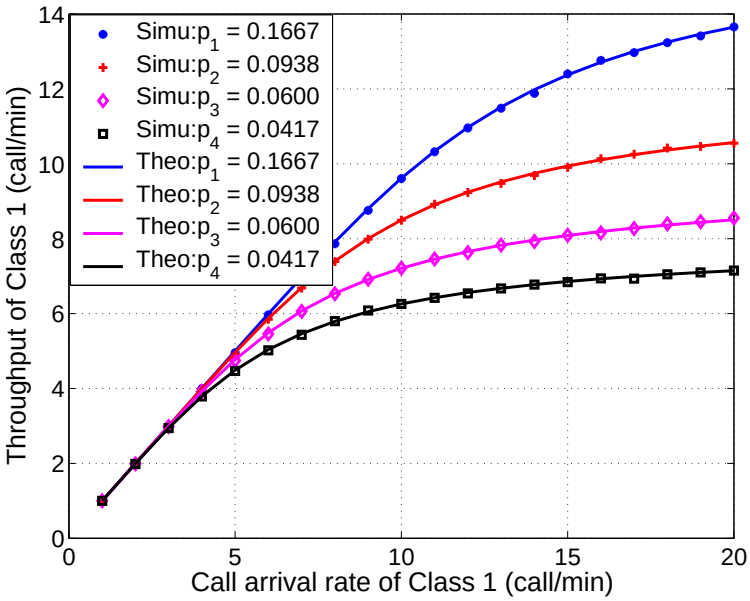


Figure 5.14: The throughput of Class 1 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively.

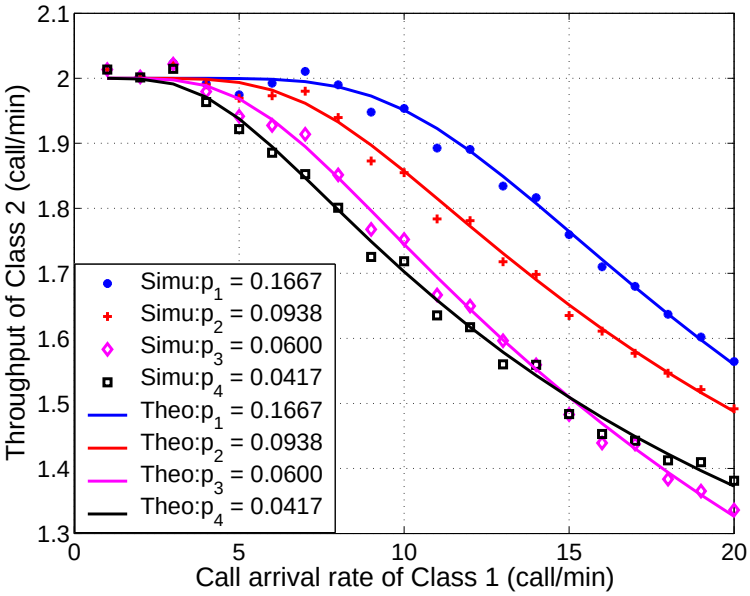


Figure 5.15: The throughput of Class 2 for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively.

Figures 5.14 and 5.15 plot the throughput of Class 1 and Class 2 in the arrival rate of Class 1. Similarly to the blocking probability in the previous two figures, the simulation results and theoretical results are provided. Again, we can see the simulation results match the theoretical results. The reason is that since the throughput is equal to the arrival rate times the non-blocking probability, i.e., $\lambda \times (1 - P_b^k)$, Figures 5.14 and 5.15 are the consequence of Figures 5.12 and 5.13, where the same reasoning is applied.

The simulation results of link utilization under different p are plotted in Figure 5.16 indicating that the link utilization is increased as the decrease of p. This is because in Case I, where the channel holding time is not related to the transmission rate, the decrease of p implies higher transmission rate and thus contributes to higher link utilization.

Remark: *Figures 5.12 to 5.16 show that by increase average packet loss probability of TCP call p, the system can obtain the trend of reduced call blocking probability, increased throughput at the slightly expense of link utilization for Case I. As we shall see in Case II, by adjusting p, the system maintains all the good performance measures.*

In [97, 75], the authors argue that the lognormal distribution of transmission time is more accurate than exponential distribution in practice. In fact, our analysis shows that

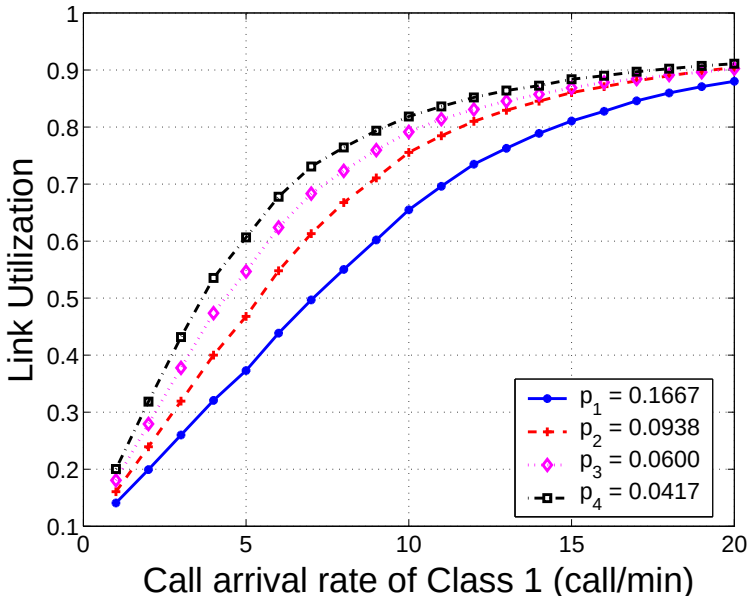
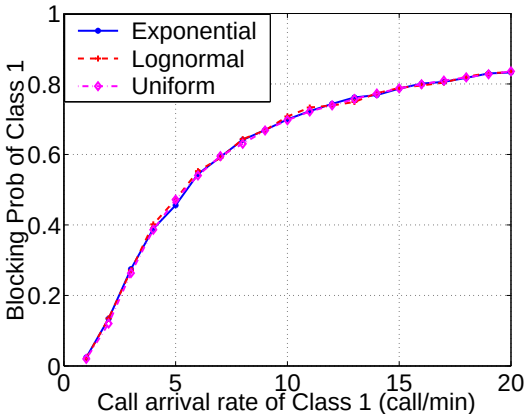


Figure 5.16: The link utilization for different TCP AIMD control: $p_1 = 0.1667, p_2 = 0.0938, p_3 = 0.0600, p_4 = 0.0417$ corresponds to $E\{\text{TCP}\}_1 = 3, E\{\text{TCP}\}_2 = 4, E\{\text{TCP}\}_3 = 5, E\{\text{TCP}\}_4 = 6$ pkt/sec, respectively.

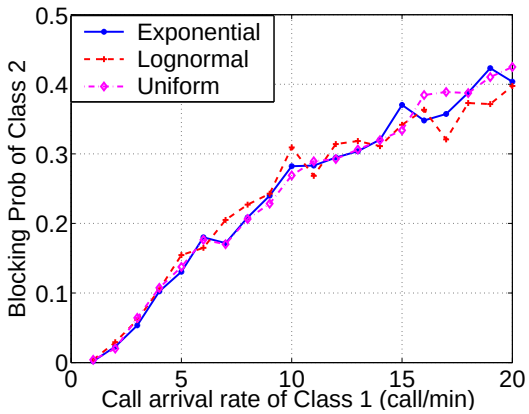
the performance under Case I is irrelevant to the distribution of transmission time. In other words, for any arbitrary distribution of transmission time, the performance maintains the same as long as the arrival process is Poisson process, which is so call insensitivity property [174, 173, 33]. In Figure 5.17, we compare the performance under three different distributions of transmission time. The results indicate that all the performance metrics are almost the same under three distributions, i.e., call blocking probabilities of Class 1 and Class 2, throughput of Class 1 and Class2, and link utilizations.

5.6.2 Case II: Dependent Transmission Time

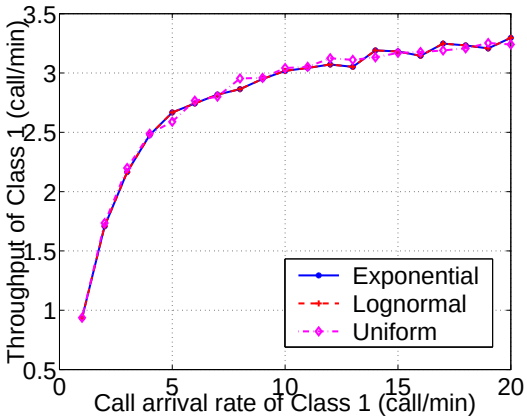
In this section, we present simulation results for Case II, i.e., the transmission time is dependent on the transmission rate, where the packet loss probability of a TCP call may change over time and is denoted by $p(t)$. The variation of $p(t)$ affects the average throughput of a TCP call (pkt/sec) and hence the transmission time of the TCP call. This becomes the difference between Case II and Case I in that for Case I, the changing of $p(t)$ does not affect the transmission time, i.e., the transmission time is independent of $p(t)$ and hence transmission rate.



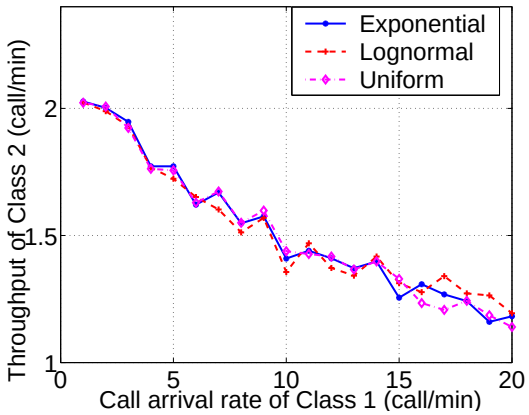
(a) Blocking Prob. of Class 1.



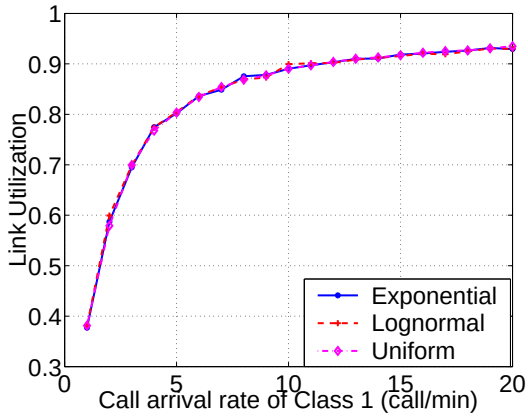
(b) Blocking Prob. of Class 2.



(c) Throughput of Class 1.



(d) Throughput of Class 2.



(e) Link utilization.

Figure 5.17: The performance comparison under different distributions of transmission time: Exponential distribution; Lognormal distribution; Uniform distribution. $p = 0.1667$.

In Case II, three strategies are compared in our simulation: (i) $p(t) = 0.1667$ is fixed during the transmission for all TCP calls and the corresponding average TCP throughput $E\{\text{TCP}\} = 3$ pkt/sec; This strategy tries to use smaller resource for each TCP call and allow the system to be able to accommodate more calls simultaneously at the expense of longer transmission time for each TCP call. (ii) $p(t) = 0.0234$ and $E\{\text{TCP}\} = 8$ pkt/sec. This strategy uses high transmission rate to reduce the transmission time and henceforth tends to make the system capable to serve the incoming calls. (iii) $p(t) \in [0.0234, 0.1667]$. The system will dynamically change $p(t)$ following TCP-AIMD aware algorithm as proposed in Section 5.3. With dynamic $p(t)$, the system takes the advantages of the previous two strategies and this will be verified by the following simulation results.

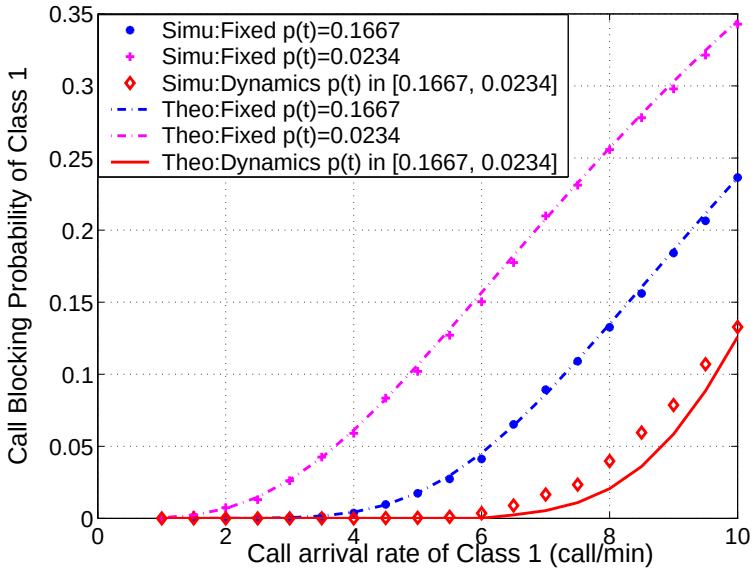


Figure 5.18: The link utilization for different arrival rate: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

The call blocking probabilities of Class 1 and Class 2 are displayed in Figures 5.18 and 5.19. The simulation results are plotted with markers but without lines, and the theoretical results are plotted with lines but without markers. In particular, for fixed $p(t) = 0.1667$ and $p(t) = 0.0238$, the theoretical results can be derived based on Case I in (5.14), and the theoretical results for dynamics $p(t) \in [0.0234, 0.1667]$ is obtained based on Case II's $P(\omega)$ in (5.15). The observation is that there is a tradeoff between $p(t) = 0.1667$ and $p(t) = 0.0238$. The blocking probability of Class 1 of $p(t) = 0.1667$ is smaller than that of $p(t) = 0.0238$, while the blocking probability of Class 2 of $p(t) = 0.1667$ is greater that of

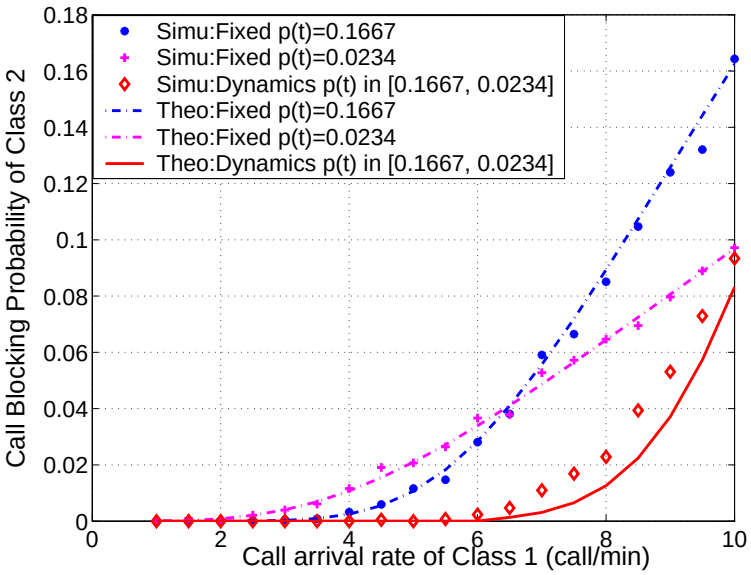


Figure 5.19: The link utilization for different arrival rate:(i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

$p(t) = 0.0238$ when the arrival rate is larger than 7 (call/min). This implies that strategies (i) and (ii) have their own advantages and one will outperform the other in some situations. However, our proposed dynamic $p(t)$ yields better performance than both (i) and (ii).

The same observation holds good for the throughput of Class 1 and Class 2 shown in Figures 5.20 and 5.21. Figure 5.22 shows the link utilization of the three strategies. Our dynamic $p(t)$ scheme produce the highest link utilization and this implies better packet level performance of our scheme in Case II.

Like Case I, we simulate these three scheme under different distributions of workload for Case II. In addition to exponential distribution, the amount of data to be transmitted by TCP calls follows lognormal and uniform distributions as well. For all the performance metrics, three distributions yield quite close results. This enforces our belief that the performance is only determined by the Poisson arrival process and irrelevant to the service processes [174, 173].

Remark: *The proposed dynamic $p(t)$ schemes yields better performance in Case II. In other words, with TCP-AIMD aware CAC scheme will improve the system QoS with regard to both call level and packet level.*

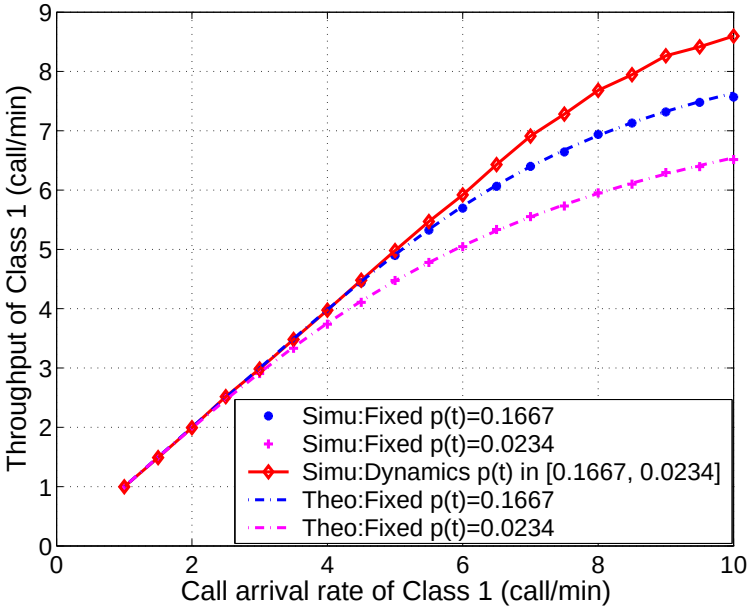


Figure 5.20: The link utilization for different arrival rate:(i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

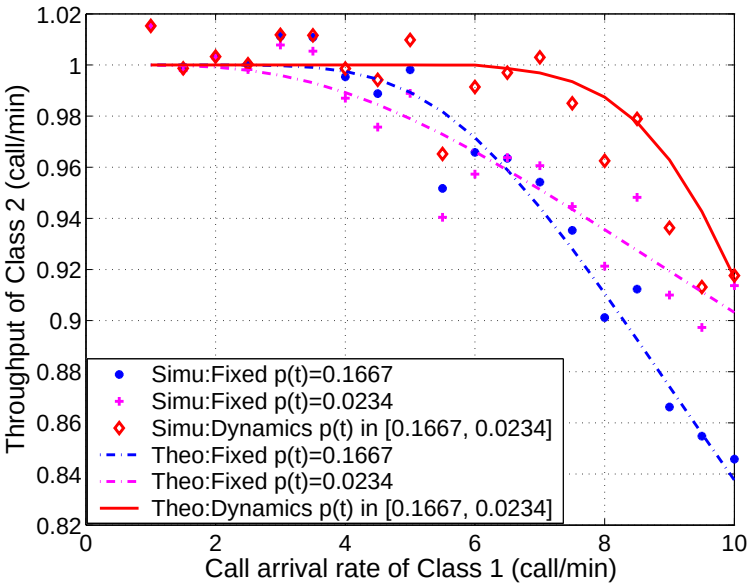


Figure 5.21: The link utilization for different arrival rate: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

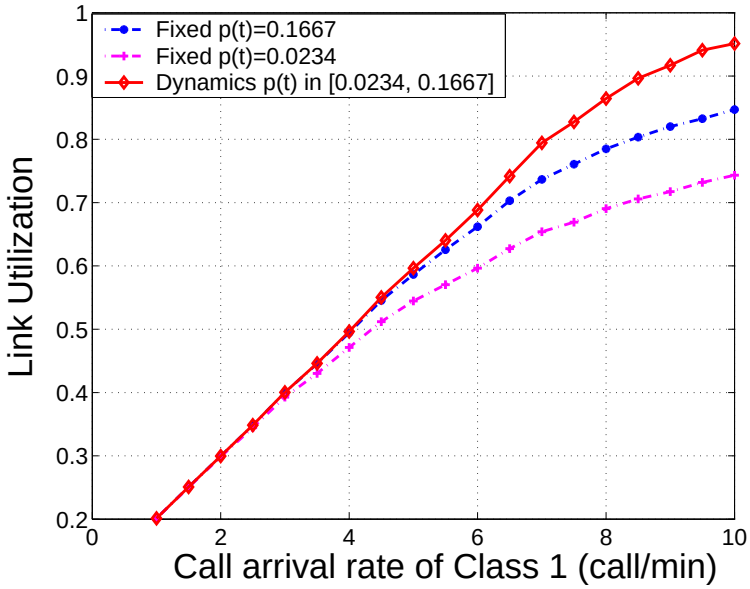
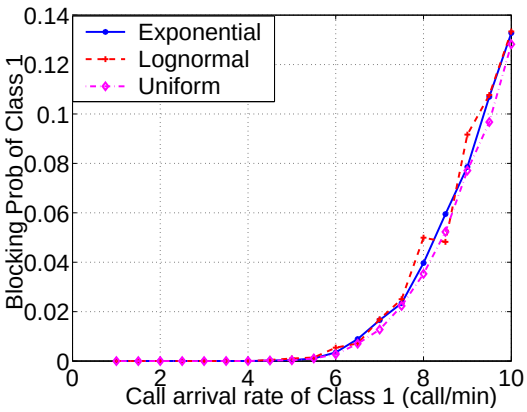


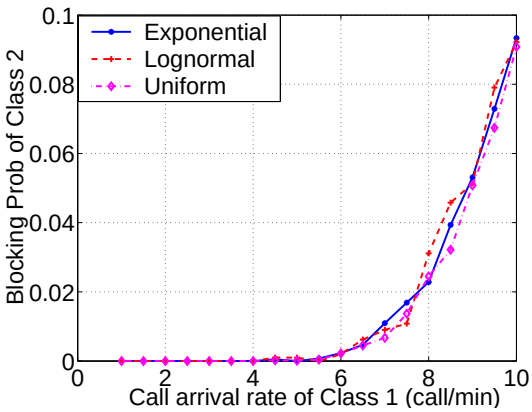
Figure 5.22: The link utilization for different arrival rate: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

5.7 Conclusion

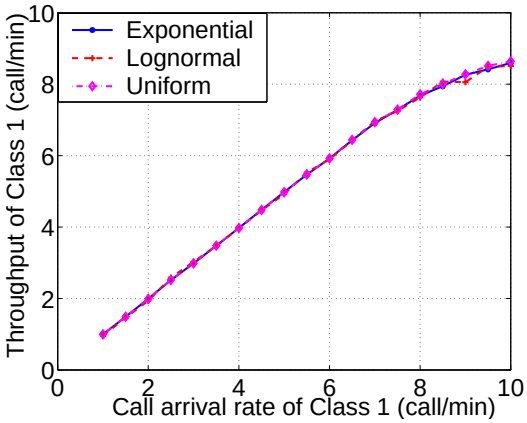
Traditional Call Admission Control (CAC) schemes only consider call level performance and are believed to be sufficient for the circuit-switched wireless network. Since the future wireless network will become packet-switched, the packet level performance cannot be ignored. This is especially true when the TCP-type of applications are running over such packet-switched wireless networks, because TCP-AIMD congestion control algorithm will exhaust all the available resource until packet loss occurs. In order to regulate the TCP applications to friendly coexist with other types of services, we therefore propose a TCP-AIMD aware CAC scheme based on a two-level framework that takes both call level and packet level performance into account. The performance of our scheme is analyzed at two levels, respectively, and the interaction between two levels is disclosed. Our extensive simulations results show that the proposed scheme can significantly improve the system performance with respect to call blocking probability (call level metric), throughput (call level metric), and link utilization (packet level metric), which are in good agreement with the theoretical analysis.



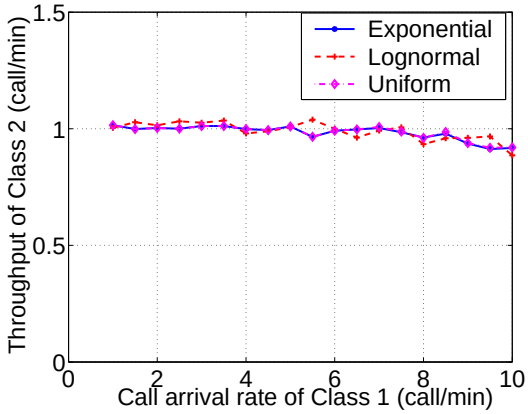
(a) Blocking Prob. of Class 1.



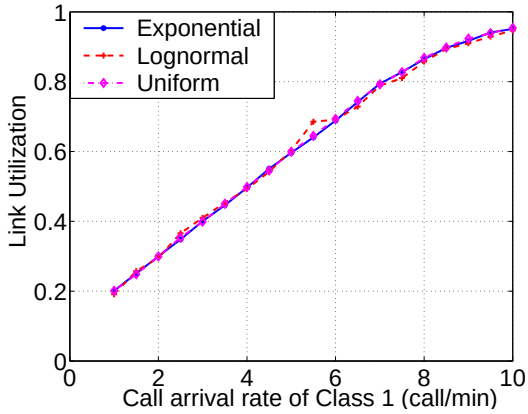
(b) Blocking Prob. of Class 2.



(c) Throughput of Class 1.



(d) Throughput of Class 2.



(e) Link utilization.

Figure 5.23: The performance comparison under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution. Dynamic $p(t) \in [0.0234, 0.1667]$.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this report, we study the local and global stability of a TCP-newReno/RED system with many flows. The existing local stability literature is designed for TCP-Reno not for TCP-newReno, and ignores the relationship between RED slope and the round trip time, and thus results in a more stringent stability criterion. As to the global stability, traditional approaches rely on constructing a suitable Lyapunov function, based on which the global stability of the system can be obtained. However, for a typical TCP/RED system with many flows, it is too difficult to directly find out the Lyapunov function, because the entire system dynamics becomes too complicated to be tracked. Instead, in this paper, we present a normalized model for the TCP/RED with many flows. This normalized model is able to capture the essential system dynamics, but simple enough to allow numerical analysis. Based on the normalized model, we derive the local stability criterion in discrete time, which takes the relationship between RED slope and round trip time into account. Our stability criterion yields large stability region compared to the existing results. Moreover, we examine the global stability of a system provided that it is locally stable. Our results show that for any given locally-stable TCP/RED system, we can make it globally stable by properly choosing some of its parameters. Our extensive numerical analysis also suggests that in most cases, any locally stable TCP/RED system becomes globally stable, and this

observation becomes more accurate as the number of flows increases. Finally, we extend our model to general AIMD case and heterogeneous RTTs' case, respectively, which allow us to investigate a more general setting.

6.2 Future Work

Essentially, TCP is a rate control protocol (or resource allocation) for networks, and the rate control related issue is one of the core problems in networking area due to the limited resource. Based on the present thesis, we can go further and exploit a much wider field of rate control and resource allocation, which are important, interesting, and challenging. We identify some problems as follows.

1. *Optimal stochastic rate allocation for wireless cellular networks*: We know that the real wireless network is full of uncertainties like the physical channel condition, the number of users in the system, etc. These uncertainties implies that the optimal rate allocation scheme should depend on the system state, rather than independent on the system state. In other words, according the system state (e.g., physical channel condition, the number of users in the system), the rate allocation strategy should adjust to cater the corresponding system state, i.e., the rate allocation is stochastic, instead of deterministic. It is worth asking what is the optimal stochastic rate allocation scheme for wireless cellular networks? The optimal here is in the sense that the rate allocation scheme will make the system achieve the maximal system utility, which incorporates all the interested performance measures like lower energy consumption, smaller blocking probability, higher throughput, etc.

2. *Admission and rate control for ad-hoc and sensor networks*: Due to the enormous success of ad-hoc and sensor networks, the extensive research has been conducted to improve the efficiency of those networks. It is well known that compared to the infra-structure networks like cellular networks, the rate control for ad-hoc networks is more difficult due to the flexibility of the structure, more diverse mobility, and the uncertainty of the connectivity. Especially, for sensor networks, the resource is limited by the scarce battery life, CPU capacity, and transmitter power etc, and thus the efficiency for the rate control and resource allocation become more imperative.

3. *Incentive compatible resource allocation for wireless networks*: The existing results regarding the resource allocation are obtained by pricing method either from the system point view or from the users' perspective. The incentive compatible resource allocation is going to achieve the double-side optimal revenue from both the system and the end users. Unlike the existing single-side optimal resource allocation, the incentive compatible resource allocation requires the cooperation of the system and the end users. Hence, the resulting resource allocation scheme will yield much better social welfare and the revenue for both the system and the end users. The so called incentive compatible is the sense that if the end users do follow the designed contract (or scheme), the their results are only worse than the optimal ones.

Bibliography

[1] http://www.indiana.edu/ rats/research/hsi/index.shtml.

[2] UMTS. In *Http://www.3GPP.org*.

[3] The Network Simulator: ns-2. In *http://www.isi.edu/nsnam/ns/*, 2004.

[4] A. Acharya and A. Rangarajan. Early Regulation of Unresponsive Flows. In *ICNP 99*, July 1999.

[5] M. Adler, J. Cai, J. K. Shapiro, and D. Towsley. SAM: Estimation of congestion price using probabilistic packet marking. *Preprint*, 2002.

[6] M. H. Ahmed. Call Admission Control in Wireless Networks: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 7(1):49–68, 2005.

[7] M. H. Ahmed and H. Yanikomeroglu. SINR Threshold Lower Bound for SINR-based Call Admission Control in CDMA Networks with Imperfect Power Control. *IEEE Communications Letters*, 9(4):331–333, April 2005.

[8] O. B. Akan, J. Fang, and I. F. Akyildiz. TP-Planet: A Reliable Transport Protocol for Interplanetary Internet. *IEEE JASC*, 22(2):348–361, Feb 2004.

[9] R. G. Akl, M. V. Hegde, and M. Naraghi-Pour. Mobility-Based CAC Algorithm for Arbitrary Call-Arrival Rates in CDMA Celluar Systems. *IEEE Transactions on Vehicular Technology*, 54(2):639–651, March 2005.

[10] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke. GridFTP protocol specification. In *GGF GridFTP Working Group Document*, Sept 2002.

[11] T. Alpcan and T. Basar. Global Stability Analysis of an End-to-End Congestion Control Scheme for General Topology Networks with Delay. In *IEEE Conference on Decision and Control (CDC)*, Maui, HI, Dec. 2003.

[12] T. Alpcan and T. Basar. Global stability analysis of an end-to-end congestion control scheme for general topology networks with delay. *Elektrik*, 2004.

[13] T. Alpcan and T. Basar. A globally stable adaptive congestion control scheme for Internet-style networks with delay. *IEEE/ACM Transactions on Networking*, to appear, 2005.

[14] E. Altman, K. Avrachenkov, C. Barakat, A. A. Kherani, and B. J. Prabhu. Analysis of MIMD congestion control algorithm for high speed networks. *Computer Networks*, 48(6):972–989, Aug 2005.

[15] D. Anick, D. Mitra, and S. Songdhi. Stochastic Theory of Data-handling Systems with Multiple Sources. *Bell Systems Technical Journal*, 61, 1982.

[16] F. M. Anjum and L. Tassiulas. Balanced-RED: An Algorithm to Achieve Fairness in the Internet. In *Proceedings of IEEE Infocom 1999*, March 1999.

[17] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In *Proceedings of ACM SIGCOMM*, Portland, OR, 2004.

[18] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48–53, May 2001.

[19] S. Athuraliya and S. H. Low. Optimization Flow Control II: Implementation. *Submitted for publication*, 2000.

[20] B. Atkin and K. P. Birman. Evaluation of an Adaptive Transport Protocol. In *Proceedings of IEEE Infocom 2003*, March 2003.

[21] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. In *IETF RFC 2702*, Sept 1999.

[22] J. Aweya, M. Ouellette, D. Y. Montuno, and A. Chapman. A Control Theoretic Approach to Active Queue Management. *Computer Networks*, 36, 2001.

[23] J. Aweya, M. Ouellette, D. Y. Montuno, and A. Chapman. An Optimization-oriented View of Random Early Detection. *Computer Communications*, 2001.

[24] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *15th International Conference on Distributed Computing Systems*, 1995.

[25] R. K. Balan, B. P. Lee, and K. R. R. Kumar. Explicit Allocation of Best Effort Packet Delivery Service. In *IEEE/ACM Transactions on Networking*, Aug 1998.

[26] U. Bodin, O. Schelen, and S. Pink. Load-tolerant Differentiation with Active Queue Management. In *CCR 2000*, July 2000.

[27] T. Bonald and A. Proutiere. Insensitivity in Processor-sharing Networks. *Performance Evaluation*, 49:193–209, Oct 2002.

[28] T. Bonald and A. Proutiere. Insensitivity Data Sharing in Data Networks. *Queueing Systems*, 44:69–100, Oct 2003.

[29] M. Bottigliengo, C. Casetti, C. F. Chiasserini, and M. Meo. Short-term Fairness for TCP Flows in 802.11b WLANs. In *submitted for publication*, 2002.

[30] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. In *RFC 2309*, 1998.

[31] A. Bria, F. Gessler, O. Queseth, R. Stridh, M. Unbehaun, J. Wu, and J. Zander. 4th Generation Wireless Infrastructures: Scenarios and Research Challenges. *IEEE Personal Commumicatons*, pages 25–31, Dec 2001.

[32] K. Brown and S. Singh. M-TCP: TCP for Mobile Celluar Networks. *ACM Computer Communications Review*, 27(5):19–43, 1997.

[33] D. Y. Burman, J. P. Lehoczky, and Y. Lim. Insensitivity of Blocking Probabilities in A Circuit-Switching Network. *Journal of Applied Probability*, 21:850–859, 1984.

[34] D. Towsley C. Hollot, V. Misra and W. Gong. A Control Theoretic Analysis of RED. In *Proceedings of IEEE Infocom 2001*, July 2001.

[35] D. Towsley C. Hollot, V. Misra and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proceedings of IEEE Infocom 2001*, July 2001.

[36] D. Towsley C. Hollot, V. Misra and W. Gong. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. *IEEE Transactions on Automatic Control*, 47(6):945–959, June 2002.

[37] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Lantecy. In *Proceedings of IEEE Infocom 2000*, July 2000.

[38] K. Cho. ATLQ: an Beta Release of RED for BSD Unix. In *http://www.icir.org/floyd/red.html*.

[39] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith. Tuning RED for Web Traffic. In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[40] Cisco. Cisco's RED implementations include Distributed Weighted Random Early Detection. In *http://www.icir.org/floyd/red.html*.

[41] D. D. Clark and W. Fang. An Optimization-oriented View of Random Early Detection. *Computer Communications*, 2001.

[42] M. Crovella and A. Bestravos. Self-similarity in World Wide Web Traffic: Evidence and Possible Cause. In *Proceedings of ACM SIGMETRICS*, June 1996.

[43] S. Deb, S. Shakkotai, and R. Srikant. Ansymptotic Behavior of Internet Congestion Controllers in a Many-Flow Regime. In *Proceedings of IEEE Infocom 2003*, April 2003.

[44] S. Deb, S. Shakkottai, and R. Srikant. Stability and Convergence of TCP-like Congestion Controllers in a Many-Flows Regime. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, April 2003.

[45] S. Deb and R. Srikant. Global Stability of Congestion Controller for the Internet. *IEEE Transactions on Automatic Control*, 48(6):1055–1060, June 2003.

[46] S. Deb and R. Srikant. Global Stability of congestion controllers for the internet. *IEEE Transactions on Automatic Control*, 48(6):1055–1060, June 2003.

[47] S. Deb and R. Srikant. Rate-Based versus Queue-Based Models of Congestion Control. In *Proceedings of ACM SIGMETRICS*, June 2004.

[48] T. Dunigan, M. Mathis, and B. Tierney. A TCP Tuning Daemon. In *Proceedings of SuperComputing: High-Performance Networking and Computing*, Nov 2002.

[49] O. Elloumi and H. Afifi. RED Algorithm in ATM Networks. In *Tech report*, June 1997.

[50] D. Eun and X. Wang. A TCP/AQM System under Aggressive Packet Marking: Model, Analysis and Simulations. In *Technical Report [TR2005-01], Electrical and Computer Engineering, North Carolina State Univeristy*, 2005.

[51] D. Y. Eun and X. Wang. Stationary Behavior of TCP/AQM with Many Flows Under Aggressive Packet Marking. In *Proceedings of IEEE ICC'2005*, May 2005.

[52] Y. Fang and Y. Zhang. Call Admission Control Scheme and Performance Analysis in Wireless Mobile Networks. *IEEE Transaction on Vehic. Technology*, 51(2):371–382, March 2002.

[53] W. Feng. ALTQ modifications for implementing Blue. In *http://www.thefengs.com/wuchang/work/blue/*.

[54] W. Feng. ns-1.1 modifications for implementing Blue and Stochastic Fair Blue. In *http://www.thefengs.com/wuchang/work/blue/*.

[55] W. Feng, D. Kandlur, D. Saha, and K. Shin. Techniques for Eliminating Packet Loss in Congested TCP/IP Networks. In *Technical Report: U. Michigan CSE-TR-349-97*, Nov 1997.

[56] W. Feng, D. Kandlur, D. Saha, and K. Shin. A Self-Configuring RED Gateway. In *Proceedings of IEEE Infocom 1999*, March 1999.

[57] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A New Class of Active Queue Management Algorithms. In *Technical Report: U. Michigan CSE-TR-387-99*, 1999.

[58] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: An Alternative Approach To Active Queue Management. In *Proc. of NOSSDAV 2001*, June 2001.

[59] W. Feng, D. Kandlur, D. Saha, and K. Shin. Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness. In *Proceedings of IEEE INFOCOM 2001*, April 2001.

[60] W. Feng, D. Kandlur, D. Saha, and K. Shin. The Blue Queue Management Algorithms. *IEEE Transactions on Networking*, 10(4), August 2002.

[61] S. Floyd. http://www.icir.org/floyd/red.html.

[62] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5):10–23, Oct 1994.

[63] S. Floyd. Ns Simulator Tests for Random Early Detection (RED) Gateways. In *Technical report*, Oct 1996.

[64] S. Floyd. A Report on Some Recent Developments in TCP Congestion Control. In *http://www.icir.org/floyd/red.html*, 2000.

[65] S. Floyd. HighSpeed TCP for Large Congestion Windows. In *IETF, Draft, draft-floyd-tcp-highspeed-01.txt*, 2003.

[66] S. Floyd. Limited Slow-Start for TCP with Large Congestion Windows. In *IETF, Draft, draft-floyd-tcp-slowstart-01.txt*, 2003.

[67] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management . *http://www.icir.org/floyd/red.html*, Aug 2001.

[68] S. Floyd, T. Henderson, and A. Gurtov. The NewReno modification to TCPs Fast Recovery Algorithm. In *RFC 3782*, April 2004.

[69] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug 1993.

[70] S. Floyd, S. Ratnasamy, and S. Shenker. Modifying TCP's Congestion Control for High Speed. In *http://www.icir.org/floyd/hstcp.html*, May 2002.

[71] A. Fox, S. d. Gribble, E. A. Brewer, and E. Amir. Adapting to Network and Client Variation via On-demand Dynamic Distillation. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 160–170, Oct 1996.

[72] C. J. Fraleigh. Provisioning Internet Backbone Networks to Support Latency Sensitive Applications. In *Ph.D. Thesis, Standford University, Department of Electrical Engineering*, June 2002.

[73] Zhenghua Fu, Haiyun Luo, Petros Zerfos, Songwu Lu, Lixia Zhang, and Mario Gerla. The Impact of Multihop Wireless Channel on TCP Performance. *IEEE Transaction on Mobile Computing*, 4(2), March/April 2005.

[74] M. Gaynor. Proactive Packet Dropping Methods for IP Gateways. In *http://www.icir.org/floyd/red.html*, Nov 1996.

[75] M. Ghaderi, R. Boutaba, and G. W. Kenward. Stochastic Admission Control for Quality of Service in Wireless Packet Networks. In *Proceedings of the 4th IFIP Networking Conference (Networking'2005)*, May 2005.

[76] M. Goyal, R. Guerin, and R. Rajan. Predicting TCP Throughput From Non-invasive Network Sampling. In *Proceedings of IEEE Infocom 2002*, July 2002.

[77] D. Grossman. New Terminology for Diffserv. In *IETF draft, draft-ietf-diffserv-new-terms-04.txt, work in progress*, Mar 2001.

[78] Y. Gu, X. Hong, M. Mazzucco, and R. L. Grossman. SABUL: A High Performance Data Transport Protocol. 2002. submitted for publication.

[79] R. Guerin and L. Gun. A Unified Approach to Bandwidth Allocation and Access Control in Fast Packet-Switched Networks. In *Proceedings of IEEE Infocom'92*, July 1992.

[80] V. Padmanabhan H. Balakrishnan, S. Seshan, and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, Dec 1997.

[81] H. Han, C. V. Hollot, Y. Chait, and V. Misra. TCP Networks Stabilized by Buffer-Based AQMs. In *Proceedings of IEEE Infocom*, March 2004.

[82] U. Herzog, L. Woo, and K. Chandy. Solution of Queueing Problems by a Recursive Technique. *IBM J. Res. Dev.*, 19:295–300, May 1975.

[83] A. Hess and G. Schafer. Performance Evaluation of AAA/Mobile IP Authentication. In *http://www-tkn.ee.tuberlin.de/publications/papers/pgts2002.pdf*, 2002.

[84] G. Holland and N. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. In *Proceedings of ACM Mobicom'99*, pages 219–230, Aug 1999.

[85] C. V. Hollot and Y. Chait. Nonlinear Stability Analysis for a Class of TCP/AQM Schemes. In *Proceedings of IEEE Conference on Decision and Control, 2001*, Dec 2001.

[86] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. A Control Theoretic Analysis of RED. In *Proceedings of IEEE INFOCOM*, Anchorage, AK, April 2001.

[87] D. Hong and D. Lebedev. Many TCP User Asymptotic Analysis of the AIMD Model. *INRIA Technical Report RR-4229*, 2001.

[88] H. Honkasalo, K. Pehkonen, M. T. Niemi, and Anne T. Leino. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. *IEEE Wireless Communications Magazine*, pages 14–18, April 2002.

[89] J. Hou and Y. Fang. Mobility-based call admission control schemes for wireless mobile network. *Wireless Communications and Mobile Computing*, 1(3):269–282, July-Sept 2001.

[90] S. Y. Hui and K. Yeung. Challenges in the Migration to 4G Mobile Systems. *IEEE Communications Magazine*, pages 54–59, Dec 2003.

[91] V. Jacobson. Congestion Avoidance and Control. *ACM Computer Communications Review*, 8(4):314–329, Aug 1988.

[92] V. Jacobson. Notes on using RED for Queue Management and Congestion Avoidance. In *talk at NANOG 13*, June 1998.

[93] M. Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation With TCP Throughput. *IEEE/ACM Trans. on Networking*, 11(4):537–549, August 2003.

[94] M. Jain, R. Prasad, and C. Dovrolis. Socket Buffer Auto-Sizing for Maximum TCP Throughput. In *Journal of Grid Computing*, volume 1, pages 361–376, 2004.

[95] R. Jain. A Delay-based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks. *ACM Computer Communications Review*, 19(5):56–71, Oct 1989.

[96] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of IEEE Infocom 2004*, March 2004.

[97] C. Jedrzycki and V. C. M. Leung. Probability Distribution of Channel Holding Time in Cellular Telephone Systems. In *Proceedings of IEEE VTC'96*, volume 1, pages 247–251, 1996.

[98] D. G. Jeong and W. S. Jeon. CDMA/TDD System for Wireless Multimedia Services with Traffic Unbalance Between Uplink and Downlink. *IEEE Journal On Selected Areas in Communications*, 17(5):939–946, May 1999.

[99] C. Jin, D. Wei, and S. H. Low. Fast TCP: Motivation Architecture Algorithm Performance. In *submited to publication*, 2004.

[100] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, H. Newman, F. Paganini, S. Ravot, and S. Singh. Fast Kernel: Background Theory and Experimental Results. In *The First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003)*, Feb 2003.

[101] R. Johari and D. Tan. End-to-end congestion control for the Internet: delays and stability. *IEEE/ACM Transactions on Networking*, 9:818–832, Dec. 2001.

[102] D. Katabi, M. Handley, and C. Rohrs. Internet Congestion Control for High Bandwidth-Delay Product Networks. In *Proceedings of ACM SIGCOMM*, Aug. 2002.

[103] F. P. Kelly. Effective Bandwidth at Multi-Class Queues. *Queueing Systems*, (9):5–16, 1991.

[104] F. P. Kelly. Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9:159–176, 2003.

[105] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, pages 237–252, 1998.

[106] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. 2002. submitted for publication.

[107] G. Kesidis, J. Walrand, and C. S. Chang. Effective Bandwidth for Multi-Class Markov Fluids and Other ATM Sources. *IEEE Transactions on Networking*, 1(4):424–428, Aug 1993.

[108] H. Khalil. Nonlinear System. *2nd Edition, Prentice Hall*, 1996.

[109] B. Kikdar, S. Kayanaraman, and K. S. Vastola. An Integrated Model for the Latency and Steady-State Throughput of TCP Connections. *Performance Evaluation*, 46(2-3):139–154, Sept. 2001.

[110] B. Kikdar, S. Kayanaraman, and K. S. Vastola. Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK. *IEEE Transactions on Networking*, 11(6):959–971, Dec 2003.

[111] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, 1992.

[112] L. Kleinrock. Queueing Systems, Volume I: Theory. *Wiley Interscience*, 1975.

[113] E. W. Knightly. H-BIND: A New Approach to Providing Statitical Performance Guarantees to VBR Traffic. In *Proceedings of IEEE INFOCOM'96*, March 1996.

[114] E. W. Knightly and H. Zhang. Traffic Characterizatioin and Switch Utilization Using a Deterministic Bounding Interval Dependent Traffic Model. In *Proceedings of IEEE INFOCOM'95*, April 1995.

[115] S. Kunniyur and R. Srikant. A time-scale decomposition approach to adaptive ECN marking. In *Proceedings of IEEE Infocom 2001*, July 2001.

[116] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue Algorithm for Active Queue Management. In *IEEE Sigcomm 2001*, Sept 2001.

[117] S. Kunniyur and R. Srikant. A Time-Scale Decomposition Approach to Adaptive ECN Marking. In *IEEE Transactions on Automatic Control*, June 2002.

[118] S. Kunniyur and R. Srikant. Designing AVQ Parameters for a General Network Topology. In *Proceedings of Asian Control Conference*, Sept 2002.

[119] S. Kunniyur and R. Srikant. Stable, Scalable, Fair Congestion Control and AQM Schemes that Achieve High Utilization in the Internet. *IEEE Transactions on Automatic Control*, pages 2024–2029, 2002.

[120] S. Kunniyur and R. Srikant. End-To-End Congestion Control: Utility Functions, Random Losses and ECN Marks. *IEEE Transactions on Networking*, Oct 2003.

[121] S. Kunniyur and R. Srikant. Stable, Scalable, Fair Congestion Control and AQM Schemes that Achieve High Utilization in the Internet. *IEEE Transactions on Automatic Control*, 48(11):2024–2028, November 2003.

[122] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. *to appear in IEEE Transactions on Networking*, 2004.

[123] S. S. Kunniyur and R. Srikant. End-to-end congestion control schemes: utility functions, random losses and ECN marks. *IEEE/ACM Transactions on Networking*, 11(5):689–702, Oct. 2003.

[124] S. S. Kunniyur and R. Srikant. An adaptive virtual queue (AVQ) Algorithm for Active Queue Management. *IEEE/ACM Transactions on Networking*, 12(2):286–299, April 2004.

[125] A. Kuznetsov. The RED implementation in Linux. In *http://www.icir.org/floyd/red.html*.

[126] G. Dullerud L. Ying and R. Srikant. Global Stability of Internet Congestion Controllers with Heterogeneous Delays. In *Proc. American Control Conference 2004.*, June 2004.

[127] T. V. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE Transactions on Networking*, 5(3):336–350, June 1997.

[128] A. Lakshmikantha, C. Beck, and R. Srikant. Robustness of Real and Virtual Queue based Active Queue Management Schemes. In *to appear in the Proc. American Control Conferences*, June 2003.

[129] C. W. Leong, W. Zhuang, Y. Cheng, and L. Wang. Call Admission Control for Integrated On/Off Voice and Best-Effort Data Services in Mobile Celluar Communications. *IEEE Transactions on Communications*, 52(5):778–790, May 2004.

[130] B. Li, L. Li, B. Li, K. M. Sivalingam, and X. Cao. Call Admission Control for Voice/Data Integrated Cellular Networks: Performance Analysis and Comparative Study. *IEEE Journal on Selected Areas in Communications*, 22(4):706–718, May 2004.

[131] D. Lin and R. Morris. Dynamics of Random Early Detection. In *ACM SIGCOMM 97*, Sept 1997.

[132] J. Liu and S. Singh. ATCP: TCP for Mobile Ad-Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300–1315, July 2001.

[133] C. Long and X. Guan. Local stability of REM algorithm with time-varying delays. *IEEE Communications Letters*, 2003.

[134] S. H. Low. A Duality Model of TCP and Queue Management Algorithms. *IEEE/ACM Transactions on Networking*, 11(4):525–536, Aug 2003.

[135] S. H. Low and D. E. Lapsley. Optimization Flow Control I: Basic Algorithm and Convergence. *IEEE Transactions on Networking*, 7(6):861–875, Dec 1999.

[136] S. H. Low and D. E. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–875, Dec. 1999.

[137] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle. Dynamics of TCP/RED and a Scalable Control. In *Proceedings of IEEE Infocom 2002*, June 2002.

[138] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle. Linear Stability of TCP/RED and a Scalable Control. *Computer Networks Journal*, 43(5):633–647, Dec 2003.

[139] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle. Linear stability of TCP/RED and a scalable control. *Computer Networks*, 43(5):633–647, Dec. 2003.

[140] S. H. Low, L. Peterson, and L. Wang. Understanding Vegas: a Dulity Model. *Journal of ACM*, 49(2):207–235, Mar 2002.

[141] S. Lu, V. Bharghavan, and R. Srikant. Fair Scheduling in Wireless Packet Networks. In *ACM Sigcomm 97*, Sept 1997.

[142] R. Mahajan and S. Floyd. RED with Preferential Dropping (RED-PD). In *ICNP 2001*, April 2001.

[143] J. Martin, A. Nilsson, and I. Rhee. Delay-Based Congestion Avoidance for TCP. *IEEE/ACM Trans. on Networking*, 11(3):335–369, June 2003.

[144] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wi reless Links. In *ACM Mobicom*, Sept. 2001.

[145] M. Mathis, J. Semke, and J. Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. In *Proceedings of ACM SIGCOMM*, July 1997.

[146] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In *Technical Report*, June 1999.

[147] M. May, T. Bonald, and F. D. Smith. Analytic Evaluation of RED Performance. In *IEEE Infocom 2000*, March 2000.

[148] V. Misra, W. Gong, and D. Towsley. A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, September 2000.

[149] M. Mitzenmacher and R. Rajaraman. Towards More Complete Models of TCP Latency and Throughput. *Journal of Supercomputing*, 20(2):137–160, Sept 2001.

[150] Kitae Nahm, Ahmed Helmy, and C. C. J. Kuo. TCP over Multihop 802.11 Networks: Issues and Performance Enhancement. In *ACM MobiHoc'05*, May 2005.

[151] B. D. Noble and M. Satyanarayanan. Experience with Adapative Mobile Applications in Odyssey. *Mobile Networks and Applications*, 4(4), 1999.

[152] T. J. Ott, J. H. B. Kemperman, and M. Mathis. The Stationary Behavior of Ideal TCP Congestion Avoidance. *http://www.icir.org/padhye/tcp-model.html*, Aug 1996.

[153] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proceedings of IEEE INFOCOM 1999*, July 1999.

[154] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE Transactions on Networking*, 8(2):133–145, April 2000.

[155] F. Paganini. A Global Stability Result in Network Flow Control. *Systems & Control Letters*, 46(3):153–163, 2002.

[156] F. Paganini, J. Doyle, and S. H. Low. Scalable laws for stable network congestion control. In *IEEE Conference on Decision and Control 40th*, volume 1, pages 185–190, 2001.

[157] F. Paganini, J. C. Doyle, and S. H. Low. Scalable Laws for Stable Network Congestion Control. In *IEEE Conference on Decision and Control (CDC)*, Orlando, FL, Dec. 2001.

[158] A. Papachristodoulou. Global Stability Analysis of a TCP/AQM Protocol for Arbitrary Networks with Delay. In *Proceedings of the IEEE Conference on Decision and Control (CDC) 04*, 2004.

[159] R. Pletka, M. Waldvogel, and S. Mannal. PURPLE: Predictive Active Queue Management Utilizing Congestion Information. In *submit to publication*, 2003.

[160] P. Ranjan, E. H. Abed, and R. J. La. Nonlinear Instability in TCP-RED. In *IEEE Infocom 2003*, Mar 2002.

[161] P. Ranjan, E. H. Abed, and R. J. La. Nonlinear Instabilities in TCP-RED. *IEEE/ACM Transactions on Networking*, 12(6):1079–1092, Dec. 2004.

[162] P. Ranjan, R. J. La, and E. H. Abed. Global Stability Conditions for Rate Control of Discretized Model with Communication Delays. In *Proc. of IEEE Globecom, 2004*, Nov 2004.

[163] P. Ranjan, R. J. La, and E. H. Abed. Global Stability Conditions for Rate Control with Arbitrary Communication Delays. In *Proc. of IEEE CDC, 2004*, Dec 2004.

[164] P. Ranjan, R. J. La, and E. H. Abed. Global stability in the presence of distributed communication delays. In *Proc. of IEEE ACC, 2005*, June 2005.

[165] K. Ratham and I. Matta. WTCP: An Efficient Transmission Control Protocol for Networks with Wireless Links. In *Proc. Third IEEE Symposium on Computers and Communications (ISCC '98), Athens, Greece*, 1998.

[166] S. Ravot. TCP Transfers Over High Latency Bandwidth Networks Grid DT. In *The First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003)*, Feb 2003.

[167] I. Rhee and L. Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. In *Proceedings of Third International Workshop on Protocols for Fast Long-Distance Networks*, Feb 2005.

[168] L. Rizzo. RED and gentle RED (GRED) are supported in the IPFW2 code in FreeBSD. In *http://www.icir.org/floyd/red.html*, 2002.

[169] V. Rosolen, O. Bonaventure, and G. Leduc. Impact of cell discard strategies on TCP/IP in ATM UBR networks. In *Proc. of the 6th Workshop on Performance Modelling and Evaluation of ATM Networks (IFIP ATM 98) Ilkley*, July 1998.

[170] V. Rosolen, O. Bonaventure, and G. Leduc. A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. *ACM Computer Communication Review*, July 1999.

[171] S. M. Ross. Stochastic processes: 2nd Edition. *New York : Wiley*, 1996.

[172] S. M. Ross. Introduction to Probability Models, Eighth Edition. *Academic Press, London*, 2003.

[173] R. Schassberger. Insensitivity of Steady-State Distributions of Generallized Semi-Markov Processes with Speeds. *Advances in Applied Probability*, 10, 836-851 1978.

[174] R. Schassberger. The Insensititivity of Stationary Probability in Networks of Queues. *Advances in Applied Probability*, 10:906–912, 1978.

[175] M. Scott. Distant Early Warning (UCLAdew): Distant Early Warning is an implementation of RED for Solaris 2.5 and Solaris 2.5. In *http://www.icir.org/floyd/red.html*, 1997.

[176] J. Semke, J. Madhavi, and M. Mathis. Automatic TCP Buffer Tuning. In *Proceedings of ACM SIGCOMM*, Aug 1998.

[177] R. F. Serfozo. Introduction to Stochastic Networks. *Springer*, 1999.

[178] S. Shakkottai and R. Srikant. Mean FDE Models for Internet Congestion Control under a Many-Flows Regime. *IEEE Transactions on Information Theory*, 50(6):1050–1072, June 2004.

[179] H. Sivakumar, S. Bailey, and R. L. Grossman. PSocket: The Case for Application-Level Network Striping for Data Intensive Applications Using High Speed Wide Area Networks. In *Proceedings of SuperComputing: High-Performance Networking and Computing*, Nov 2000.

[180] J. Sun, M. Zukerman, K. Ko, G. Chen, and S. Chan. Effect of Large Buffers on TCP Queueing Behavior. In *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.

[181] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar. ATP: A Reliable Transport Protocol for Ad-hoc Network. In *ACM Mobihoc'03*, 2003.

[182] B. Suter, T. V. Lakshman, D. Stiliadis, and A. Choudhury. Efficient Active Queue Management for Internet Routers. In *Interop 98*, 1998.

[183] P. Tinnakornsrisuphap and R. J. La. Characterization of Queue Fluctuations in Probabilistic AQM Mechanisms. In *Proceedings of ACM SIGMETRICS*, New York, NY, June 2004.

[184] P. Tinnakornsrisuphap and R. J. La. Characterization of Queue Fluctuations in Probabilistic AQM Mechanisms. In *ACM SIGMETRICS/Performance 2004*, June 2004.

[185] P. Tinnakornsrisuphap and R. J. La. Asymptotic Behavior of Heterogeneous TCP Flows and RED Gateway . *accepted by IEEE/ACM Transactions on Networking*, 2005.

[186] P. Tinnakornsrisuphap and A. M. Makowski. Limit Behavior of ECN/RED Gateways Under a Large Number of TCP Flows. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, April 2003.

[187] P. Tinnakornsrisuphap and A. M. Makowski. Limit Behavior of ECN/RED Gateways Under a Large Number of TCP Flows. In *Proceedings of IEEE INFOCOM 2003*, April 2003.

[188] P. Tinnakornsrisuphap and A. M. Makowski. On the Behavior of ECN/RED Gateways Under a Large Number of TCP Flows: Limit Theorems. *submitted to QUESTA*, 2005.

[189] C. Villamizar and C. Song. High Performance TCP in Ansnet. *ACM Computer Communication Review*, 24(5):45–60, 1994.

[190] G. Vinnicombe. On the stability of networks operating TCP-like congestion control. Technical report, IFAC 2004.

[191] W. Wang, X. Wang, and A. Nilsson. A New Admission Control Scheme under Energy and QoS Constraints for Wireless Networks. In *Proceedings of IEEE Infocom'05, Miami, FL*, March 2005.

[192] W. Wang, X. Wang, and A. Nilsson. Energy-Efficient Bandwidth Allocation in Wireless Networks: Algorithms, Analysis, and Simulation. In *IEEE Transactions on Wireless Communications*, April 2005.

[193] Z. Wang and F. Paganini. Global Stability with time-delay in Network Congestion Control. In *Proceedings of IEEE Conference on Decision and Control*, Dec 2002.

[194] J. T. Wen and M. Arcak. A Unifying Passivity Framework for Network Flow Control. In *Proceedings of IEEE Infocom*, April 2003.

[195] B. Wydrowski and M. Zukerman. GREEN: An Active Queue Management Algorithm for a Self Managed Internet. In *Proceedings of IEEE ICC 2002*, volume 4, pages 2368–2372, June 2002.

[196] K. Xu and N. Ansari. Stability and Fairness of Rate Estimation-Based AIAD Congestion Control in TCP. *IEEE Communications Letters*, 9(5):378–380, April 2005.

[197] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. In *Proceedings of IEEE Infocom 2004*, March 2004.

[198] Q. Yin and S. H. Low. Convergence of REM flow control at a single link. *IEEE Communications Letters*, 5(3):119–121, March 2001.

[199] Q. Yin and S. H. Low. On stability of REM algorithm with uniform delay. In *IEEE Globecom 2002*, Nov 2002.

[200] L. Ying, G. Dullerud, and R. Srikant. Global Stability of Internet Congestion Controllers with Heterogenous Delays. In *Proceedings of Americal Control Conference*, June 2004.

[201] D. Zhang, X. Shen, and J. W. Mark. Efficient Call Admission Control for Heterogeneous Services in Wireless Mobile ATM Networks. *IEEE Communications Magazine*, 38(10):72–78, Oct 2000.

[202] D. Zhang, X. Shen, and J. W. Mark. Radio Resource Management for Cellular CDMA Systems Supporting Heterogeneous Services. *IEEE Transactions on Mobile Computing*, 11(2):32–39, April-June 2003.

[203] Y. Zhang, S. Kang, and D. Loguinov. Delayed Stability and Performance of Distributed Congestion Control. In *Proceedings of ACM SIGCOMM*, Portland, OR, 2004.

[204] Y. Zhang and L. Qiu. Understanding the End-to-End Performance Impact of RED in a Heterogeneous Environment. In *Cornell CS Technical Report 2000-1802*, July 2000.

[205] B. Zheng and M. Atiquzzaman. Low Pass Filter/Over Drop Avoidance (LPF/ODA): An algorithm to improve the performance of RED gateways. In *Technical Report, University of Oklahoma: CS-TR-01-001*, Feb 2001.