MODELING AND CONTROL OF NETWORK TRAFFIC FOR PERFORMANCE AND SECURE COMMUNICATIONS

A Dissertation

by

YONG XIONG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2004

Major Subject: Computer Science

MODELING AND CONTROL OF NETWORK TRAFFIC FOR PERFORMANCE AND SECURE COMMUNICATIONS

A Dissertation

by

YONG XIONG

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Jyh-Charn (Steve) Liu (Chair of Committee)	
Dmitri Loguinov (Member)	Wei Zhao (Member)
Zixiang Xiong (Member)	Valerie E. Taylor (Head of Department)

December 2004

Major Subject: Computer Science

ABSTRACT

Modeling and Control of Network Traffic for Performance and Secure Communications. (December 2004)

Yong Xiong, B.S., Tsinghua University (China);

M.S., Chinese Academy of Space Technology (China)

Chair of Advisory Committee: Dr. Jyh-Charn (Steve) Liu

The objective of this research is to develop innovative techniques for modeling and control of network congestion. Most existing network controls have discontinuous actions, but such discontinuity in control actions is commonly omitted in analytical models, and instead continuous models were widely adopted in the literature. This approximation works well under certain conditions, but it does cause significant discrepancy in creating robust, responsive control solutions for congestion management. In this dissertation, I investigated three major topics. I proposed a generic discontinuous congestion control model and its design methodology to guarantee asymptotic stability and eliminate traffic oscillation, based on the *sliding mode control* (SMC) theory. My scheme shows that discontinuity plays a crucial role in optimization of the I-D based congestion control algorithms. When properly modeled, the simple I-D control laws can be made highly robust to parameter and model uncertainties. I discussed applicability of this model to some existing flow or congestion control schemes, e.g. XON/XOFF, rate and window based AIMD, RED, etc.

It can also be effectively applied to design of detection and defense of *distributed*

denial of service (DDoS) attacks. DDoS management can be considered a special case of the flow control problem. Based on my generic discontinuous congestion control model, I developed a backward-propagation feedback control strategy for DDoS detection and defense. It not only prevents DDoS attacks but also provides smooth traffic and bounded queue size.

Another application of the congestion control algorithms is design of private group communication networks. I proposed a new technique for protection of group communications by concealment of sender-recipient pairs. The basic approach is to fragment and disperse encrypted messages into packets to be transported along different paths, so that the adversary cannot efficiently determine the source/recipient of a message without correct ordering of all packets. Packet flows among nodes are made balanced, to eliminate traffic patterns related to group activities. I proposed a sliding window-based flow control scheme to control transmission of payload and dummy packets. My algorithms allow flexible tradeoff between traffic concealment and performance requirement.

To *My Wife and Parents*

ACKNOWLEDGMENTS

I would like to thank Dr. Jyh-Charn Liu for his guidance, support and patience through my graduate study. He has been the constant source of inspiration and help. This dissertation would not be finished without his support and encouragement.

I would like also to express my appreciation to the members of my committee, Dr. Dmitri Loguinov, Dr. Wei Zhao and Dr. Zixiang Xiong. They gave me very helpful suggestions on improving the quality of this thesis. I am grateful to them for their guidance and support. They spent time to discuss with me about my research although they were very busy.

I must also appreciate my beautiful wife, Yan Sun, and my parents for their dedication and patience. Without their love, understanding and support, I would not have finished my Ph.D. study.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

The broad acceptance of the IP protocol is driving the Internet to a new level of performance expectations for its users. A key issue related to the network performance is congestion control. Adaptability, fairness, utilization, packet loss and smoothness of traffic flows are common design factors. How to balance these conflicting factors is an extremely challenging undertaking. Coarse approximation of subtle system attributes in an ad hoc design process may work on average cases, but they cannot guarantee real-time performance, and often lead to significant traffic oscillation while real-time streaming applications require smooth transmission rate. One might argue that oscillation helps rapid response to resource changes. We will show that, a system can achieve equally good, if not better, responsiveness and high network utilization, without inducing oscillation.

Oscillation means boundary stability, but achieving traffic smoothness requires asymptotical stability of short-term behavior instead of long-term average behavior. One might argue that linear or other approximation schemes are necessary to contain the computing complexity, and they would be sufficient to deliver quality service. But we will show that this subtle issue is critical to asymptotic stability of the system, and one can make significant performance improvement over existing schemes by proper incorporation of the discontinuity in control actions into the control and modeling process, at lower complexity.

The objective of this research is to develop innovative techniques for modeling and control of network congestion. Most existing network controls have discontinuous

---

The journal model is *IEEE Transactions on Networking*.

actions, but such discontinuity in control actions is commonly omitted in analytical models, and instead continuous models were widely adopted in the literature. This approximation works well under certain conditions, but it does cause significant discrepancy in creating robust, responsive control solutions for congestion management. In this dissertation, I investigated three major topics. I proposed a generic discontinuous congestion control model and its design methodology based on the *sliding mode control* (SMC) theory. I discussed applicability of this model to some existing flow or congestion control schemes, e.g. XON/XOFF, rate and window based AIMD, RED, etc. It can also be effectively applied to design of detection and defense of *distributed denial of service* (DDoS) attacks. The fundamental understanding of control actions, queuing dynamics, and delay leads to a traffic concealment algorithm that can be used to create pattern-free multicast overlay networks.

Discontinuous congestion control rule, e.g. *increase and decrease* (I-D) congestion control, is widely adopted in network traffic management because of its cost-effectiveness. In the second chapter of this dissertation, I proposed a modeling and design methodology for discontinuous congestion controllers to guarantee asymptotic stability and eliminate traffic oscillation, based on the SMC theory. A highly appealing attribute of SMC is that its stability analysis does not require a precise network dynamics model. My scheme shows that discontinuity plays a crucial role in optimization of the I-D based congestion control algorithms. When properly modeled, the simple I-D control laws can be made highly robust to parameter and model uncertainties.

DDoS management can be considered a special case of the flow control problem. Based on the network traffic models derived in chapter II, I developed a backward-propagation feedback control strategy for DDoS defense. When a host finds itself becoming a hot spot, it informs neighboring nodes and routers to reduce influx of

packets. Reduction of packet influx is propagated backward to the sources. If a source is normal, it will reduce its sending rate when backpressure is propagated to it. If a source ignores traffic backpressure and keeps infusing packets, it will be identified as an attacker and its packets will be dropped. This backward-propagation feedback control strategy adopts a simple hop-by-hop on-off control scheme. It not only prevents DDoS attacks but also provides smooth traffic and bounded queue size.

Another application of the congestion control algorithms is design of private group communication networks, which are critical to some applications, e.g. online bank transactions, battle field communication, etc.. In addition to cryptographic protection of information contents, concealment of the network traffic patterns (volumes, peak times, etc.) is also important to prevent unveiling the interactions between group members. Otherwise, adversary might use passive analysis of the network traffic to match certain traffic or patterns with significant application events without explicitly cracking encrypted messages. Workload padding and packet routing are basic mechanisms to counter passive traffic analysis attacks, but workload padding can be very expensive in terms of bandwidth consumption. The challenge here is how to make an optimal tradeoff between traffic pattern anonymity and network performance. I proposed a new technique for protection of group communications by concealment of sender-recipient pairs. The basic approach is to fragment and disperse encrypted messages into multicast packets to be transported along different paths, so that the adversary cannot efficiently determine the source/recipient of a message without correct ordering of all packets. Packet flows among nodes are made balanced, to eliminate traffic patterns related to group activities. I proposed a sliding window-based flow control scheme to control transmission of payload and dummy packets. Our algorithms allow flexible tradeoff between the degree of traffic uniformity, and that of the performance costs. This is particularly important for an open network environment,

where the users may not have full control of the network resources.

Details on the three major research themes, their preliminary results, and their contributions are presented in Chapter II, III and IV, respectively.

CHAPTER II

MODEL AND DESIGN OF DISCONTINUOUS NETWORK CONGESTION CONTROL SYSTEMS

Increase and *decrease* (I-D) congestion control rule is widely adopted in network traffic management because of its cost-effectiveness. In this chapter we propose a modeling and design methodology for I-D congestion controllers to guarantee asymptotic stability and eliminate traffic oscillation, based on the *sliding mode control* (SMC) theory. Our scheme addresses the discontinuous operations of I-D controller that has been largely disregarded in existing literature, and shows that discontinuity plays a crucial role in optimization of the I-D based congestion control algorithms.

We show that the design of I-D congestion control systems must consider its *relative degree* and *zero dynamics* to guarantee their asymptotic stability. Analytical and experimental results show that the relative degree of the rate-based, AIMD flow-control algorithms is two, while that of the window-based schemes is one. That is, for optimal control performance (rapid convergence and minimal oscillation), rate-based AIMD algorithms should use both the queue length error and its first order time derivative to construct the *switching function* of an I-D scheme. On the other hand, for window-based AIMD algorithms one should only use the queue length error in the switching function.

Based on our analysis, we develop a simple yet accurate queuing model for a heavily loaded window-based flow control system. Our model matches both the self-clocking properties of window protocols and their relative degree, i.e., one. Otherwise, the relative degree of a classic analytical model for window-based AIMD is found to be two, inconsistent with that of ns-2 simulation, nor does it reflect the fact that the total input rate is bound under the additive increase rule.

A. Introduction

Sound bandwidth management is essential to sustain the continual growth of Internet. Congestion/flow control schemes used in different transport protocols need to guarantee performance for the broad range of end users, and maximize the network bandwidth utilization. We use "congestion control" and "flow control" interchangeably in this chapter Bandwidth management schemes, which may appear to have good performance results in small scale simulation, need to be closely examined before they can be deployed. Otherwise, overlooking certain subtle system issues can lead to substantial performance loss due to persistent or intermittent oscillation. Wild oscillation results in packet loss, low bandwidth utilization and large delay jitter.

Precise modeling of the network dynamics is critical to the design process in optimizing the tradeoff between responsiveness, fairness and the degree of oscillation. In this chapter, we propose a novel modeling and design approach for increase and decrease (I-D) congestion control based on sliding mode control (SMC) theory [2][3]. Our method sheds new light on understanding of the effect of control discontinuity on stability and performance of I-D schemes such as Additive-Increase-Multiplicative-Decrease (AIMD) [4]. By control discontinuity it means that the reaction to changes of network states for bandwidth adjustment is a discontinuous function. The methodology is general enough to serve as a theoretical basis to the design of a broad range of discontinuous congestion control schemes. Unlike most conventional approaches that omitted the discontinuity of control actions, our analysis takes discontinuity into account and it proves affirmatively the asymptotic stability of different congestion control schemes.

I-D flow control schemes are widely implemented in different network protocols, e.g. XON/XOFF MAC layer flow control scheme, AIMD, etc. The simplest I-D

control rule uses a single binary bit sent to the I-D controller to indicate occurrence of congestion, so that only the sign of the output error is fed back to the I-D controller. This binary feedback can be implicit, e.g., the 3 duplicate ACK packets in TCP, or explicit, e.g., the explicit congestion notification (ECN) bit [5]. One or more [6] ECN bit is commonly used in active queue management (AQM) schemes to inform a sender of congestion conditions.

A congestion control scheme of this nature can be viewed as a binary decision control (BDC) system. That is, if an error function, which is also called the switching function, has a negative (positive) reading then a positive (negative) control action is engaged. Three critical issues related to design of BDC systems are (i) timing in setting of the "congestion" bit, (ii) switching function construct, and (iii) the amount of adjustment in control, i.e., the increase or decrease of the transmission rate or congestion window sizes. Some AQM schemes investigate the timing in determining when to set the congestion bit(s) of flows on the routers. For example, the "binary feedback" scheme proposed by Ramakrishnan and Jain (henceforth called RJBF) [7], RED [8][9], BLUE [10], the proportional-integral (PI) controller [11], proportional-differential (PD) controller [12][13], AVQ [14], R-SMVS (a sliding mode variable structure control scheme proposed by Ren et. al)[15] and variations of RED [16][17][18][19][20]. They are all designed to work with the additive-increase-multiplicative-decrease (AIMD) scheme [21] of TCP Tahoe/Reno. How to provide smooth transport for streaming applications, yet be "TCP-friendly" remains an open problem [22][23].

Construction of a switching function is the focus of congestion avoidance schemes, such as AIMD and binomial control [24]. It is widely believed that one could reduce the degree of oscillation by adjustment of the increase/decrease amounts. For example, in [25], Zhang and Shin proposed an α-control scheme with the goal of control-

ling the maximum queue length to a target range, by adjusting the increase rate of AIMD. In [26], Lee et. al proposed AIMD/H (AIMD with history), which updates the decrease ratio of AIMD according to history, to smooth rate/window variations. However, this is true only to a limited extent because oscillation of I-D control systems can be caused by delay, measurement error or incorrect configuration of relative degree in switching function. An incorrectly configured I-D control system can become unstable or even divergent regardless of the adjustment amounts. We demonstrate this case by a simple example in subsection 5 of section C of this chapter.

AQM and congestion avoidance, being two closely related issues, can be modeled into a binary decision control (BDC) system, unless when the AQM schemes are not designed to work with AIMD, e.g., REM [27], GKVQ (Gibbens-Kelly virtual queue) [28][29], and probabilistic price marking [30][31]. These exceptions will not be considered in the rest of discussion. Analysis and optimization of the asymptotic stability and transient behavior of BDC systems are complicated by discontinuity of the control behavior. When the control action switches according to the + and - sign of the switching function, it changes the trajectory of the system state.

Despite the profound importance of discontinuity in a BDC system, this issue is largely ignored in the analysis of AIMD control [8][11][9][14][15][16][17][18][19][20][12][13]. Instead, most existing work adopted a continuous dynamic model, such as that of window-based AIMD scheme [32] in the TCP Reno as

$$\frac{dW(t)}{dt} = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t)}\rho(t-R(t)), \tag{2.1}$$

where ρ is the packet dropping/marking probability, W the congestion window size, and R the roundtrip time (RTT). In this model, the discontinuity of AIMD is approximated by its statistically-averaged behavior. Inherently the control algorithm

designed based on this model acts according to the statistical average of the controlled objective, rather than its real-time condition.

Our modeling approach leads to significant performance gains, guaranteed stability conditions, and quantitative understanding of the tradeoff between oscillation, delay and fairness. Ignoring control discontinuity might still lead to acceptable performance under certain conditions, but one cannot guarantee the asymptotic stability of the system. Without knowing the behavior of the system dynamics, the system is susceptible to recurrent traffic oscillations, and significant packet delays or packet losses in short time windows. The SMC theory used for analysis of the BDC systems suggests that the relative degree of an I-D controller plays a significant role in its stability. While [3] gives a rigorous definition of relative degree, a simple heuristic method to determine the relative degree is that it equals to the number of times to take the derivative of the output, until the control input first appears in the right-hand side of the equation without being encapsulated in another function.

The rest of this chapter is organized as follows. In section B of this chapter, we briefly introduce the idea of sliding mode control. In section C, we describe the general I-D flow control model and discuss applicability of this model to some existing flow or congestion control schemes, e.g. XON/XOFF, rate and window based AIMD, RED, etc. Section D presents our design methodology for general I-D control rules. Section E of this chapter presents our main results on rate-based and window-based congestion control. Section F and G provide a theoretical proof of our results. Section H discusses the transient behavior and delay effects. We make the conclusion in section I.

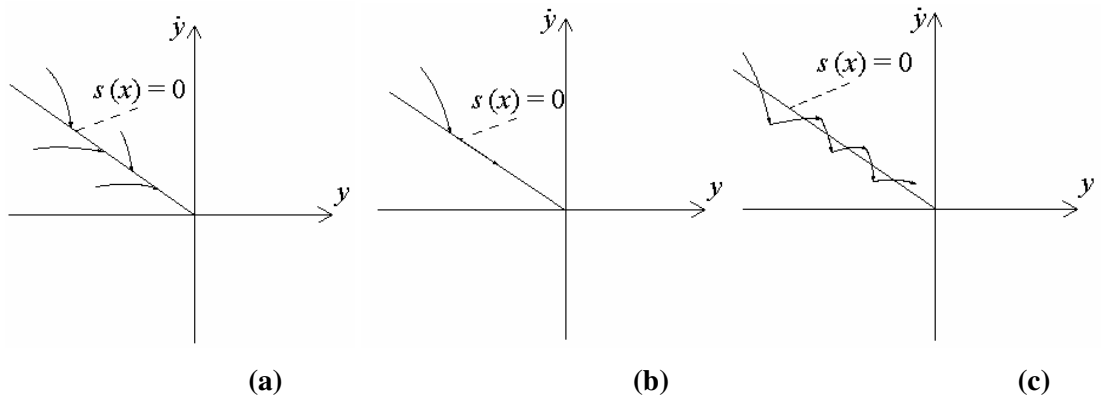

Fig. 1. Two-dimensional case. (a) the attraction effect of the switching surface $s(x) = 0$; (b) a state trajectory with an idea sliding mode; (c) chattering behavior when switching does not take place exactly on the switch surface.

B. Sliding Mode Control

A sliding-mode controller force the system under control to evolve on a predetermined switch manifold, which results in new system dynamics not present in the uncontrolled system. When the system state deviates from the switching surface $s(x) = 0$, where x is the state variable vector of the dynamic system, control is applied on the system to force it back to the switching surface (see Fig. 1). Control is discontinuous on the switching surface. For ideal sliding mode control, the system state trajectory will move toward the switching surface, and then keep evolving on the switching surface (i.e., the term *sliding mode*) to converge to the origin if $s(x) = 0$ is stable. The systems under the sliding-mode control approach are robust and insensitive to parameter uncertainties, making them highly tolerant of model uncertainties and disturbance. The main down side of this approach, however, is the chattering phenomenon, which is caused by the abrupt change of the state trajectory crossing the sliding manifold, even for the ideal, zero-delay cases. Switching cannot exactly take place on the switching surface for real systems, which leads the state trajectory to move back and forth over

the surface $s(x) = 0$.

To construct a sliding mode controller, one must promise the existence and stability of the sliding mode. For a special case where the switching surface is constructed only with output and its derivatives, the dynamics of the sliding mode consists of two parts, the zero dynamics of the system and the switching function $s(y, \dot{y}, \ddot{y}, \cdots) = 0$. Therefore, the sliding mode is stable if and only if the system's zero dynamics and the switching function $s(y, \dot{y}, \ddot{y}, \cdots) = 0$ are both stable. The stability of zero dynamics means the system is minimum-phase.

C. I-D Flow Control Models

In this section, we present the general I-D flow control model, and show its applicability to some typical flow control protocols. We assume a generic congestion control model shown in Fig. 2, where the sender adjusts its sending rate λ by an I-D flow control rule, based on the traffic conditions. The congested node has a first-in-first-out (FIFO) queue and it uses some feedback message(s) to inform senders of the congestion conditions. The primary system parameters include the transmission rate λ, the queue length q, and the link capacity C.

The switching behavior of a general I-D controller in the sender is depicted in Fig. 3, where the controller acts in one of three modes, the increase, decrease and cruise modes. We note that when $S_1 = S_2$ "sliding mode" is the standard term used in SMC theory literature. At the increase (decrease) mode the sender increases (decreases) its sending rate, and at the cruise mode, the sender tries to stay at the same state (with possibly some minor rate adjustments.)

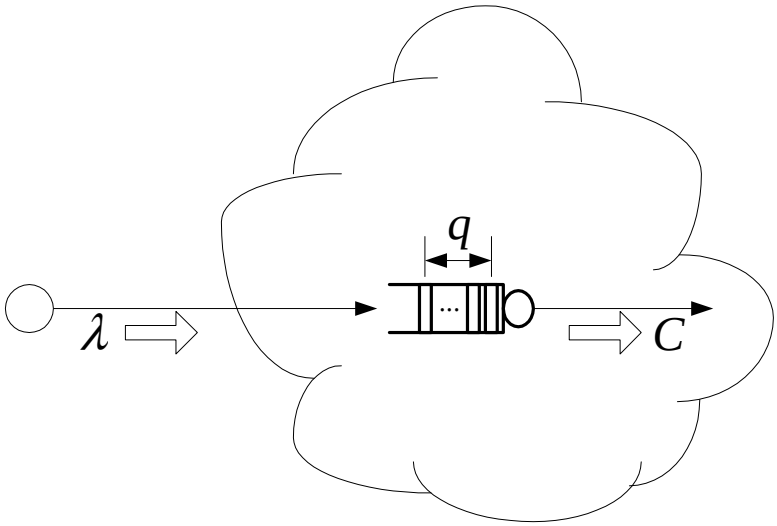

Fig. 2. An FIFO queuing model for congestion control.

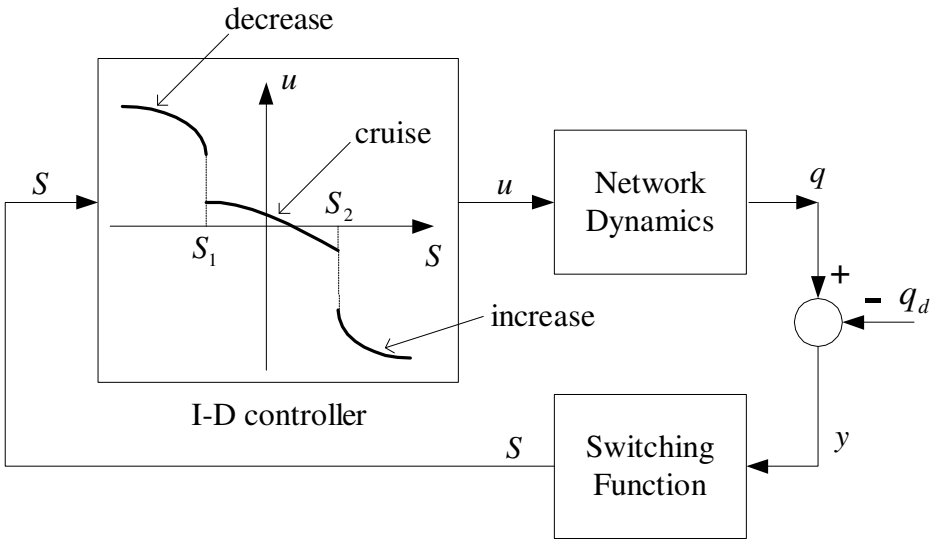

Fig. 3. A general I-D controller networking model.

The I-D control rule for the three modes is defined by

$$u = \begin{cases} u_-, & \text{if } S > S_2, \\ u_+, & \text{if } S < S_1, \\ u_c, & \text{else,} \end{cases} \tag{2.2}$$

where u is the control input, u_-, u_+ and u_c are the *decrease*, *increase* and *cruise* rule, S is the switching function, S_1 and S_2 ($S_1 < S_2$) are the thresholds. The design issue is how to choose i) the switching function S, ii) thresholds S_1 and S_2, and iii) u_-, u_+ and u_c. Knowing that full-state feedback is unlikely, we only consider the following output based switching function (2.3)

$$S = c_0 y + \sum_{i=1}^{r-1} c_i \frac{d^i y}{dt^i}, \tag{2.3}$$

where y is some measurement output, e.g. queue length error. A generic description of the network dynamics is expressed as

$$\dot{x} = f(x, u, t), \tag{2.4}$$

where f is a smooth function, t is the time variable, x stands for the state variable vector of the network dynamics, and u the control input. The output y is a function of the network state variables,

$$y = h(x, t). \tag{2.5}$$

The network state variables x could be the queue length, changing rate of the queue length, etc. Our objective is to drive the network dynamics to a desired equilibrium point in the cruise mode to achieve maximum bandwidth utilization and minimal oscillation. Next, we show the applicability of our model to some widely adopted flow control protocols.

1. XON/XOFF Protocol

IEEE 802.3x standard defines a data-link layer, hop-by-hop backpressure flow control scheme called XON/XOFF. The flow control rule of XON/XOFF can be described as

$$\lambda = \begin{cases} 0, & \text{if } q > q_2, \\ C, & \text{if } q < q_1, \\ \text{hold}, & \text{else}, \end{cases} \tag{2.6}$$

where λ is the sending rate, C is the link capacity, q_1 and q_2 are the XON and XOFF thresholds respectively. A node sends an XOFF packet to its upstream node when its queue length exceeds q_2 and it sends an XON packet when its queue length reduces to q_1. A node stops sending packets when it receives an XOFF packet. It resumes transmission when it receives an XON packet from the same node, or the XOFF message expires. The goal of XON/XOFF scheme is to control the queue length to the range between q_1 and q_2 to avoid queue overflow and underflow. Eq. (2.6) captures the key property of XON/XOFF scheme although it ignores the XOFF expiration effects. Clearly (2.6) is consistent with (2.2).

2. Window-Based AIMD Protocol

In a generic window-based AIMD protocol, e.g. RJBF [7] and the congestion avoidance scheme of TCP Tahoe/Reno [33], senders exponentially decrease their congestion windows (denoted as W) when they receive a marked ACK packet, otherwise they linearly increase their congestion windows. In RJBF, the congested node marks an incoming packet when its queue length is larger than a threshold. The congestion avoidance scheme of TCP Tahoe/Reno can be viewed as a special case in which the queue length threshold is the buffer size of the congested node. A TCP sender halves

its congestion window when it receives three duplicate ACKs, otherwise increases its congestion window by one per round trip time. Duplicated ACKs are caused by packet loss. We ignore packet loss caused by transmission errors because bit error ratio in modern wired network is very low. The cases of wireless network is beyond the range of this thesis.

This scheme has only increase and decrease modes, and it does not have the cruise mode. The control input u is the changing rate of congestion window size, denoted as $\dot{W}$. u_- and u_+ denote the multiplicative decrease and additive increase rules respectively. The output y is the queue length of the congested node. The window-based AIMD control rule can be written as [24]

$$\dot{W}_i = \begin{cases} \frac{\alpha_w}{R_i}, & \text{if } q < q_d, \\ -\frac{W_i}{\beta_w}, & \text{else,} \end{cases} \tag{2.7}$$

where R_i is the roundtrip time of the i^{th} session.

3. Rate-Based AIMD Protocol

Some representative schemes in this category include the Rate Adaptation Protocol (RAP) [34], the Loss-Delay Based Adaptation Algorithm LDA+ [6], and the Loss Tolerant Rate Controller (LTRC) [35]. In RAP, each data packet is acknowledged, and sender uses ACK packets to detect packet loss and round-trip time. When packet loss is detected, the sender halves its sending rate. Otherwise, it periodically increases its sending rate by one packet per round-trip time. LDA+ adaptively adjusts its additive increase and multiplicative decrease rates based on the network condition. LTRC uses loss threshold values to determine whether the sender is allowed to increase, maintain

or decrease its rate. The control rule of this type of protocols can be expressed as

$$\dot{\lambda}_i = \begin{cases} \alpha_r, & \text{if } q < q_d, \\ -\dfrac{\lambda_i}{\beta_r}, & \text{else.} \end{cases} \tag{2.8}$$

4. Random Early Detection (RED) Protocol

The AQM scheme *Random Early Detection* (RED) proposed in [8] is designed to work with TCP or similar protocols. It detects router congestion based on its average queue length, and disseminates congestion information by packet dropping or ECN bit marking. The probabilistic rules for packet dropping (or ECN marking) are as follows:

$$\rho = \begin{cases} 0, & \text{if } \bar{q} < q_1, \\ 1, & \text{if } \bar{q} > q_2, \\ \dfrac{\bar{q} - q_1}{q_2 - q_1}, & \text{else,} \end{cases} \tag{2.9}$$

where ρ is the dropping/marking probability, and $\bar{q}$ is the average queue length. The goal of RED is to drive $\bar{q}$ to a desired equilibrium point, which falls within (q_1, q_2). Obviously, (2.9) is a special case of (2.2).

Most literatures on RED, e.g. [11][9], only analyzed its local stability by linearizing the system around a neighbor of the equilibrium point within (q_1, q_2) with little or no discussion on the issues of the transition between the three states. There is no guarantee that when initially the queue length is less than q_1 or larger than q_2, RED would drive the network dynamics into the cruise mode. This assumption makes the congestion control scheme at a sender susceptible to traffic fluctuation from other sources. Our model can alleviate this shortfall by taking the full dynamic ranges of operations into account.

5. A Simple Example

Referring to the FIFO queue example in Fig. 2, let C be 150 pkt/sec and zero delay, then the system dynamics of this queuing system can be described by

$$\dot{q} = \lambda - 150, q \geq 0. \tag{2.10}$$

We also assume the buffer size is infinite in order to more clearly show if the queuing system convergent. Our goal is to drive q to 50 packets through increasing /decreasing the sending rate λ. Thus, the output-measurement error

$$y = q - 50, \tag{2.11}$$

and the control variable

$$u = \dot{\lambda}. \tag{2.12}$$

Now we study the stability of (2.10) under the following additive-increase-additive-decrease (AIAD) control law with different switching functions:

$$u = \begin{cases} -K, & \text{if } S \geq 0, \\ K, & \text{else,} \end{cases} \tag{2.13}$$

where K is a positive constant. We consider the following three switching functions:

$$S = y, \tag{2.14}$$

$$S = -y + \dot{y}, \tag{2.15}$$

$$S = y + \dot{y}. \tag{2.16}$$

By solving these equations with the 4^{th} order Runge-Kutta numerical integration method, we got the results plotted in Fig. 4, 5 and 6. For (2.14), with initial state

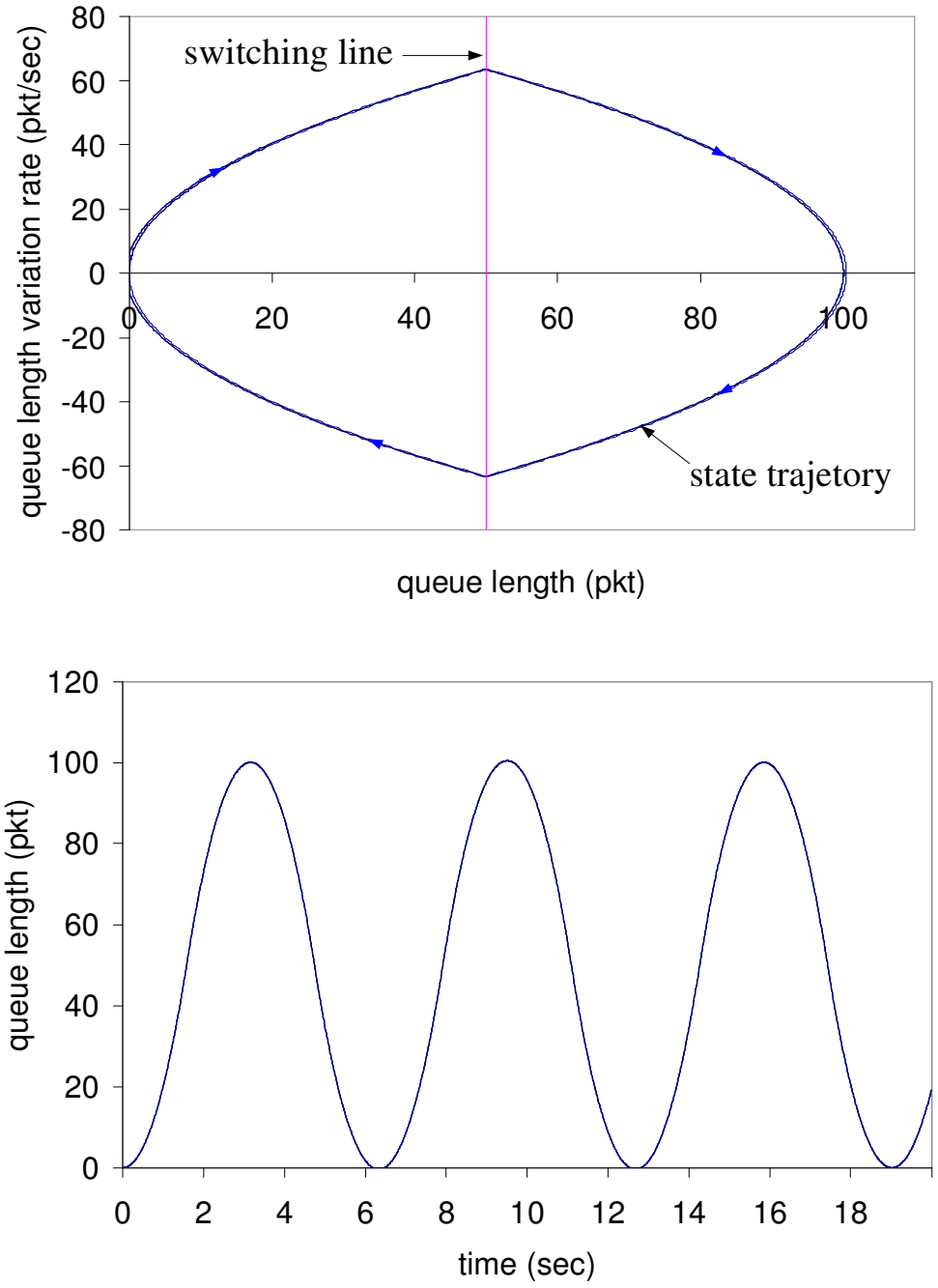

Fig. 4. The state and temporal trajectories with switching function $S = y$ ($K = 40$). Delay is ignored. The switching line is defined by $S = 0$.

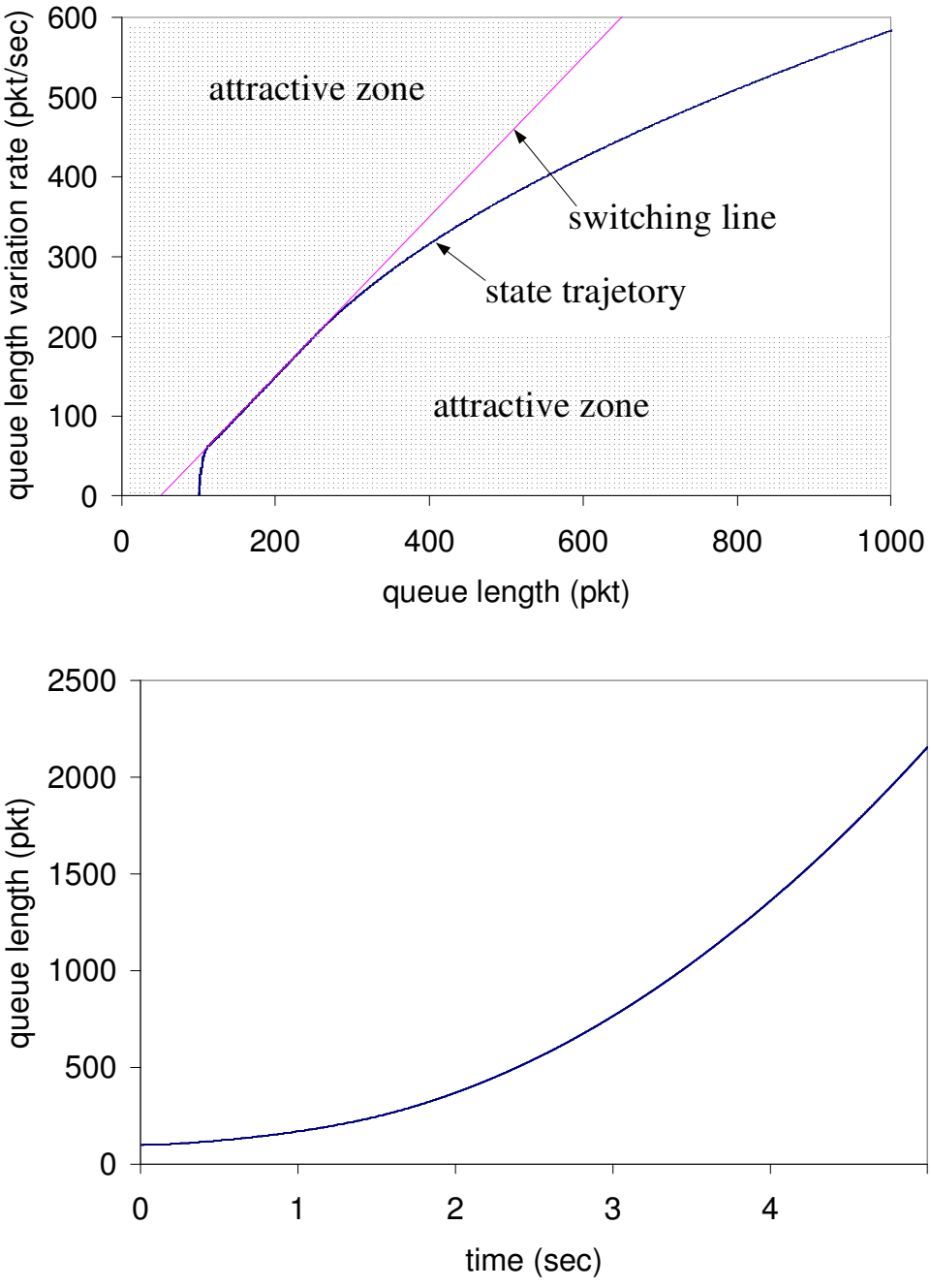

Fig. 5. The state and temporal trajectories with switching function $S = -y + \dot{y}$ ($K = 200$). Delay is ignored. The switching line is defined by $S = 0$.

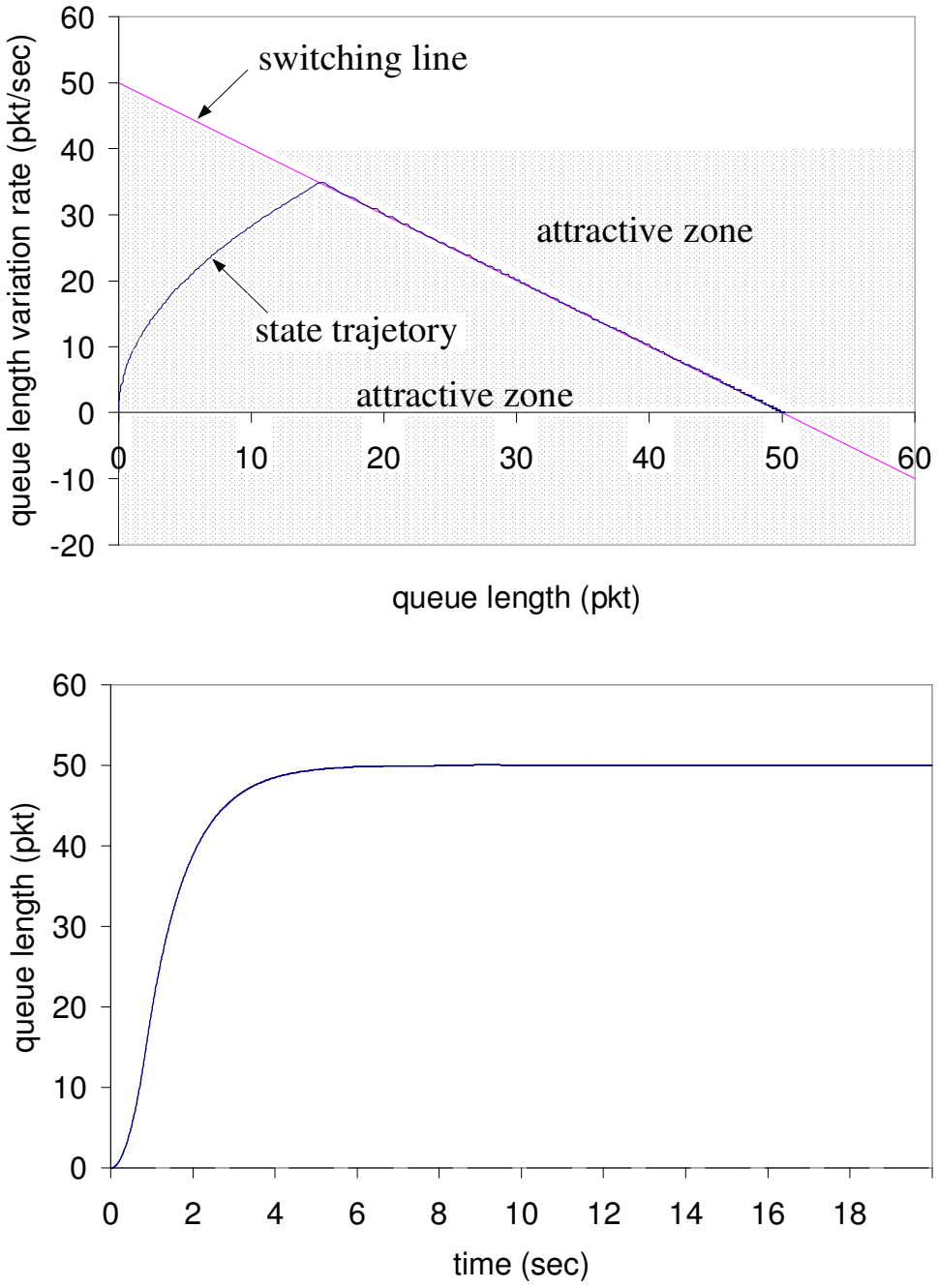

Fig. 6. The state and temporal trajectories with switching function $S = y+\dot{y}$ ($K = 40$). Delay is ignored. The switching line is defined by $S = 0$.

$(q = 0, \dot{q} = 0)$, the close-loop system oscillates, see Fig. 4. For (2.15), with initial state $(q = 100\text{pkt}, \dot{q} = 0\text{pkt/sec})$, the system is divergent, see Fig. 5. For (16), with initial state $(q = 0, \dot{q} = 0)$, it asymptotically converges to an equilibrium point, see Fig. 6. This example clearly shows the importance of choosing the proper switching functions, based on the system dynamics, regardless of the complexity (or simplicity) of the system architecture. We will discuss attractive zones in Fig. 4, 5 and 6 in the next section.

D. General Design Methodology

The main idea of our method is that we first design the increase and decrease rules (u_+ and u_- in (2.2)) to make the system monotonously converge to the cruise mode within a finite time period, and then maintain the system state at the cruise mode. By "monotonously converge" it means that the distance to the cruise mode monotonously decreases until the network state reaches the cruise mode. It makes the system state converge to the cruise mode and maintain on the cruise mode after the system state reaching the cruise mode. This is enough for most flow control applications. The second step is to design the cruise rule (u_c in (2.2)) to make the system asymptotically converge to the desired equilibrium point with continuous system design methods, e.g. root locus, pole placement, LQG regulator design, etc.

Let $\hat{S}$ denote the "distance" to S_1 and S_2 outside the cruise mode.

$$\hat{S} = \begin{cases} S - S_2, & \text{if } S > S_2, \\ S - S_1, & \text{if } S < S_1, \\ 0, & \text{else.} \end{cases} \tag{2.17}$$

Thus the I-D control law can be rewritten as

$$u = \begin{cases} u_-, & \text{if } \hat{S} > 0, \\ u_+, & \text{if } \hat{S} < 0, \\ u_c, & \text{else.} \end{cases} \tag{2.18}$$

Monotonously decreasing of $|\hat{S}|$ implies that $\hat{S}\dot{\hat{S}} < 0$. That is, when $\hat{S} < 0$, $\dot{\hat{S}} > 0$ and when $\hat{S} > 0$, $\dot{\hat{S}} < 0$, where $\dot{\hat{S}}$ is the time derivative of $\hat{S}$. The *attractive zone* of the cruise mode is defined as $\{x : \hat{S}\dot{\hat{S}} < 0\}$. Within the attractive zone, the network states will be attracted into the cruise mode. Furthermore, when

$$\hat{S}\dot{\hat{S}} \leq \eta|\hat{S}|, \text{where } \eta > 0, \tag{2.19}$$

is satisfied, the system will reach the cruise mode within a finite time period $\frac{|\hat{S}_0|}{\eta}$ [3], where $\hat{S}_0$ is the initial value of $\hat{S}$, and the convergence rate to the cruise mode is proportional to η.

Based on this methodology, to guarantee overall system asymptotical stability, i.e., the controlled object (2.4) and (2.5) together with the controller (2.2) and (2.3), we must meet the following conditions [36]:

i. If the relative degree of the input-output system defined by (2.4) and (2.5) with input u and output S is r, one should use to construct the switching function S.

ii. The zero dynamics of the input-output system defined by (2.4) and (2.5) is asymptotically stable, or it is minimum phase.

iii. $c_{r-1}p_{r-1} + c_{r-2}p_{r-2} + ... + c_1 p + c_0$ is a Hurwitz polynomial.

iv. Under the cruise control u_c, (2.4) is asymptotically stable.

The four conditions are sufficient for asymptotical convergence to an equilibrium point. Condition i guarantees that the control input u explicitly appears in the expression of $\dot{S}$ so that we can change the sign of $\dot{S}$ through switching of the control input. Satisfying Conditions i-iii is sufficient for asymptotically converging to a limited cycle with bounded oscillation level, roughly $S_1 < y < S_2$. When $S_1 = S_2$, Conditions i-iii are sufficient for asymptotically converging to an equilibrium point, where $y = S_1$.

For a nonlinear system, its zero dynamics is equivalent to the role of zeros for a linear system. When Condition iii holds, Condition ii is equivalent to the condition that the zero dynamics with output S defined in (2.3) is asymptotically stable, or, it is minimum phase with output S. In fact, the zero dynamics with output S defined in (2.3), which is exactly the dynamics of the sliding mode, is the combination of two elements [2][36]: the zero dynamics with output y and

$$c_0 y + \sum_{i=1}^{r-1} c_i \frac{d^i y}{dt_i} = 0. \tag{2.20}$$

Condition iii guarantees the differential equation (2.20) is asymptotically stable.

Coefficients $c_r, c_{r-1}, \cdots, c_0$ are manually determined to meet Condition iii. $u_-(x,t)$ and $u_+(x,t)$ determine the range of the attractive zone of the cruise mode, so that within the attractive zone, the control rules can bring the state trajectory back to the cruise mode. Their values are proportional to the speed of state trajectory changes. When the system has non-negligible delays, measurement errors, or finite control frequency, control switching cannot take place exactly on the switching line (manifold), which will cause state chattering. For these non-ideal cases, large control values will aggravate the degree of chattering (see Fig. 7). The quantitative tradeoff between these factors is essential to the optimal design of the congestion control algorithms.

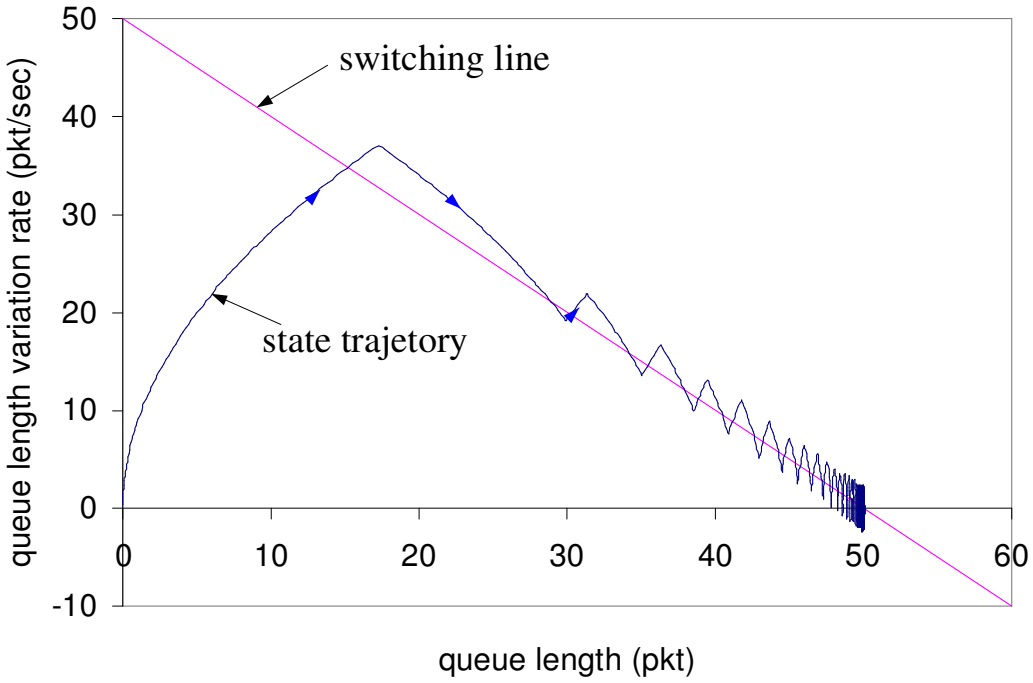

Fig. 7. The state trajectory of the simple FIFO queue example under control law (2.13), where $S = y + \dot{y}$, $K = 40$ and with delay 50 ms.

For the simple FIFO queue control example given in the end of section C of chapter II, on page 14, its relative degree is two. By choosing $S = y$, it violates Condition i so that one cannot make $S = 0$ attractive through switching control input u. Its oscillation level is just dependent on the initial q and λ. $S = -y + \dot{y}$ and $S = y + \dot{y}$ satisfy Condition I so that within the attractive zone both of them can drive the state trajectory to $S = 0$. However, the attractive zone under control law (2.13) is limited because its control is limited. $S = -y + \dot{y} = 0$ is divergent because it violates Condition iii so that the state trajectory will escape from the attractive zone.

E. Main Results

In this section, we study the stability of rate-based and window-based AIMD congestion control protocols. To single out the effects of relative degree on the performance outcomes, we focus on the relationship between relative degree and the asymptotic stability, without considering the transient on-set behavior of a new session.

In the experiments, we tested the effects of relative degree by using two different switching functions,

$$\text{SF1:} \quad S = y, \tag{2.21}$$

and

$$\text{SF2:} \quad S = y + c_1\dot{y}, \tag{2.22}$$

where $y = q - q_d$. q is the actual queue length and q_d is the target queue length. Eqs. (2.21) and (2.22) respectively represent feedback control schemes that have relative degrees of one and two.

In the real implementation of (2.22), we use the following differential equation to approximate $\dot{q}$

$$\dot{q}(t) \approx \frac{\bar{q}_i - \bar{q}_{i-1}}{\Delta t} \tag{2.23}$$

where $t \in [i\Delta t, (i+1)\Delta t)$ and $\bar{q}_i$ is the average queue length within the time interval $[(i-1)\Delta t, i\Delta t)$.

1. Rate-Based AIMD

In rate-based AIMD scheme, the sender keeps sending data at the current rate until it receives one-bit congestion status, which is set by the router, and is transmitted back by the receiver to the sender via the ACK packet. The source linearly increases its sending rate with rate $\alpha(> 0)$ if the congestion bit is 0; otherwise, the sending rate

is decreased exponentially with time constant $\beta(> 0)$. Note that in our simulation, the source does not retransmit lost packets, because this assumption does not have effect on the relative degree properties. This AIMD configuration is consistent with (2.2), and thus can be analyzed by our model.

In our ns-2 simulation, we assume that a single link of capacity 10 Mbps is shared by 15 connections with RTT ranging from 40 ms to 200 ms. These connections started randomly within 0.1 second. The average packet size is 1000 bytes, the buffer size 100 packets, and $q_d = 50$ packets is our control goal. Because we are mainly interested in the recurrent behavior of AIMD, we simply initialized the sending rate of each sources as 650 Kbps and set $\alpha = 10\text{pkts/sec}^2, \beta = 8.3\text{sec}$, without considering the on-set effects. We always set $c_1 = 1$ unless otherwise stated. This configuration satisfies $\alpha\beta = C/N$, where C is the link capacity, so that at equilibrium, the increase rate is equal to the decrease rate [37].

The ns-2 simulation results of using SF1 and SF2 are plotted in Fig. 8. It is clear that the oscillation of the queue length in (2.22) is much smaller than that in (2.21). Consistent results from numerical and ns-2 simulations repeatedly showed that oscillation was reduced drastically when the relative degree was correctly configured. Next, we consider the state trajectories of the two switching functions. For SF1, which does not take into account of the relative degree, there exists a large cycle on the state trajectory, (see Fig. 9) in which the switching manifold is the vertical dotted line passing the coordinate (50,0). On the other hand, for SF2, the state trajectory (see Fig. 10) does chatter around the switching manifold, which is the straight dotted line connecting coordinates (0, 50) and (50, 0). From the design viewpoint, the results suggest that without taking into account the relative degree, it will be very difficult to contain the control behavior of the system to a certain range of the target utility function, let alone any notion of performance guarantee. Chattering and scattering of

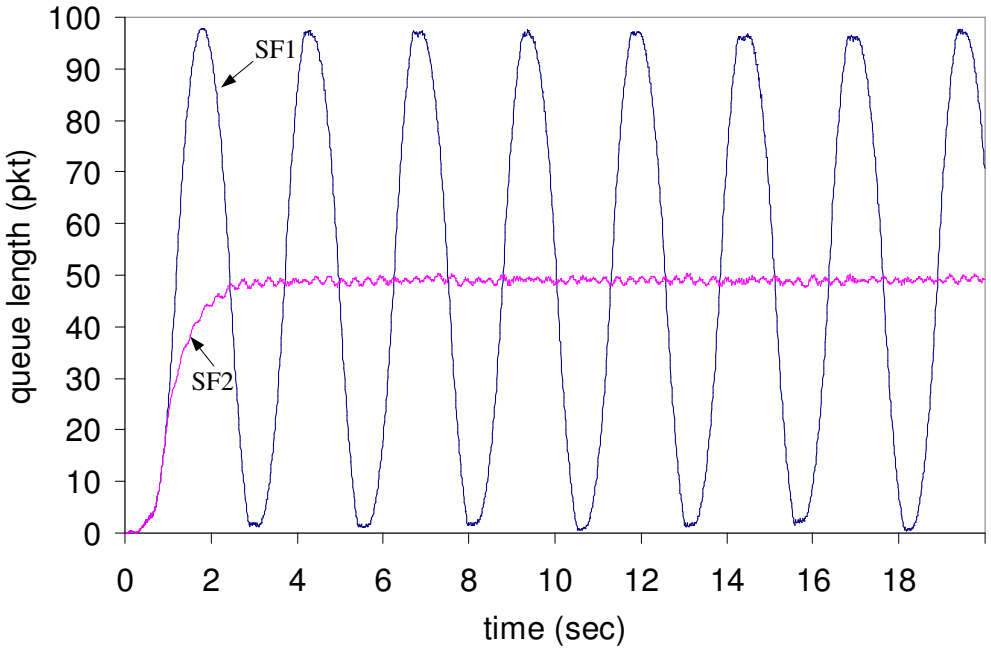

Fig. 8. Instant queue length traces based on SF1 and SF2.

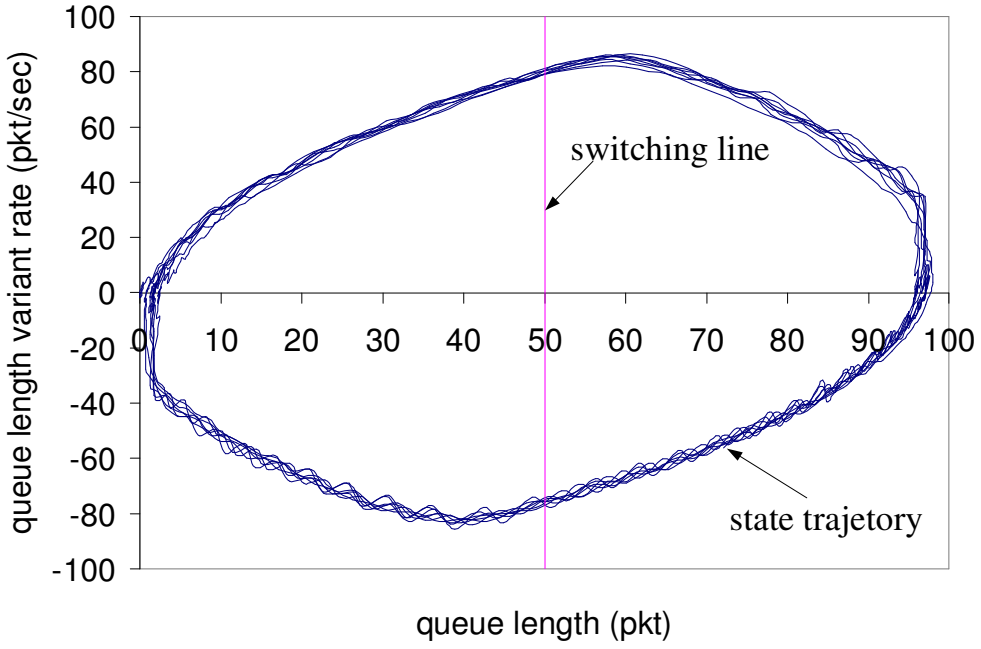

Fig. 9. State trajectory for SF1.

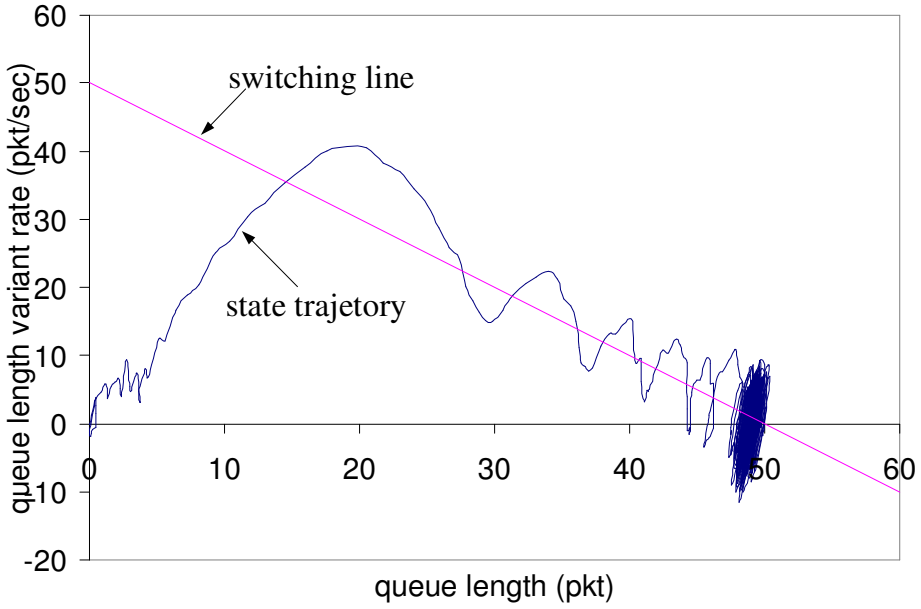

Fig. 10. State trajectory for SF2 with $c_1 = 1$.

state trajectories depicted in the simulations were caused by delay, non-zero control period and measurement error that prevented switching from falling on the switching manifold perfectly. Provided that the relative degree is properly configured, these factors only have limited effects on our analysis outcomes, as it was shown in our numerical and simulation studies.

Next, we compared the performance results of using different α and β values. The RTT is set to be less than 10^{-6} ms to rule out the effects of delays. Our simulation results on using different α and β values in SF1 are plotted in Fig. 11. For SF1, despite the significant phase differences under different configurations, their levels of oscillation were very close to each other. We caution that after the relative degree is properly chosen, one still needs to optimize the α and β values to minimize chattering caused by delay. There is tradeoff between minimizing chattering and response time. For the SF2 example depicted in Fig. 12, one can see that lager α and smaller β result in larger oscillation and faster response (smaller rising time).

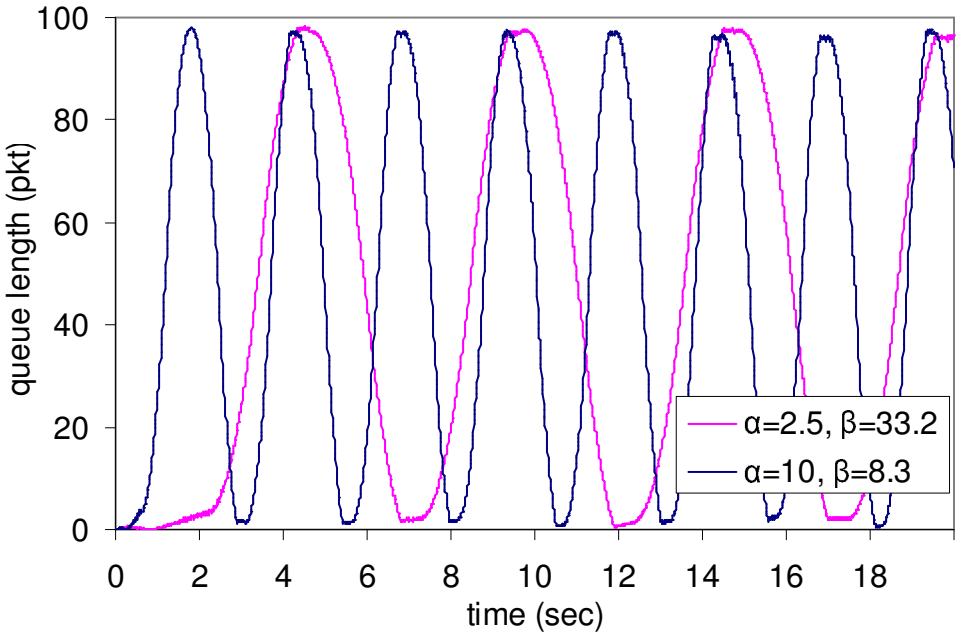

Fig. 11. Traces of queue lengths for SF1 with different α and β values.

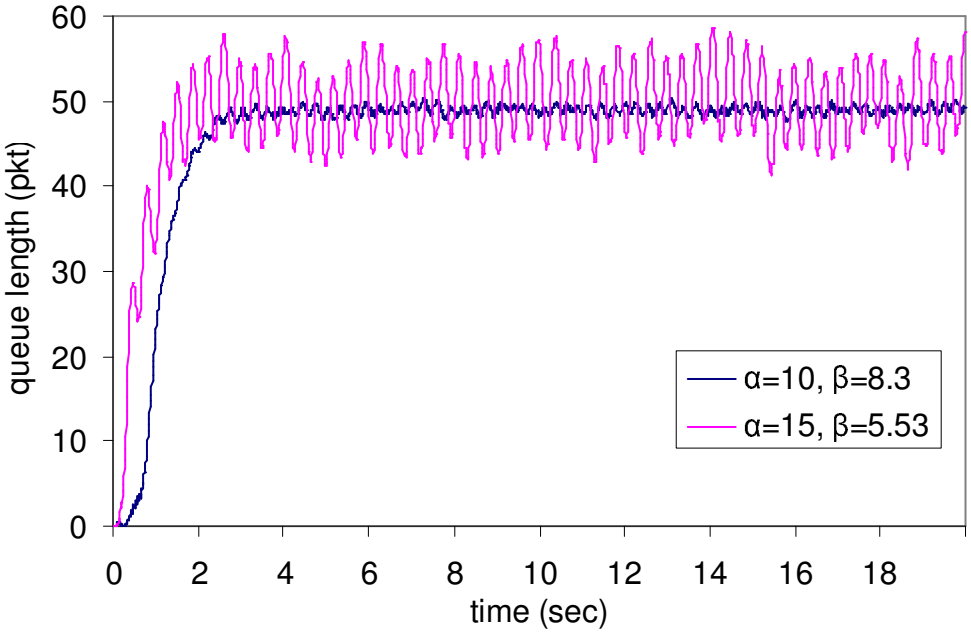

Fig. 12. Traces of queue lengths for SF2 with different α and β values.

2. Window-Based AIMD

The self-clocked behavior [21] of the window-based congestion control systems makes its dynamic characteristics totally different from that of its rate-based counterpart. Through a similar modeling and simulation process of SF1 and SF2 for a window-based system, we concluded that the relative degree of the window-based congestion control system is one when the output is queue length error and control input is the variation rate of congestion window size.

In the experiment, we assume that a single link of capacity 2Mbps is shared by 15 TCP NewReno [33] connections that started randomly within the first 0.1 second, and the delay was set at 60 ms and 66 ms. Other simulation parameters remain unchanged. That is, average packet size is 1000 bytes, buffer size 100 packets, and $q_d = 50$ packets. A TCP NewReno sender decreases its $cwnd$ by $\beta_w cwnd$ packets for each received marked ACK, while increases its $cwnd$ by $\alpha_w/cwnd$ packets for each received unmarked ACK. For reasons similar to the analysis of rate based systems, we directly initialized the congestion window ($cwnd$) size of each source as one packet and disabled the slow start mechanism. The router marks a packet when S defined by (2.21) or (2.22) is positive. We set $\alpha_w = 0.5$, $\beta_w = 0.01$, without considering the on-set phase of a session. As usual, $c_1 = 1$.

Fig. 13 shows that the queue length of the congested link asymptotically converges for both SF1 and SF2. However, SF1 reaches the target queue length, which is 50, but SF2 has a large steady-state error from the target queue length. Given the equilibrium point (50, 0), the state trajectory diagram for SF1 is shown in Fig. 14 where the system converges to the equilibrium point. On the other hand, Fig. 15 clearly shows that the system swings in but stays off the target equilibrium point for SF2. Fig. 14 and Fig. 15 also show that for window-based AIMD, the total input rate

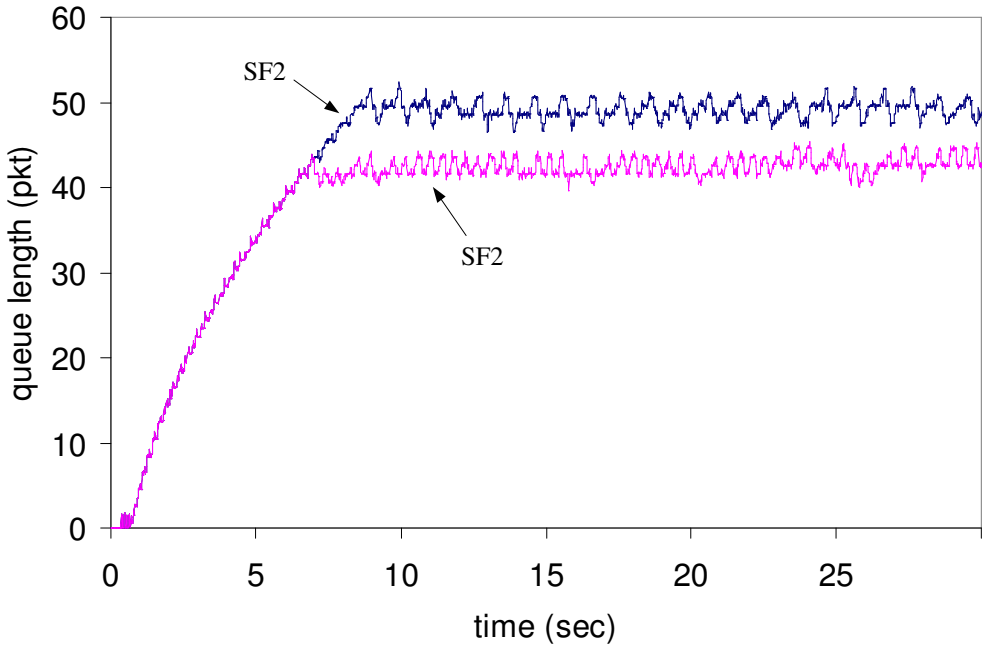

Fig. 13. Queue length traces based on SF1 and SF2 in window-based congestion control.

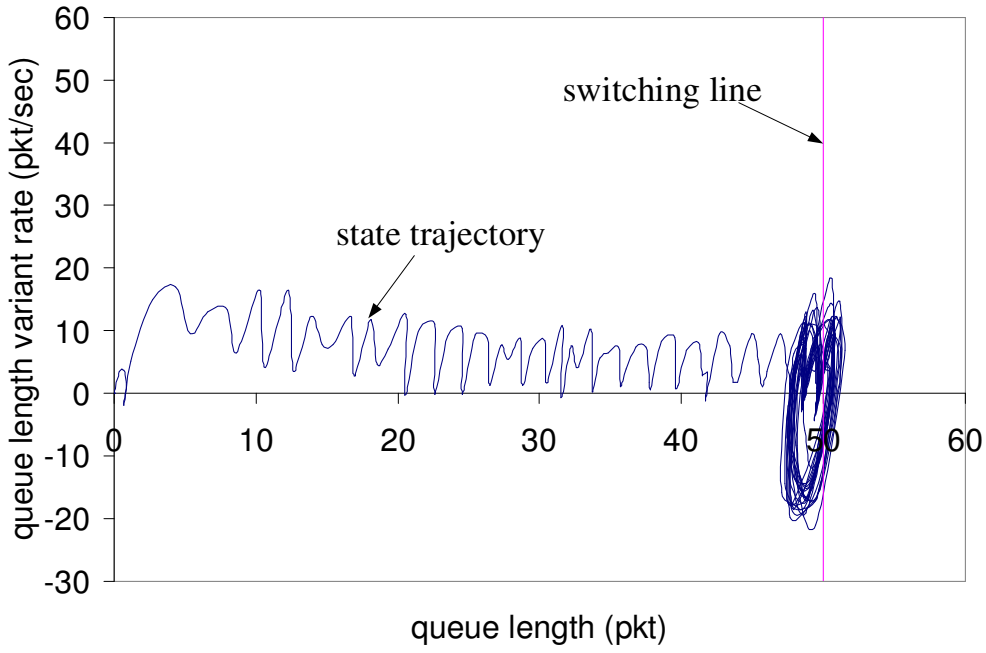

Fig. 14. State trajectory for SF1 (window-based case).

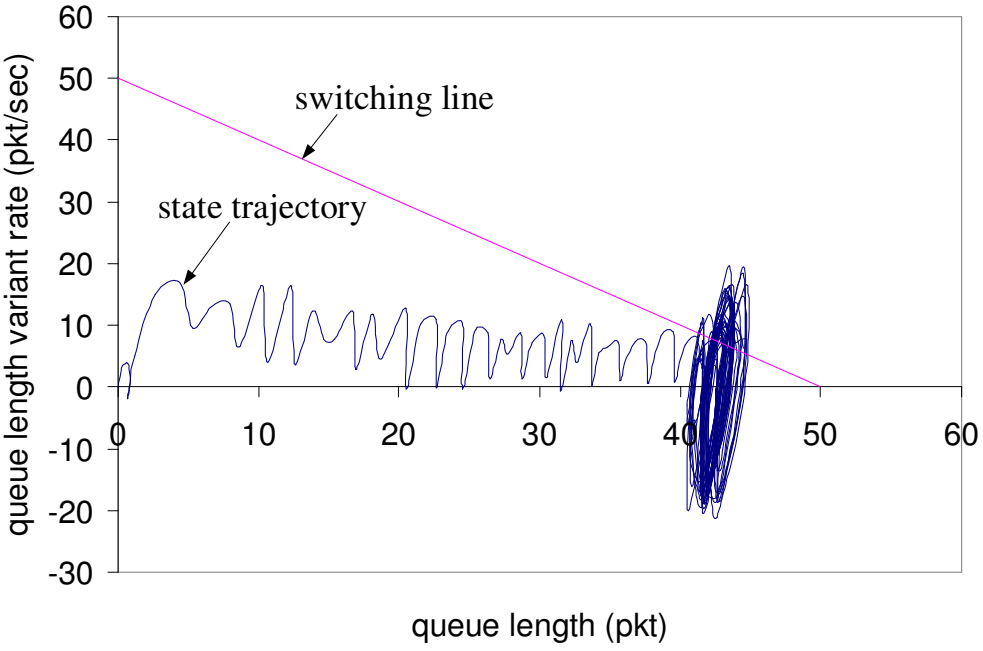

Fig. 15. State trajectory for SF2 with $c_1 = 1$ (window-based case).

(equals to the queue length variant rate plus the link capacity) is bounded, while Fig. 9 and Fig. 10 show that for rate-based systems, there is an overshot-the total input rate keeps increasing until crossover the switching line. Bounded total input rate in window-based AIMD is caused by its self-clocking property. The bounded total input rate cannot be explained by a widely-accepted queuing model for window-based systems. We will further discuss this issue in section G of this chapter.

F. Analysis for Rate-Based Systems

In this section, we discuss asymptotic stability conditions and other related issues for rate-based AIMD systems.

1. Rate-Based Network Model

We assume that the congested link is shared by N sources. Let q denote the queue length of that link, C link capacity, λ_i the sending rate of source i, ζ the total input rate to the link, $\zeta(t) = \sum_{i=1}^{N} \lambda_i(t - \tau_{Fi})$, and τ_{Fi} the (outbound) delay from source i to the link. We get

$$\dot{q}(t) = -CG(q(t), \zeta(t)) + \zeta(t) + \hat{C}(t), \tag{2.24}$$

where $\hat{C}(t)$ represents short-term non-responsive traffic, and $G(\cdot, \cdot)$ the instantaneous link utilization. In a general sense, we assume that $G(\cdot, \cdot)$ is a function of q and ζ so that it can represent bandwidth allocation schemes on the router that make allocation decisions based on current queue length and input rate.

It is obvious that $0 \leq G(q, \zeta) \leq 1$ for $q > 0$, $\zeta > 0$. For mathematical completeness, we further assume that $G(q, \zeta)$ is twice differentiable with respect to q and ζ. In a real system, any increase in ζ should lead to increase in $\dot{q}$. That is, $\frac{\partial \hat{q}}{\partial \zeta} > 0$ should hold. By plugging this condition into (2.24), it is easy to get

$$\frac{G}{\zeta} < \frac{1}{C}. \tag{2.25}$$

Now, our goal is to drive $y = q - q_d$ to zero by adjusting λ_i. To reduce chattering of λ_i, we adjust $\hat{\lambda}_i$ in the same way as the existing AIMD scheme. By introducing new control variables $u_i \equiv \dot{\lambda}_i$, we get

$$\dot{\zeta}(t) = \sum_{i=1}^{N} u_i(t - \tau_{Fi}). \tag{2.26}$$

2. System Properties

The first property that we want to address is the relative degree of rate-based congestion control schemes, in addition to the empirical evidences presented earlier.

Theorem 1 *The relative degree of the input-output system defined by (2.24) and (2.26) is two, when the output is y and control variable is u_i, respectively.*

Proof: It is easy to verify that $\frac{\partial \dot{y}}{\partial u_i} = 0$ and $\frac{\partial \ddot{y}}{\partial u_i} = 1$. This means that control input u_i explicitly appears in $\ddot{y}$ but not in $\dot{y}$. The theorem stands based on the definition of the relative degree. □

According to conditions I and III in Section D of chapter II, we now choose $S = y + c_1\dot{y}$, $c_1 > 0$, as the switching function. Next we consider the minimum phase condition.

Theorem 2 *The zero dynamics of the system defined by (2.24) and (2.26) is asymptotically stable when the output is $S = y + c_1\dot{y}$, $c_1 > 0$.*

Proof: We define a transform

$$\begin{cases} z_1 = S, \\ z_2 = y. \end{cases} \tag{2.27}$$

Its Jacobi matrix

$$\begin{bmatrix} \frac{\partial z_1}{\partial q} & \frac{\partial z_1}{\partial \zeta} \\ \frac{\partial z_2}{\partial q} & \frac{\partial z_2}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} 1 - c_1 C\frac{\partial G}{\partial q} & c_1 - c_1 C\frac{\partial G}{\partial \zeta} \\ 1 & 0 \end{bmatrix} \tag{2.28}$$

is nonsingular because $\frac{\partial G}{\partial \zeta} \neq \frac{1}{C}$ for arbitrary q and ζ. Thus, transform (2.27) is a global *diffeomorphism*. Applying (2.27) to (2.24) and (2.26), we get

$$\dot{S} = \dot{z_1} = (1 - c_1 C\frac{\partial G}{\partial q}\dot{q}) + c_1(1 - C\frac{\partial G}{\partial \zeta})\dot{\zeta} + c_1\dot{\hat{C}}, \tag{2.29}$$

$$\dot{z_2} = \frac{z_1 - z_2}{c_1}. \tag{2.30}$$

Substituting $z_1 = 0$ into (2.30), we conclude that, when the output is $S = y + c_1\dot{y}$,

the zero dynamics of the system defined by (2.24) and (2.26) can be expressed as

$$\dot{z}_2 = \frac{-z_2}{c_1}. \tag{2.31}$$

Obviously, (2.31) is asymptotically stable for $c_1 > 0$. □

Theorem 2 is equivalent to the condition that the zero dynamics of the system defined by (2.24) and (2.26) is asymptotically stable when the output is y.

3. Parameter Setting

There are three parameters to configure an AIMD controller with the switching function SF2: c_1, α and β. First, we consider c_1, which determines the dynamics of the cruise mode. Generally, we set $c_1 = 1$ in experiments. If it is set too large, there will be significant damping effect on the sliding mode. If c_1 is set too small, it would require significant increase (decrease) of the α (β), in order to guarantee stability, i.e., see (2.36) and (2.37), without running into implementation problems.

Next, we discuss how to determine α and β values in AIMD. By definition, the AIMD adjustment rules can be expressed as

$$u_i(t) = \begin{cases} \alpha, & \text{if } S(t - \tau_{Bi}) < 0, \\ -\dfrac{\lambda_i(t)}{\beta}, & \text{else,} \end{cases} \tag{2.32}$$

where τ_{Bi} is the backward delay from the link to source i. In selecting the values of α and β, one needs to ensure that the system (including the AIMD controller) is asymptotically stable, and that it converges in the shortest time.

For practicality reasons, we want to derive a simple design rule of α and β that can adequately address our system needs, without overly complicating the analysis.

First, we consider the case of homogeneous sources. That is,

$$\zeta \cong N\lambda_i. \tag{2.33}$$

The simulation results show that the impact of delay is *relatively* small comparing with relative degree, and thus, we omit the delay terms in the subsequent discussion. We will discuss delay effects later. To satisfy the sliding condition (2.19), we discuss the following cases:

- When $S < 0$, $\dot{S} > \eta$. From (2.29), (2.25), (2.32) and (2.33), we have

$$\alpha > \frac{\eta - (1 - c_1 C\frac{\partial G}{\partial q}\dot{q}) - c_1\dot{\hat{C}}}{(1 - C\frac{\partial G}{\partial \zeta})c_1 N}. \tag{2.34}$$

- Similarly, when $S > 0$, $\dot{S} < -\eta$. We have

$$\frac{1}{\beta} > \frac{\eta + (1 - c_1 C\frac{\partial G}{\partial q}\dot{q}) + c_1\dot{\hat{C}}}{(1 - C\frac{\partial G}{\partial \zeta})c_1 \zeta}. \tag{2.35}$$

where $\eta > 0$. Eqs. (2.34) and (2.35) together guarantee the sliding mode exists. The state trajectory will reach $S = y + c_1\dot{y} = 0$ within time $\frac{|S_0|}{\eta}$, S_0 is the initial value of S. This condition gives a clear guideline on how long one needs to wait before the system can reach its sliding mode from an initial state. Based on Theorem 2, the asymptotical stability of the sliding mode is guaranteed, and we directly arrive at the following theorem:

Theorem 3 *Under the AIMD control law (2.32) with a switching function $S = y + c_1\dot{y}$, $c_1 > 0$, the queuing dynamic system defined by (2.24) and (2.26) is stable if (2.34) and (2.35) hold.*

Note that the stability conditions (2.34) and (2.35) define the attractive zone of

the switching manifold $S = y + c_1\dot{y} = 0$. Within the attractive zone, $|S|$ monotonically decreases until reaching the switching manifold. Increasing α and $1/\beta$ values will increase the size of the attractive zone, and also increase the rate of convergence to the sliding mode (see (2.29)). However, due to delay, inaccurate measurement, etc., the switching between increase and decrease cannot take place on the switching manifold perfectly, and it causes chattering around the switching manifold. The degree of chattering is proportional to the values of α and $1/\beta$.

Theorem 4 *Given that a desired attractive zone is Z_A and disturbance $\hat{C}$ is bound within D, we should choose the following α and $1/\beta$ values to guarantee that the switching manifold is attractive within Z_A:*

$$\alpha = \sup_{\{q,\zeta\}\in Z_A} \left[\frac{\eta - (1 - c_1 C\frac{\partial G}{\partial q})\dot{q} - c_1\dot{\hat{C}}}{(1 - C\frac{\partial G}{\partial \zeta})c_1 N} \right], \tag{2.36}$$

$$\frac{1}{\beta} = \sup_{\{q,\zeta\}\in Z_A} \left[\frac{\eta + (1 - c_1 C\frac{\partial G}{\partial q})\dot{q} + c_1\dot{\hat{C}}}{(1 - C\frac{\partial G}{\partial \zeta})c_1 \zeta} \right]. \tag{2.37}$$

By using (2.34) and (2.35), it is trivial to prove that these conditions hold. For heavily-loaded cases, $G \cong 1$, and (2.36) and (2.37) can be simplified into

$$\alpha = \sup \left[\frac{\eta - \zeta + C - \hat{C} - c_1\dot{\hat{C}}}{c_1 N} \right], \tag{2.38}$$

$$\frac{1}{\beta} = \sup \left[\frac{\eta + \zeta - C + \hat{C} + c_1\dot{\hat{C}}}{c_1 \zeta} \right]. \tag{2.39}$$

When the disturbance is slow-changing the required attractive zone of the switching manifold is defined by $\zeta \in [k_1 C, k_2 C]$, where $0 < k_1 < 1 < k_2$, and $\hat{C} \in [0, k_3 C]$, $0 < k_3 < 1$. In this case, we ignore $\dot{\hat{C}}$ and simplify the equations for α and β, i.e.,

Eqs. (2.38), and (2.39), into the following simple forms:

$$\alpha \cong \frac{\eta + (1 - k_1)C}{c_1 N}, \tag{2.40}$$

$$\beta \cong c_1 \left[1 + \frac{k_3}{k_1} - \frac{1}{k_2} + \frac{\eta}{k_1 C}\right]^{-1}. \tag{2.41}$$

At the equilibrium point, when the system remains stable, it would be useful to keep the increase rate the same as decrease rate [37]. In that case, we can set

$$\alpha\beta \cong \frac{C}{N}. \tag{2.42}$$

If this constraint is adopted, then one of the two equations (2.40) and (2.41) can be omitted.

I note that in addition to the theoretical bounds derived above, one needs to exercise caution in physical design, to make sure that the degree of oscillation within the guaranteed zone is acceptable. For instance, the α and $1/\beta$ values used in the example of Fig. 12 are smaller than the bounds defined in (2.38) and (2.39). Even though the system eventually converges, the level of oscillation is not negligible.

G. Analysis for Window-Based Systems

In this section, we present theoretical analysis of the window-based AIMD systems to prove their asymptotic stability conditions, and other related issues.

1. Window-Based Network Model

For a window-based AIMD system, our ns-2 simulations show that its relative degree is *one*. This finding contradicts the widely adopted model (2.43) where the relative degree is two, because one must differentiate the output q twice until the control

variable $\dot{W}_i$ explicitly appears.

$$\dot{q} = \sum_{i=1}^{N} \frac{W_i}{R_i} - C. \tag{2.43}$$

Moreover, according to (2.43), the total input rate would increase until overshooting the switching line, while ns-2 simulation results showed that the total input rate stopped increasing far before crossing the switching line (see subsection 2 of section E). Here, we suggest a simple alternative to eliminate the discrepancy between (2.43) and simulations. Note that (2.43) characterizes the fact that within a round trip time R_i, sender i sends out W_i packet, but it does not reflect the fact that the total ACK packet rate equals to the congested link's capacity when heavily loaded. By the self clocking property, a new packet does not enter the network until an ACK packet is received. Omitting this property results in the discrepancy between (2.43) and ns-2 simulations. To improve this model, within time period Δt, sender i gets m_i ACKs and increases its congestion window by ΔW_i, so that it sends out $m_i + \Delta W_i$ new packets within time period Δt. Then we have

$$\Delta q(t) = \sum_{i-1}^{N} [m_i(t - \tau_{Fi}) + \Delta W_i(t - \tau_{Fi})] - C\Delta t, \tag{2.44}$$

where τ_{Fi} is session i's forward transmission delay from sender i to the congested link. By the self-clocking property we have

$$\Delta q(t) = \sum_{i-1}^{N} [m_i(t - \tau_{Fi}) + \Delta W_i(t - \tau_{Fi})] - C\Delta t, \tag{2.45}$$

where τ_{Bi} is session i's transmission delay from the congested link to receiver i and then to sender i, $C_i(t)$ is the portion of the congested link's capacity consumed by

session i at time instant t, where

$$\sum_{i-1}^{N} C_i(t) = C. \tag{2.46}$$

Therefore,

$$\Delta q(t) = \sum_{i-1}^{N} \Delta W_i(t - \tau_{Fi}) + \varpi \Delta t, \tag{2.47}$$

where $\varpi \equiv \sum_{i-1}^{N} C_i(t - \tau_{Fi} - \tau_{Bi}) - C$. Thus, in the heavily loaded cases,

$$\dot{q}(t) = \sum_{i-1}^{N} \dot{W}_i(t - \tau_{Fi}) + \varpi. \tag{2.48}$$

ϖ can be viewed as a noise due to heterogeneous roundtrip delays. The relative degree of (2.48) is *one* when control input is $\dot{W}_i$ and output is q, which is consistent with ns-2 simulations.

Theorem 5 *The relative degree of the system defined by (2.48) is one, when the output is y and control variable is $\dot{W}_i$, respectively.*

Proof: We just need to differentiate y once to make $\dot{W}_i$ explicitly appear. □

With (2.48), it is easy to explain why the total input rate is bounded for window-based systems. By definition, the window-based AIMD adjustment rules [24] can be expressed as

$$u_{wi} \equiv \dot{W}_i = \begin{cases} \alpha_w, & \text{if } S(t - \tau_{Bi}) < 0, \\ -\dfrac{W_i(t)}{\beta_w}, & else. \end{cases} \tag{2.49}$$

Eq. (2.7) is a special case of (2.49) with $S = q - q_d$. Substituting (2.49) into (2.48), we get

$$\dot{q}(t) = N\alpha_w + \varpi. \tag{2.50}$$

in the increase mode. Therefore, the total input rate

$$\zeta = \dot{q}(t) + C = N\alpha_w + \sum_{i-1}^{N} C_i(t - \tau_{Fi} - \tau_{Bi}). \tag{2.51}$$

Eq. (2.51) clearly shows that the total input rate is bounded when N is constant because α_w is constant and all $C_i(t - \tau_{Fi} - \tau_{Bi})$ is bounded, which means the total input rate does not increase with window sizes W_i.

Theorem 6 *The zero dynamics of the system defined by (2.48) is asymptotically stable when the output is* $S = q - q_d$.

Proof: (2.48) is a linear equation. It is minimum phase. □

2. Window-Based Parameter Setting

Two parameters α_w and β_w need to be configured for a window-based AIMD controller with the switching function SF1. In selecting the values of α_w and β_w, one needs to ensure that the system (including the AIMD controller) is asymptotically stable, and converges in minimum time. Similar to the rate-based cases, we first consider the case of homogeneous sources and omit the delay terms. For SF1, we get

$$\dot{S} = \dot{q}(t) = N\dot{W}_i + \varpi. \tag{2.52}$$

According to (2.19), we discuss the following two cases:

- When $S < 0$, $\dot{S} > \eta$, from (2.48), we have

$$\alpha_w > \frac{\eta - \varpi}{N}. \tag{2.53}$$

- When $S > 0$, $\dot{S} < -\eta$, we have

$$\frac{1}{\beta_w} > \frac{\eta + \varpi}{NW_i}. \tag{2.54}$$

where $\eta > 0$. Eqs. (2.53) and (2.54) together guarantee the sliding mode is attractive and the state trajectory will reach $S = y = 0$ within time $\frac{|S_0|}{\eta}$, from an initial state $S = S_0$. Based on Theorem 6, the asymptotical stability of the sliding mode is guaranteed. Based on these results, we directly arrive at the following theorem:

Theorem II.1 *Under the AIMD control law (2.49) with a switching function $S = y$, the queuing dynamics defined by (2.48) is stable if (2.53) and (2.54) hold.*

Inequalities (2.53) and (2.54) define the attractive zone of the switching manifold $S = y = 0$. Increasing α_w and $1/\beta_w$ values will enlarge the attractive zone, and also increase the rate of convergence to the sliding mode (see (2.52)). Similar to rate-based systems, the degree of chattering due to delay or inaccurate measurements is proportional to the values of α_w and $1/\beta_w$.

H. Transient Behavior and Delay Effects

η determines the speed of convergence toward the switching manifold. Within the cruise mode, the system dynamics is shaped by the switching function. To reduce the system response time, we prefer a larger η, or equivalently, larger α and $\beta - 1$. On the other hand, in order to reduce the chattering caused by delay, we need smaller α and $\beta - 1$. Clearly, there is a need to balance the two conflicting factors. A tradeoff is made by using the *boundary layer* method [2][3]. It has been proven that this method can contain the oscillation in a bounded range. The basic idea of the boundary layer method is to suppress chattering by smoothing the discontinuity of the control within the cruise mode. It is based on an observation that the control variable should be reduced with S approaching to S_0 defined as $(S_1 + S_2)/2$. As such, we adopt the

following cruise mode control law:

$$u_{ci} = \begin{cases} \alpha \left| \dfrac{S - S_0}{S_1 - S_0} \right|, & S_1 \leq S \leq S_0, \\ -\dfrac{\lambda_i}{\beta} \left| \dfrac{S - S_0}{S_2 - S_0} \right|, & S_0 < S \leq S_2. \end{cases} \tag{2.55}$$

I empirically choose $S_1 = Q/4$ and $S_2 = (3Q)/4$. If $(S_2 - S_1)$ is too small, the performance effect is not obvious. But if too large, it tends to reduce the sensitivity of the control. α and β are determined based on (2.40) and (2.41). To implement (2.55), it needs to use explicit feedback messages that carry the numerical values of $\left| \dfrac{S - S_0}{S_1 - S_0} \right|$ or $\left| \dfrac{S - S_0}{S_2 - S_0} \right|$ to the sources for execution.

I. Conclusion

In this chapter, we developed a general I-D congestion control model that characterizes the control dynamics, queuing dynamics, and the predominating system performance factors. We proved that the relative degree of rate-based congestion control systems is two while that of the window-based congestion control systems is one. We devised a design methodology to guarantee the asymptotic stability of the congestion control system. Without such considerations, heuristic design approaches to optimization of congestion control systems will only have limited effects.

With our model, the complex interplay between different system parameters can be governed via a few simple equations and conditions. Our method permits one to make the tradeoff between the amount of feedback information, delay, and the expected performance effects, under explicitly defined conditions. There is little restriction on the system model, making it broadly applicable to a wide range of congestion control systems.

CHAPTER III

DEFENSE OF THE DISTRIBUTED DENIAL OF SERVICE ATTACKS

In this chapter we propose a coordinated defense scheme of *distributed denial of service* (DDoS) network attacks, based on the backward-propagation, on-off control strategy. When a DDoS attack is in effect, a high concentration of malicious packet streams are routed to the victim in a short time, making it a *hot spot*. The performance impact on the hot spot is related to the total *hot packet rate* that can be tolerated by the victim. Based on the *sliding mode control* (SMC) theory, we present a backward pressure propagation, feedback control scheme to defend DDoS attacks. We develop our solutions based on the generic discontinuous flow control model developed in chapter II to analyze the dynamics of network traffic, and develop the algorithms for rate-based and queue-length-based feedback control. We show a simple design to implement our control scheme on a practical switch queue architecture.

A. Introduction

Distributed Denial of Service (DDoS), is a relatively simple, yet very powerful technique to attack Internet resources. With little or no advance warning, a DDoS attack can abruptly drain the computing and communication resources of its victim within a short time, until the attack is resolved. Even some of the largest computer makers and web-based service providers are not immune from this problem [38][39][40]. DDoS exploits the inherent weakness of the Internet system architecture and its open resource access model. Unless special care is taken, a DDoS victim can suffer from damages ranging from system shutdown, file corruption, and total or partial loss of services. Even if the victim can cope with the surged demands, the unwanted, dis-

rupting packet flows often lead to serious performance degradation. DDoS attacking programs have very simple logic structures and small memory sizes, making them relatively easy to implement and hide. Given the openness of the Internet, a malicious hacker can gradually scan the Internet to plant attacking programs into unprotected hosts, at relatively low risks of being noticed. After the attacker accumulates enough compromised hosts, he just needs to issue the attacking command to a coordinating machine, which then wakes up the dormant attacking programs to hit the victim(s) from everywhere. The attacker can set the actual attack time so far away from release of attack commands that it becomes virtually impossible to trace the real attackers. It is trivial to hit victims at scheduled strengths and durations.

Several DDoS attack schemes [41][42][43][44][45][46][47] have been widely published on the Internet, with many others being developed. The first is UDP flooding in which the victim(s) is flooded with UDP packets. TCP SYN flooding exploits the 3-way handshaking procedure in the TCP protocol. The attacker merely sends large amounts of SYN packets, with spoofed source addresses, to the victim requesting for connections. The victim may quickly exhaust its resources if a large number of bogus connection requests are not resolved in time. A brute force attack is simply sending a large number of ACK/data packets to the victim even without established TCP connections. This attack approach is designed to suffocate the communication channels of the victim. The "ping of death" attack is to flood the victim with the ICMP echo packets, until it brings an unprepared host to its knee. A simple variation of the ping of death is to set the spoofed address to that of another victim. This way, receivers of both the ICMP echo and ICMP echo reply packets will suffer from high performance losses.

As mentioned earlier, the cooperative resource access model of the Internet architecture is the root of the DDoS problems. Numerous operational solutions are pub-

lished for the Internet community [48][49][50][51][52][53][54][55][56][47][57][58]. We note that it is possible to defend DDoS attacks using simple flow control techniques, with little or no modifications to the IP protocols. The center issue here is how to design an efficient congestion controller that can deal with ill posed traffic conditions.

DDoS management can be considered a special case of the flow control problem, which has been widely studied, e.g. [1],[59],[60],[21],[61],[30],[31], etc. The optimal control theory is proposed in [1] to manage the network traffic, on the basis of a nonlinear dynamic flow model. This technique needs to use explicit parameters of the network model, making it of limited practical values in real systems, because of its high sensitivity to parameter uncertainties. Binary feedback control [7] is widely used in network traffic control for its simplicity and efficiency. It relies upon a one-bit indicator in each probing packet to determine the congestion status of a network resource. The key design issue here is the threshold value(s) to set and reset the indicator bit. When combined with the sliding mode control approach [2][3][36][62], they become a very effective and robust approach to optimize the threshold(s) of the binary feedback control scheme. The advantages of the sliding mode control scheme are its simplicity and robustness to parameter and model uncertainty.

In this chapter, we develop a backward-propagation feedback control strategy for DDoS defense. When a host finds itself becoming a hot spot, it informs neighboring nodes and routers to reduce influx of attacking packets. By using the notion of relative degree (see [3],[63] or Appendix A) of nonlinear dynamic systems, we develop a simple on-off control strategy that does not require the use of a precise traffic model for such information like attack sources, and system configuration. A routing device just needs to be informed of the acceptable level of traffic rates from its (destination) neighbors, so that it can set the acceptable input rates to its (source) neighbors. Clearly, selection of the on and off thresholds is a key factor. The obvious selection of

using one fixed-goal threshold variable for throttling is unrealistic because of the sizes and complex behavior of a large network. One can see that this control scheme is not involved with any routing information, but is merely based on the mutually agreed throughput levels for coordinated defense of the DDoS attacks. We derive the stable (oscillation free, asymptotically convergent) conditions to turn on-off throttling.

The rest of this chapter is organized as follows. Section B of this chapter presents the network model being considered, and the basic characteristics of a generic routing device. section C discuss the rate-based and queue-based control algorithms. A simple architecture to implement the flow control algorithms is presented in Section D, and the chapter concludes in section E.

B. System Model

Our model is aimed at a macroscopic solution approach. We assume that participating routing devices support flow control for their input and output ports. When a peer relationship is established between two nodes, they execute flow regulating policies periodically. For its enormous sizes, and the significant differences of routing devices, it is impractical to take the entire Internet into a single model for our design. Instead, we propose a divide and conquer approach to partition the global network into smaller clusters, so that the DDoS defense can be implemented in a hierarchical manner. As a heuristic choice, and without loss of generality, we propose to define a cluster as three connected nodes, see Fig. 16. It is not difficult to expand the model to 2, 4, or 5-node based clusters.

In this chapter, we focus on the intra-cluster analysis, where the delays for data exchange and control actions are negligible. The inter-cluster coordinated defense needs to use different design methodologies and solution algorithms, because of their

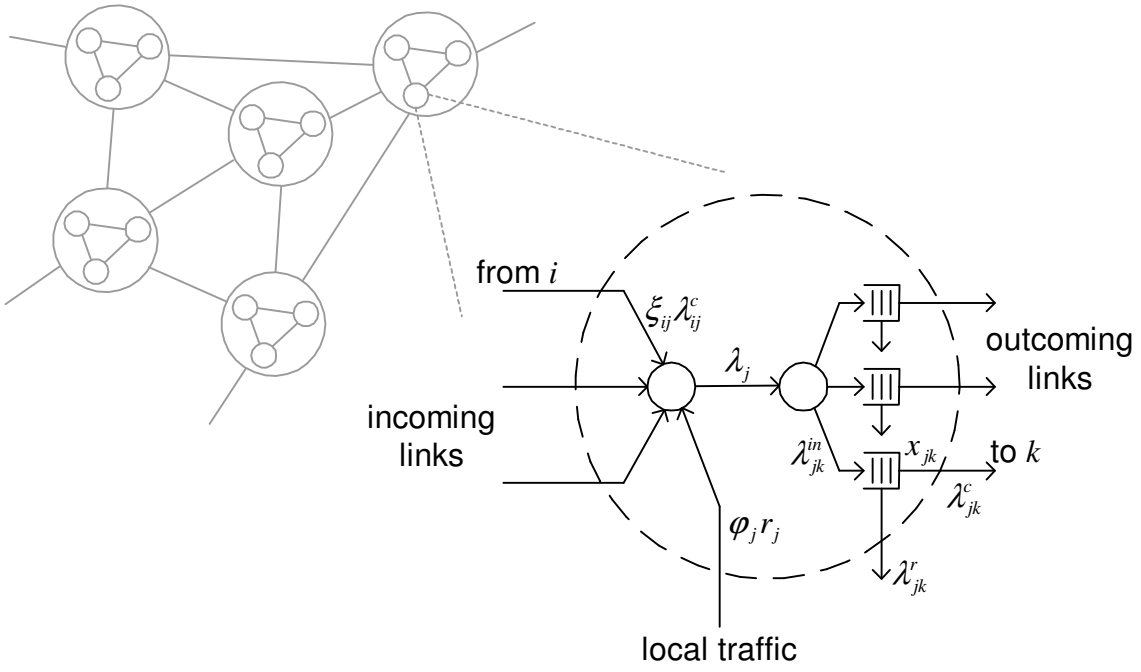

Fig. 16. The network model for backward propagation, feedback control, and the generic switch architecture and its packet flow model in the network.

much larger capacity, delays, and traffic volume. It is also necessary to better understand the relative degree properties of large networks, in order to prevent the coordinated defense algorithms from oscillation and other well known stability issues.

Let us now consider the switch architecture in the cluster. Modern switching fabrics have high throughput with small blocking probabilities . Most of the packet delays in the routing devices are due to buffering and channel contention. Despite the obvious benefit of using dedicated resources for channels in the defense of DDoS, the shared memory-bus architecture is the choice of architecture in practice, for its better flexibility and lower costs. As a result, we adopt the switch architecture shown in Fig. 16, and the packet flow model proposed in [1]. We note that when an IP resource, e.g., a popular Internet web site, is under DDoS attack, its neighbors may or may not experience heavy traffic. A feedback control strategy based on the relative

performance measure of a routing device, e.g., "the device is experiencing unusually high traffic," is not a reliable indicator for detection of DDoS attacks, because such observation could be caused by harmless surge of the service requests. On the other hand, if a DDoS attack has been positively identified, such an indicator (traffic surge) can be an excellent cue to trace and curtail the DDoS attack. Once an attack is confirmed, the backward prorogation control rules should be engaged, regardless of the current workload condition. Although not explicitly discussed here, one can use propagation paths of the feedback control signals to determine the sources, routing, density and frequency of the attack.

For packet throttling, we use the sliding mode control theory to construct rate-based, and queue-length-based feedback control strategies. In practice, some routing devices that support layer-4 switching would be able to differentiate malicious packets but most others not. In our study, we use the same system models but different switching functions to analyze both cases. One might speculate that given that the DDoS streams are identified, why not just dropping their packets. We note that this approach implies that the routing devices must have full knowledge of the DDoS attacks, i.e., packet types, interactive sequences between the attacking machines and the victim, session states, etc., to avoid dropping regular packets. Furthermore, without packet dropping, one can more reliably push the throttling pressure back to the DDoS sources. An unexpected advantage of this non-dropping scheme is that we can easily generalize the DDoS defense solutions for congestion prevention and flow control during normal operations.

The backward-propagation feedback control is inexpensive and easy to implement. One key issue for the backward-propagation feedback control strategy is its stability. It was observed in [64] that, even for the highly regular multiprocessor environment, a backward-propagation congestion control algorithm may oscillate or

even break down at change of workload conditions. Due to the highly dynamic nature of the network traffic, we adopt the SMC model of chapter II for congestion control that only makes use of the only output measurements (traffic rate or buffer queue length) without considering the complete system states. As discussed in chapter II when the sliding controller is applied to a system with relative degree r, one must use $0, 1, \cdots, (r-1)^{th}$ order derivatives of the outputs, to construct the sliding manifolds [65][66]. The other condition [65][66] for system stability is that the corresponding zero dynamic is stable, or equivalently, the system is minimum-phase [3][63]. The minimum-phase property can be obtained through proper selection of the outputs. The third factor that commonly leads to oscillation is the control delay, which when large enough prevent switching from taking place on the switch manifold(s), and produce chattering.

1. Fluid Dynamic Model [1]

In this chapter, we adopted the notations and system conditions from [1], where multiple input flows enter the switch from directly connected hosts or other switches, and then be routed to the output queues. We denote the collection of nodes by $N = \{N_i, N_j, N_k, \cdots\}$, and the collection of unidirectional links by $L = \{L_{ij}, L_{jk}, \cdots\}$. The volume of total traffic entering N_j is

$$\lambda_j(t) = \varphi_j(t) r_j(t) + \sum_i \xi_{ij}(t) \lambda_{ij}^c(t), \tag{3.1}$$

where

$r_j(t)$ is the rate of packets arriving at N_j from outside of the cluster,

$\varphi_j(t)$ denotes the admitted portion of input traffic, $r_j(t)$,

$\lambda_{ij}^c(t)$ is the traffic demand generated by $N_i$to N_j,

$\xi_{ij}(t)$ denotes the admitted portion of the traffic from N_i to N_j, $\xi^c_{ij}(t)$.

The first-order equilibrium equations for $L_{j,k}$ are defined in (3.2) to (3.5).

$$\dot{x}_{jk}(t) = -\mu_{jk}(t)G_{jk}(x_{jk}(t)) + \lambda^{in}_{jk}(t), \tag{3.2}$$

where the occupancy of the buffer associated with $L_{j,k}$ is denoted as x_{jk}, μ_{jk} ($\mu_{jk} > 0$) the bandwidth of $L_{j,k}$, and the function $\mu_{jk}G_{jk}(x_{jk})$ is the outbound traffic:

$$\mu_{jk}(t)G_{jk}(x_{jk}(t)) = \lambda^r_{jk}(t) + \lambda^c_{jk}(t). \tag{3.3}$$

The traffic rejected at $L_{j,k}$ is

$$\lambda^r_{jk}(t) = \mu_{jk}(t)H_{jk}(x_{jk}(t))G_{jk}(x_{jk}(t)), \tag{3.4}$$

where $H_{jk}(x_{jk})$ specifies the rejected fraction of the outbound traffic $\mu_{jk}G_{jk}(x_{jk}(t))$, $0 \leq H_{jk}(x_{jk}(t)) \leq 1$. The traffic from N_j to N_k

$$\lambda^c_{jk}(t) = \mu_{jk}(t)\{1 - H_{jk}(x_{jk}(t))\}G_{jk}(x_{jk}(t)) \equiv \mu_{jk}(t)\hat{G}_{jk}(x_{jk}(t)). \tag{3.5}$$

For a real system, $G_{jk}(x_{jk})$ must be positive, moreover, $G_{jk}(x_{jk})$ and $H_{jk}(x_{jk})$ are defined for $x_{jk} \geq 0$. Therefore, $\hat{G}_{jk}(x_{jk}) \equiv [1 - H_{jk}(x_{jk})]G_{jk}(x_{jk})$ is positive also. In subsequent discussions, we assume that $\dfrac{\partial \hat{G}_{jk}}{\partial x_{jk}}$ (This is reasonable, because outgoing traffic increases with buffer occupancy, and vice versa) and $\hat{G}_{jk}(x_{jk})$ can be differentiated twice with respect to x_{jk}. Given that

$$\lambda^{in}_{jk}(t) = \alpha_{jk}(t)\lambda_j(t), \tag{3.6}$$

where $\alpha_{jk}(t)$ denotes the portion of the total traffic $\lambda_j(t)$ entering N_j, which is routed

to N_k. So the following relations hold for $L_{j,k}, \forall j, k$,

$$\dot{x}_{jk} = -\mu_{jk} G_{jk}(x_{jk}) + \alpha_{jk}[\varphi_j r_j + \sum_i \xi_{ij} \mu_{ij} \hat{G}_{ij}(x_{ij})], \tag{3.7}$$

where

$$\begin{aligned} &0 < \alpha_{jk}(t) \leq 1, \\ &0 \leq \varphi_j(t) \leq 1, \\ &0 \leq \xi_{ij}(t) \leq 1, \\ &\textstyle\sum_k \alpha_{jk}(t) = 1. \end{aligned} \tag{3.8}$$

The physical meaning of the above constraints is obvious and is reasonable. For packet throttling, φ_j and ξ_{ij} are the natural choices of control variables. When $\varphi_j = 1$ and 0, respectively, it means that all of input traffic, $r_j(t)$, is accepted and rejected. A similar argument applies to ξ_{ij}. If φ_j ($N_j \in N$) and ξ_{ij} ($L_{i,j} \in L$) are all used as control variables, it means that the entire cluster is fully controlled. If we just use φ_j ($N_j \in N$) as control variables, it is the case of "edge control" which means we just throttle the traffic entering the network.

C. Backward Propagation Feedback Control

In this section, we discuss the intra-cluster, rate-based and queue-based backward propagation feedback control algorithms using sliding mode control. We assume that the time difference between nodes in the same cluster is negligible, and the packet rate information of nodes in a cluster can be obtained and exchanged at reasonable costs. For the rate-based solution, the primary control and measurement parameter is the flow rate of a particular link. For the queue-based solution, we use the queue length/occupancy of a buffer as the performance indicator. We found that the relative degree of the rate-based approach is one, so we need not use the derivative of the

packet flow rate for the control algorithm. On the other hand, the relative degree of the queue-based scheme is two, implying that we do need to obtain the first derivative of the queue length measurement for the flow control algorithm. Intuitively, the first approach is more suitable for low traffic intensity situations, because of the lower levels of packet queuing. The second approach, on the other hand, is more suitable for high intensity attacks, because packet queue lengths become very good indicators of the congestion/traffic level.

It is relatively simple for a DDoS victim to detect the presence of the attack: many of regular users cannot access the service, and/or their productivity measures drop abruptly. It is much more difficult, sometimes impossible, to trace the exact routes of attacks, especially when the attacker is able to change the attack patterns. Obviously, the effectiveness of the defense is proportional to the number of routing devices on the affected hot paths that participate in the defense. We note that a routing device may not have full knowledge of the hot and cold streams that pass through it. Under such a condition, it is not possible to have perfect control of the backward pressure propagation, but rather, we will need to rely on a search approach to identify the attacking paths and apply the greatest backward pressure to the strongest attack sources. One can achieve this objective by combining a classical depth-first or breadth first search algorithm, and a two way handshaking process between a routing device and the victim. Here, we focus on the traffic stability of the on-off control algorithm to avoid oscillation of traffic in both the victim and the related network resources.

The sliding mode, on-off control scheme is very simple and inexpensive. The feedback controller defines a switch function s, which is a function of the outputs (rates, queue length, etc.) and possibly their derivatives of the proper relative degree. At each control time instance, the system measures the new traffic value, possibly

after filtering of short-term bursts, examine the required backward pressure between nodes. The value of s changes with time, and the control objective is to force s value toward the value of 0 at each time instance of the control action. By the on-off control rule, in an ideal system where only on and off are permissible actions, the throttling turns off when $s < 0$ (all traffic permitted), and throttling turns on when $s > 0$ (all traffic blocked.) That is, on the state-space trajectory of s, the control action has a step jump at $s = 0^+$ and $s = 0^-$ that will lead to chattering problem. To avoid bursty traffic, the on-off control actions can be eased by continuous smoothing. That is, we change the control actions from a step jump to a slope function, so that throttling levels adjust with the s values, to eliminate oscillation, see Section D for more details. In the analysis, we assume that the bursty traffic is adequately smoothed to represent the true demand levels.

1. Rate-Based Control Algorithm

Consider the cluster topology in Fig. 17, assuming that the hot spot N_3 and the hot packet streams have been identified. Two different types of hot packet sources are considered: hosts and network routing devices. r_1 and d_1 are respectively the hot and cold input rates from *src1* to N_1. r_2 and d_2 are defined in a similar manner. Now, let us consider the dynamic traffic flow model of the above topology, which is derived

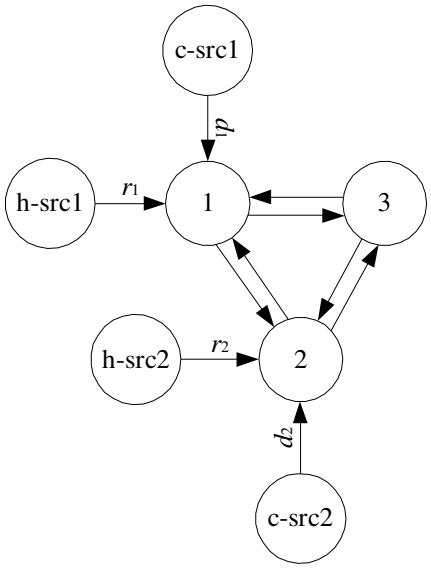

Fig. 17. The flow balance diagram within a cluster.

from the basic balance equation in (3.7):

$$\begin{cases} \dot{x}_{12} = -\mu_{12}G_{12} + \alpha_{12}[d_1 + \varphi_1 r_1 + \mu_{21}\hat{G}_{21} + \mu_{31}\hat{G}_{31}], \\ \dot{x}_{13} = -\mu_{13}G_{13} + \alpha_{13}[d_1 + \varphi_1 r_1 + \mu_{21}\hat{G}_{21} + \mu_{31}\hat{G}_{31}], \\ \dot{x}_{21} = -\mu_{21}G_{21} + \alpha_{21}[d_2 + \varphi_2 r_2 + \mu_{12}\hat{G}_{12} + \mu_{32}\hat{G}_{32}], \\ \dot{x}_{23} = -\mu_{23}G_{23} + \alpha_{23}[d_2 + \varphi_2 r_2 + \mu_{12}\hat{G}_{12} + \mu_{32}\hat{G}_{32}], \\ \dot{x}_{31} = -\mu_{31}G_{31} + \alpha_{31}[\mu_{13}\hat{G}_{13} + \mu_{23}\hat{G}_{23}], \\ \dot{x}_{32} = -\mu_{32}G_{32} + \alpha_{32}[\mu_{13}\hat{G}_{13} + \mu_{23}\hat{G}_{23}]. \end{cases} \tag{3.9}$$

Throttling applies only to the hot packet rates, r_1 and r_2. This implies that for equations (3.9), the coefficients of r_1 and r_2 are φ_1 and φ_2 respectively, but the coefficients of d_1 and d_2 are one. We note that, it may be difficult to determine the explicit forms of $G_{ij}(x_{ij})$ and $H_{ij}(x_{ij})$ for real systems. Here, according to equations (3.1) and (3.5), and considering $\xi_{ij} = 1$(the edge control case), the total input rate to

the hot spot, N_3, is

$$\lambda_3 = \mu_{13}\hat{G}_{13} + \mu_{23}\hat{G}_{23} \equiv y. \tag{3.10}$$

After some simple reorganization, we can rewrite the above equations into the following forms,

$$\begin{cases} \dot{x}_{12} = f_{12}(x_{12}, x_{21}, x_{31}) + \alpha_{12}d_1 + \alpha_{12}r_1\varphi_1, \\ \dot{x}_{13} = f_{13}(x_{13}, x_{21}, x_{31}) + \alpha_{13}d_1 + \alpha_{13}r_1\varphi_1, \\ \dot{x}_{21} = f_{21}(x_{21}, x_{12}, x_{32}) + \alpha_{21}d_2 + \alpha_{21}r_2\varphi_2, \\ \dot{x}_{23} = f_{23}(x_{23}, x_{12}, x_{32}) + \alpha_{23}d_2 + \alpha_{23}r_2\varphi_2, \\ \dot{x}_{31} = f_{31}(x_{31}, x_{13}, x_{23}), \\ \dot{x}_{32} = f_{32}(x_{32}, x_{13}, x_{23}). \end{cases} \tag{3.11}$$

where

$$f_{12}(x_{12}, x_{21}, x_{31}) \equiv -\mu_{12}G_{12}(x_{12} + \alpha_{12}\mu_{21}\hat{G}_{21}(x_{21}) + \alpha_{12}\mu_{31}\hat{G}_{31}(x_{31}). \tag{3.12}$$

f_{13}, f_{21}, f_{23}, f_{31} and f_{32} are defined similarly. The arguments of these functions will be omitted in later discussion, whenever it is convenient and without risk of confusion.

From the viewpoint of a control system, the defense of DDoS attacks is a strongly-coupled, nonlinear system. In addition, the system parameters, e.g. $G_{ij}(x_{ij})$ and $H_{ij}(x_{ij})$, may not be easily determined in an explicit form. Our goal is to keep the total input rate of the hot spot N_3 at a desired value, y_d, through real-time adjustment of φ_1 and φ_2 according to current traffic conditions. Considering the fact that y monotonically increases with φ_1 and φ_2, we make a reasonable assumption that system (3.9) or (3.11) is minimum phase. Here, we can express the control objective by the following equation.

$$\hat{y} = y - y_d, \tag{3.13}$$

$$\begin{aligned}
&\dot{\hat{y}} \equiv \dot{y} \\
&= \mu_{13}\frac{\partial \hat{G}_{13}}{\partial x_{13}}\dot{x}_{13} + \mu_{23}\frac{\partial \hat{G}_{23}}{\partial x_{23}}\dot{x}_{23} \\
&= \mu_{13}\frac{\partial \hat{G}_{13}}{\partial x_{13}}(f_{13} + \alpha_{13}d_1 + \alpha_{13}r_1\varphi_1) \\
&+\mu_{23}\frac{\partial \hat{G}_{23}}{\partial x_{23}}(f_{23} + \alpha_{23}d_2 + \alpha_{23}r_2\varphi_2).
\end{aligned} \tag{3.14}$$

The control variables φ_1 and φ_2 appear in the first order derivative of $\hat{y}$, so the relative degree is one. According to the sliding-mode control rules, one can select $s = \hat{y}$ as the threshold function ($s = 0$ defines the sliding manifold). Intuitively, when $s > 0$, we adjust φ_1 and φ_2 to make $\dot{s} < 0$. On the other hand, when $s < 0$, we select φ_1 and φ_2 to satisfy $\dot{s} > 0$.

Case 1: For $s > 0$ or $y > y_d$,

$$\dot{s} = \dot{\hat{y}} = \dot{y} < 0, \tag{3.15}$$

where $\dot{\hat{y}}$ is defined in (3.14). Given that $\frac{\partial \hat{G}_{ij}}{\partial x_{ij}}$, α_{ij} and μ_{ij}, through some simple algebraic manipulation we set the constraints on control variables φ_1 and φ_2,

$$\begin{cases} \varphi_1 < -\frac{f_{13} + \alpha_{13}d_1}{\alpha_{13}r_1}, \\ \varphi_2 < -\frac{f_{23} + \alpha_{23}d_2}{\alpha_{23}r_2}, \end{cases} \tag{3.16}$$

to satisfy $\mu_{13}\frac{\partial \hat{G}_{13}}{\partial x_{13}}(f_{13}+\alpha_{13}d_1+\alpha_{13}r_1\varphi_1) < 0$ and $\mu_{23}\frac{\partial \hat{G}_{23}}{\partial x_{23}}(f_{23}+\alpha_{23}d_2+\alpha_{23}r_2\varphi_2) < 0$, and thus $\dot{s} < 0$.

Case 2: For $s < 0$ or $y < y_d$, we need to have

$$\dot{s} = \dot{\hat{y}} = \dot{y} > 0, \tag{3.17}$$

Through a similar argument as in the case of $s > 0$, we can derive the control variables

φ_1 and φ_2 such that

$$\begin{cases} \varphi_1 > -\dfrac{f_{13} + \alpha_{13} d_1}{\alpha_{13} r_1}, \\ \varphi_2 > -\dfrac{f_{23} + \alpha_{23} d_2}{\alpha_{23} r_2}, \end{cases} \tag{3.18}$$

to satisfy $\mu_{13} \dfrac{\partial \hat{G}_{13}}{\partial x_{13}} (f_{13} + \alpha_{13} d_1 + \alpha_{13} r_1 \varphi_1) > 0$ and $\mu_{23} \dfrac{\partial \hat{G}_{23}}{\partial x_{23}} (f_{23} + \alpha_{23} d_2 + \alpha_{23} r_2 \varphi_2) > 0$, and thus $\dot{s} > 0$.

A main concern of the control variable setting is the system stability. If and only if when (3.15) and (3.17) are satisfied, $s\dot{s} < 0$ holds. This means that $\dfrac{d}{dt} s^2 < 0$, and by the Lyapunov stability law (see Appendix B), the system will converge to $s = 0 (y = \hat{y}$ under the control strategy (3.15) and (3.17). Given that (3.16) and (3.18) have stricter constraints than (3.15) and (3.17), using the control rules (3.16) and (3.18) guarantees that s will asymptotically converge to zero, based on the Lyapunov conditions. When combined with (3.8), the final control strategy can be rewritten as

$$\begin{cases} 0 \leq \varphi_1 < min\{-\dfrac{f_{13} + \alpha_{13} d_1}{\alpha_{13} r_1}, 1\} \\ 0 \leq \varphi_2 < min\{-\dfrac{f_{23} + \alpha_{23} d_2}{\alpha_{23} r_2}, 1\} \end{cases} , \text{for } s > 0, \tag{3.19}$$

and

$$\begin{cases} 1 \geq \varphi_1 > max\{-\dfrac{f_{13} + \alpha_{13} d_1}{\alpha_{13} r_1}, 0\} \\ 0 \geq \varphi_2 > max\{-\dfrac{f_{23} + \alpha_{23} d_2}{\alpha_{23} r_2}, 0\} \end{cases} , \text{for } s < 0, \tag{3.20}$$

Note that, when $s > 0$, it is possible that $-\dfrac{f_{13} + \alpha_{13} d_1}{\alpha_{13} r_1} < 0$ or $-\dfrac{f_{23} + \alpha_{23} d_2}{\alpha_{23} r_2} < 0$, which means the cold traffic rate entering N_1 or N_2 is so large that the buffer occupancy at links $L_{1,3}$ or $L_{2,3}$ cannot be effectively reduced even if there is no hot packets entering N_1 ($\varphi_1 = 0$) or N_2 ($\varphi_2 = 0$). This could mean formation of a new hot spot, or misidentification of the hot spot. To bring down the traffic level, one should treat some portion of the cold traffic d_1 and d_2 the same as hot traffic, and throttle them accordingly. Similarly, when $s < 0$, it is possible that $-\dfrac{f_{13} + \alpha_{13} d_1}{\alpha_{13} r_1} > 1$ or

$-\frac{f_{23}+\alpha_{23}d_2}{\alpha_{23}r_2} > 1$. In this case, it implies that even if the hot traffic is not throttled ($\varphi_1 = 1$ or $\varphi_2 = 1$), $\dot{s}$ cannot be greater than zero, and thus is always less than zero. Under such conditions, the total input rate to the hot spot, N_3, cannot exceed the desired value $\hat{y}$, i.e., hot traffic r_1 and r_2 are not "hot" anymore.

As a final note, the relative degree of this 3-node is indeed one because the switching function turns out to be not related to any high order derivatives of the controlled variables. If the relative degree turns out to be two, we should redefine the switching function as $s = \hat{y} + \omega\dot{\hat{y}}(\omega > 0)$. For a larger system analysis, such as the inter-cluster model, it will be necessary to consider the minimum phase conditions for (3.9) or equivalently (3.11).

2. Queue-Based Control Algorithm

Now, let us consider the cluster in Fig. 18 where N_3 is the destination of DDoS attacks, and the control objective is aimed at keeping the length of two queues in N_3, x_{34} and x_{35}, at desired values $\hat{x}_{34}$ and $\hat{x}_{35}$, respectively. Assume that the hot packet streams have been identified, where r_1 and d_1 are respectively the hot and cold input rates from *src1* to N_1, and r_2 and d_2 are defined in a similar manner. In fact, N_3 can represent a sub-network with links $L_{1,3}$, $L_{2,3}$, $L_{3,4}$ and $L_{3,5}$ as the input and output links to the sub-network. N_3 could also represent a collection of nodes including the hot spot and its adjacent nodes that need to be protected from the attack. The

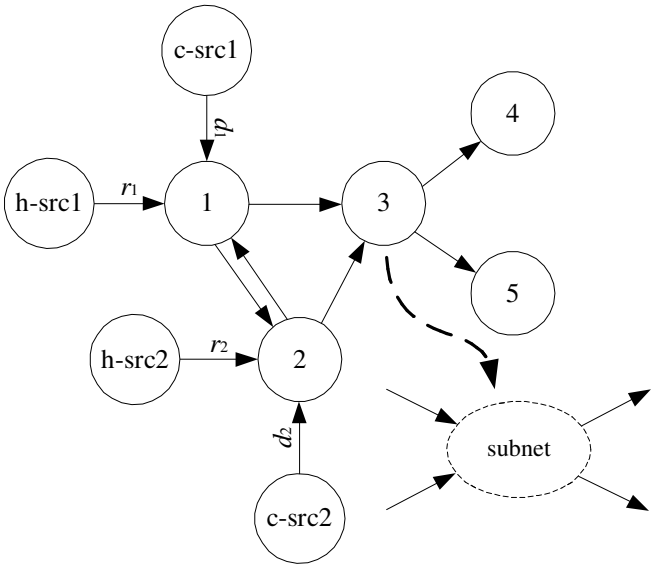

Fig. 18. The queue-based congestion control model.

system equilibrium equations are

$$\begin{cases} \dot{x}_{12} = -\mu_{12}G_{12} + \alpha_{12}(d_1 + \varphi_1 r_1 + \mu_{21}\hat{G}_{21}), \\ \dot{x}_{13} = -\mu_{13}G_{13} + \alpha_{13}(d_1 + \varphi_1 r_1 + \mu_{21}\hat{G}_{21}), \\ \dot{x}_{21} = -\mu_{21}G_{21} + \alpha_{21}(d_2 + \varphi_2 r_2 + \mu_{12}\hat{G}_{12}), \\ \dot{x}_{23} = -\mu_{23}G_{23} + \alpha_{23}(d_2 + \varphi_2 r_2 + \mu_{12}\hat{G}_{12}), \\ \dot{x}_{34} = -\mu_{34}G_{34} + \alpha_{34}(\mu_{13}\hat{G}_{13} + \mu_{23}\hat{G}_{23}), \\ \dot{x}_{35} = -\mu_{35}G_{35} + \alpha_{35}(\mu_{13}\hat{G}_{13} + \mu_{23}\hat{G}_{23}). \end{cases} \tag{3.21}$$

Based on this control objective, its control algorithms and system dynamics, we have concluded that this is a 2-input-2-output control system, whose control variables are φ_1 and φ_2, outputs x_{34} and x_{35}, and relative degrees (2, 2). As a result, we must use the first order derivatives of the outputs, x_{34} and x_{35}, to construct the sliding manifolds. Similar to the rate-based control system, we assume that system (3.21) is minimum phase. Using the same Lyapunov stability law, we propose the following

switch functions:

$$\begin{cases} s_1 = x_{34} + \omega_1 \dot{x}_{34} - \hat{x}_{34}, \\ s_2 = x_{35} + \omega_2 \dot{x}_{35} - \hat{x}_{35}, \end{cases} \tag{3.22}$$

where $\hat{x}_{34}$ and $\hat{x}_{35}$ are the target queue lengths. The sliding manifolds for this queue length control system are $s_1 = 0$ and $s_2 = 0$, and we select coefficients $\omega_1 > 0$ and $\omega_2 > 0$ so that the sliding manifolds are stable. For queue length control, it is not sufficient to make $s_1 = 0$ and $s_2 = 0$, since it does not guarantee that $y_{34} = x_{34} - \hat{x}_{34}$ and $y_{35} = x_{35} - \hat{x}_{35}$ would converge to zero. After some iterative derivations, we propose the following switching functions,

$$\begin{aligned} \dot{s}_1 &= \dot{x}_{34} + \omega_1 \ddot{x}_{34} \\ &= p_1 + q_{11}\varphi_1 + q_{12}\varphi_2, \end{aligned} \tag{3.23}$$

where the intermediate variables p_1, q_{11} and q_{12} are defined as follows.

$$\begin{aligned} p_1 \equiv\ & (1 - \omega_1 \mu_{34} \frac{\partial G_{34}}{\partial x_{34}}) \dot{x}_{34} \\ & + \omega_1 \alpha_{34} \mu_{13} \frac{\partial G_{13}}{\partial x_{13}} [-\mu_{13} G_{13} + \alpha_{13}(d_1 + \mu_{21} \hat{G}_{21})] \\ & + \omega_1 \alpha_{34} \mu_{23} \frac{\partial G_{23}}{\partial x_{23}} [-\mu_{23} G_{23} + \alpha_{23}(d_2 + \mu_{12} \hat{G}_{12})], \end{aligned} \tag{3.24}$$

$$q_{11} \equiv \omega_1 \alpha_{34} \mu_{13} \frac{\partial G_{13}}{\partial x_{13}} \alpha_{13} r_1, \tag{3.25}$$

$$q_{12} \equiv \omega_1 \alpha_{34} \mu_{23} \frac{\partial G_{23}}{\partial x_{23}} \alpha_{23} r_2. \tag{3.26}$$

Similarly,

$$\dot{s}_2 = \dot{x}_{35} + \omega_2 \ddot{x}_{35} = p_2 + q_{21}\varphi_1 + q_{22}\varphi_2, \tag{3.27}$$

where

$$\begin{aligned} p_2 \equiv & (1-\omega_2\mu_{35}\frac{\partial G_{35}}{\partial x_{35}})\dot{x}_{35} \\ & +\omega_2\alpha_{35}\mu_{13}\frac{\partial G_{13}}{\partial x_{13}}[-\mu_{13}G_{13}+\alpha_{13}(d_1+\mu_{21}\hat{G}_{21})] \\ & +\omega_2\alpha_{35}\mu_{23}\frac{\partial G_{23}}{\partial x_{23}}[-\mu_{23}G_{23}+\alpha_{23}(d_2+\mu_{12}\hat{G}_{12})], \end{aligned} \tag{3.28}$$

$$q_{21} \equiv \omega_2\alpha_{35}\mu_{13}\frac{\partial G_{13}}{\partial x_{13}}\alpha_{13}r_1, \tag{3.29}$$

$$q_{22} \equiv \omega_2\alpha_{35}\mu_{23}\frac{\partial G_{23}}{\partial x_{23}}\alpha_{23}r_2. \tag{3.30}$$

The control method is to adjust the values of φ_1 and φ_2 so that $s_i\dot{s}_i < 0$ (for $i = 1, 2$). To simplify the derivation, we define the following notation

$$l_i = \begin{cases} l_i^+, & l_i^+ + p_i > 0, \quad \text{when } s_i < 0, \\ l_i^-, & l_i^- + p_i < 0, \quad \text{when } s_i > 0, \end{cases} \tag{3.31}$$

for $i = 1, 2$.

We need to derive p_i and q_{ij} from on-line performance measures, which may not be readily available. To overcome this problem, we instead derive in (3.31) the range of l_i^+ and l_i^-, so that they can be used in the following equation to set the bounds of control changes. That is, the control law becomes

$$\begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}^{-1} \begin{bmatrix} l_1 \\ l_2 \end{bmatrix}. \tag{3.32}$$

Obviously, $s_i\dot{s}_i < 0$ holds for $i = 1, 2$. In fact, when $s_i < 0$, we have $\dot{s}_i = l_i^+ + p_i > 0$, and when $s_i > 0$, we have $\dot{s}_i = l_i^- + p_i > 0$. These control laws guarantee that the system states converge to $s_i = 0 (i = 1, 2)$, based on the Lyapunov stability law. Because the sliding manifolds $s_i = 0 (i = 1, 2)$ are stable (we select manually $\omega_1 > 0$ and $\omega_2 > 0$, x_{34} will converge to $\hat{x}_{34}$ and x_{35} to $\hat{x}_{35}$ under the above control law. Note

that it is necessary for the dynamic system (3.21) to be minimum phase to guarantee the system stability under the control law (3.22).

Chattering increases with system delays, because switching will not take place exactly on the switch surface. Significant chattering will cause the oscillation of the network traffic, making it a major hurdle to advancement of quality-guaranteed data networks. Continuous approximation [3] is a technique widely used to reduce chattering. The main idea of this technique is to replace abrupt control variable adjustment by slopes, or other similar functions, so that the chattering effects on the state space can be reduced. A main challenge is to make sure that the state trajectory follows the switch manifold as close as possible, with minimal chattering effects. Assume that it is difficult to measure system parameters precisely, and based on the continuous approximation method [3], we simplify the strategy (3.19) (3.20) as (shown in Fig. 19)

$$\varphi_1 = \begin{cases} -k_1 s + \nu_1, & |s| \le \delta_1, \\ 1, & s < -\delta_1, \\ 0, & s > \delta_1, \end{cases} \tag{3.33}$$

and

$$\varphi_2 = \begin{cases} -k_2 s + \nu_2, & |s| \le \delta_2, \\ 1, & s < -\delta_2, \\ 0, & s > \delta_2, \end{cases} \tag{3.34}$$

where $k_1, k_2 > 0$.

D. Implementation and Simulation

Our scheme can be readily implemented on existing routing devices without changing their communication protocols. We will focus on the ATM implementation here,

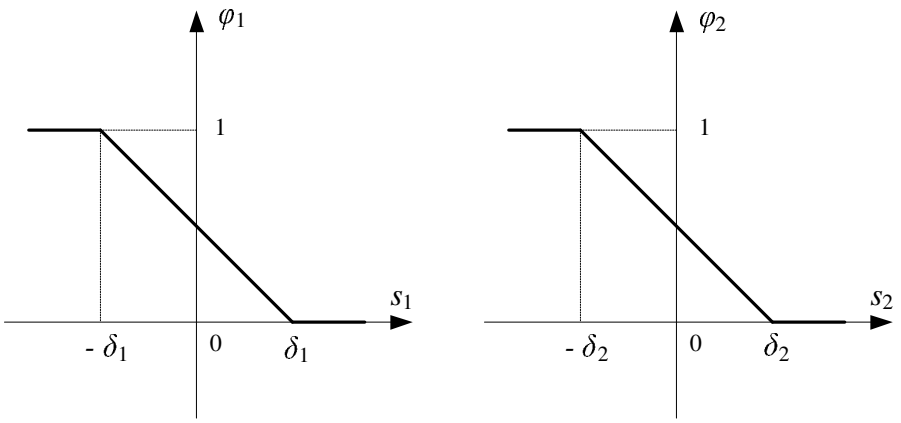

Fig. 19. A smooth, simplified control strategy to reduce chattering.

because of its popularity in wide area networks. ATM supports priority transmission. During normal operations, an ATM switch (may) block inflow of new packets with lower priorities, when it decides that its buffer would be full "soon". For backward pressure propagation control, a single backward pressure bit is needed for each channel. When this BP bit is set for the channel, all activities in the switch for cell transport of the channels are subject to the DDoS control rules that are set by the upper layer algorithms. One must have proper measures to guarantee that DDoS control packets can pass through clogged routing devices for execution of flow control decisions. This implies that the priority setting of internal control signals for channel arbitration will also need to be adjusted, in order to enforce preemption rules between channels. This feature is particularly important to counter DDoS attacks that are aimed at saturating the entire transport channels.

For rate enforcement, the leaky bucket algorithm can be readily applied. Here, CBR and VBR connections can be policed for Peak Cell Rate using one and two buckets, respectively. Each of the connections can be guaranteed to be within three percent of granularity for bandwidth allocation, ranging from 64Kbps to 622Mbps in commercial products. ATM uses the RM cell to carry information such as Explicit

Cell Rate, Current Cell Rate, Queue Length for flow control. A leaky bucket is mainly controlled by three parameters: 1) splash, the amount of fluid added to the bucket when a cell is taken into the network, 2) leak, the leak rate of the bucket, and 3) Blimit, the maximum bucket size. A typical leaky bucket algorithm can be summarized as follows, where "BLevel" stands for "Bucket Level" and "MOLT" stands for "Moment Of Last Transfer".

/* Calculate actual Bucket Level*/

NewBLevel := BLevel - (SCount - MOLT) * Leak + Splash

/* Bucket is empty if calculated level is < 0 */

if NewBLevel < 0 then NewBLevel := 0

/* Decide contract conformance */

if NewBLevel < BLimit then

Decision := Conforming

else

Decision := Non-Conforming

endif

/* Update BLevel and MOLT */

if Decision = Conforming then

Blevel := NewBLevel

MOLT := SCount

endif

DDoS defense can be modeled as a special case of the regular traffic policing problems in ATM transport (see Fig. 20). For normal operations, the traffic flow is examined for its conformance to the traffic contract, at arrival of each cell. The leaky bucket can be viewed as a virtual scheduler with continuous states, because it needs to be shared among different connections, and for its fine granularity in rate control.

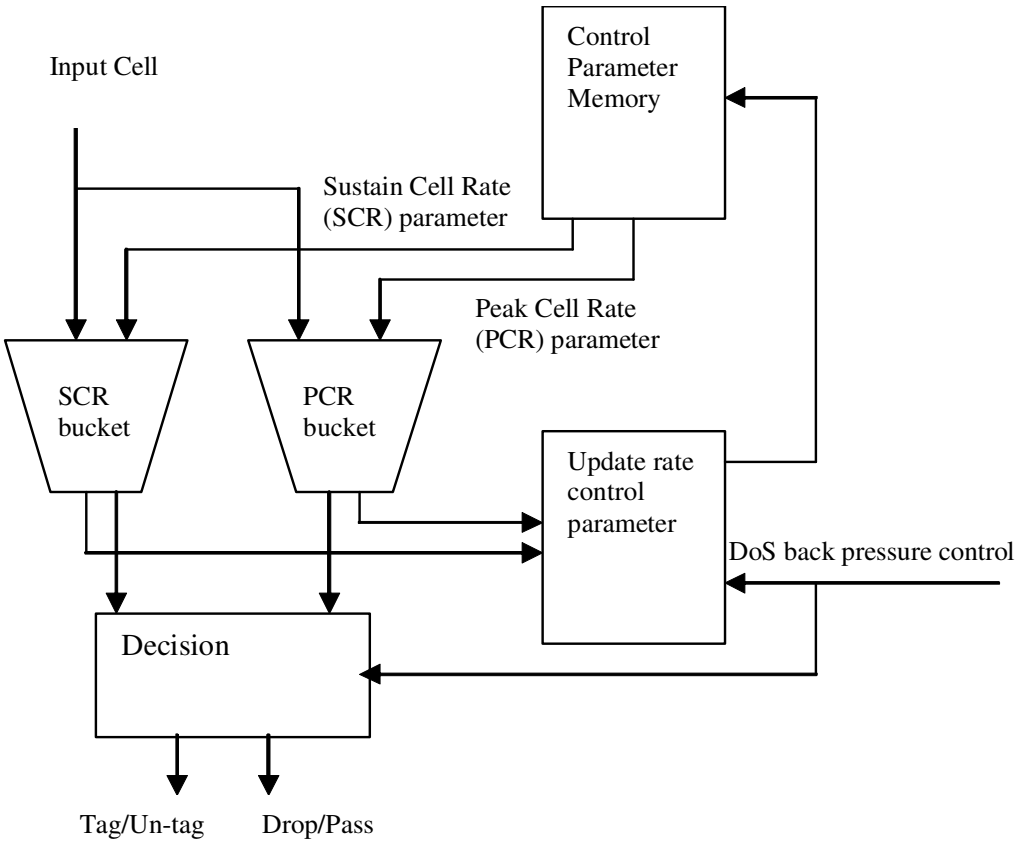

Fig. 20. Dual leaky bucket scheme for PCR and SCR conformation.

The leaky bucket needs to control the sustained constant rates (SCR) and the Burst Tolerance (BT). BT can be derived from SCR, peak constant rate (PCR), or other measures. When continuous smoothing is needed, one can easily use a counter to keep track of the incremental that the leaky bucket must take to adjust the permissible flow rate at each control instance. Control actions for DDoS defense can be readily accommodated by Usage Parameter Control (UPC) which is usually defined as the set of actions taken by the network to monitor traffic and enforce the traffic contract. The leaky bucket can either drop or tag offending cells of contract violators and DDoS offenders.

The total logic gate count to implement the leaky bucket, and the control block

for update rate control parameter is estimated at less than 100k, or roughly about one mm^2 of the silicon area using the contemporary 0.15μ CMOS technology. Control parameter memory stores the rate control parameters such as: leak, splash, and bucket limit for each flow. A total of 32 bits are needed for the rate control parameter entry to adjust the flow rate ranging from 2.5Gbps to 64Kbps. Each flow also has 32 sliding window that is controlled by the update rate control parameter block. If 32k $\times$ 32bit SRAM is implemented as on chip memory, the silicon area is estimated at $10mm^2$ in 0.15μ CMOS. This represents insignificant amount of VLSI area for most router design.

We evaluate the proposed algorithms using time driven simulations. Three interconnected nodes in a cluster work together to defend the flood of packets caused by a DDoS attack that attacked node 3. The total hot packet rate exceeds 1, the average cold packet rate is equal to 0.2, and the control interval is set at 10 clock cycles. The total buffer size is set at 60, and the target (average) queue length is set at 20, and the switching function s is defined using the queue-based algorithm, based on the queue length and its first derivative. Referring to Fig. 21 and Fig. 22, we compare the queue length between non-smoothing and smoothing control approaches. Referring to Fig. 23 and Fig. 24, we compare the switching function and throttling level between non-smoothing and smoothing control approaches. In Fig. 25 and Fig. 26, we compare the throughput between non-smoothing and smoothing control approaches. Before DDoS was engaged, we can see that the queues in the three nodes steadily grew until they became full. Then, when the DDoS defense kicked in, at clock time = 500, the queue length at node 3 steadily declined until it reached the approximated target queue length. Both nodes 1 and 2 did not have buildup, because the inflow control has been backward propagated to their inputs, according to the control rules. In this case, the queue length is also averaged for every ten cycles for readability. The

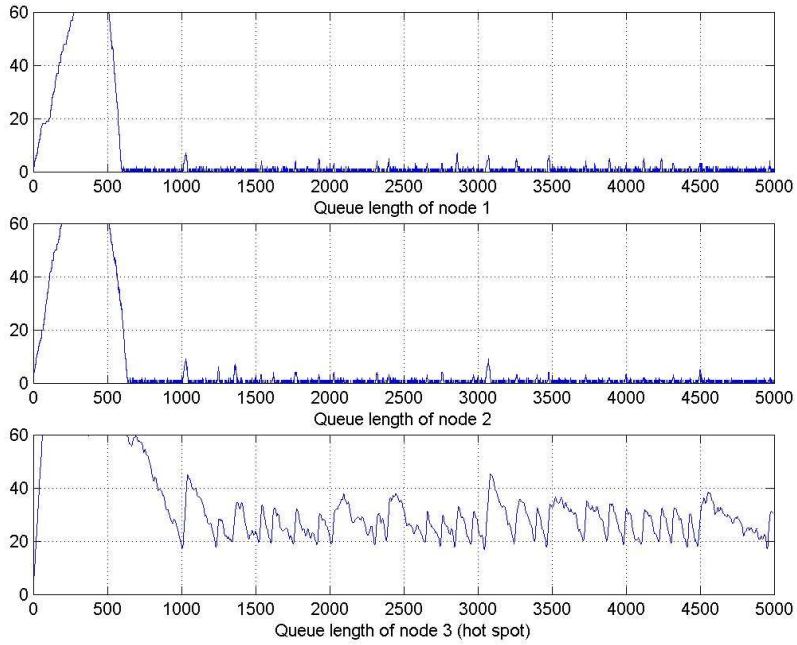

Fig. 21. Queue length with the non-smoothed control rule.

next issue of interest is the switching behavior. One can see that the values of the non-smoothed switching clearly oscillated much more than the smoothed one, and the throttling actions were much more bursty than its smoothed counterpart (at the right hand side column). Finally, let us examine the performance of the cold packet streams. In both cases, the cold packet streams were affected, because of the powerful hot packet storms. The cold packet streams have similar throughput levels, but the smoothed control scheme produced less bursty fluctuation, for obvious reasons.

E. Conclusion

This chapter gives a comprehensive treatment of the complex system dynamics related to DDoS defense, based on two different performance metrics, and control algorithms. We believe that the algorithms can be implemented in VLSI with negligible overheads, and no significant protocol change is necessary. Filtering techniques that can deter-

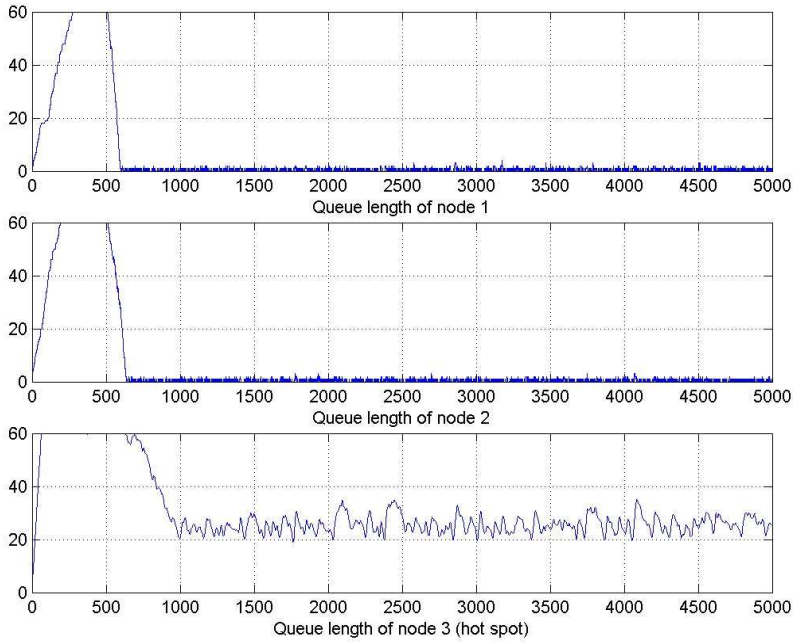

Fig. 22. Queue length with the smoothed control rule.

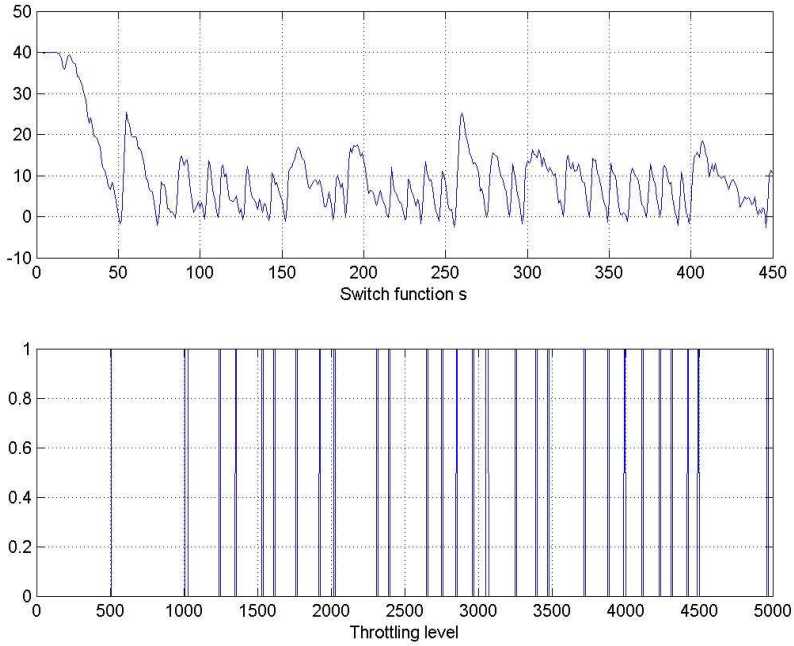

Fig. 23. Switching function and throttling level with the non-smoothed control rule.

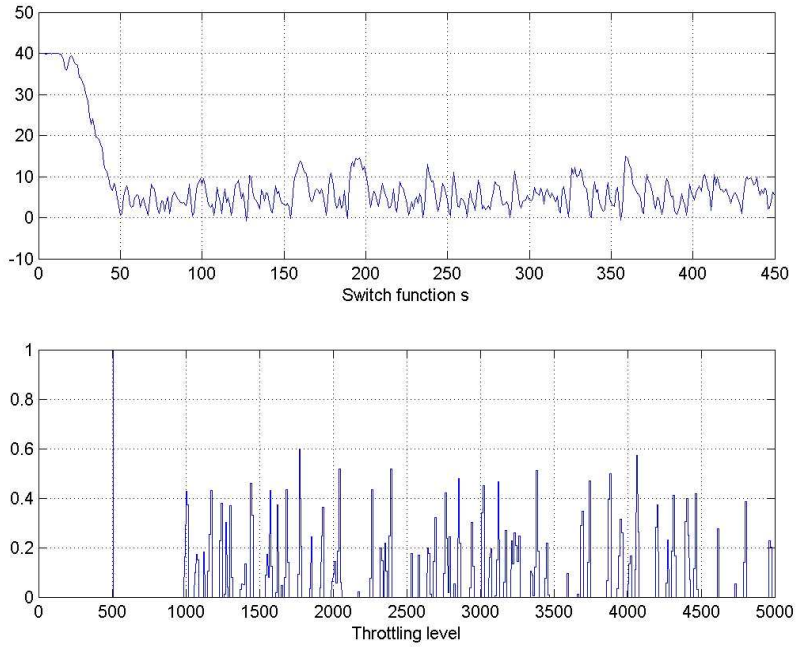

Fig. 24. Switching function and throttling level with the smoothed control rule.

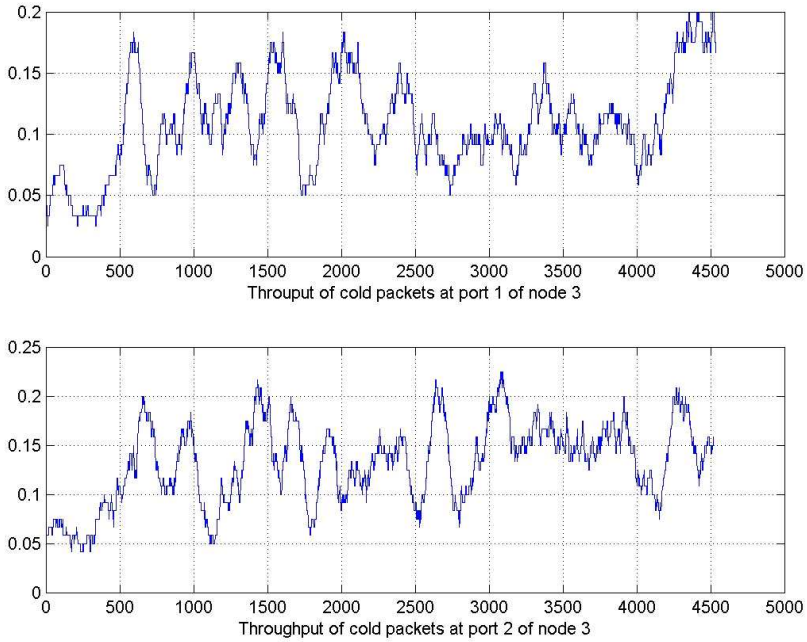

Fig. 25. Throughput with the non-smoothed control rule.

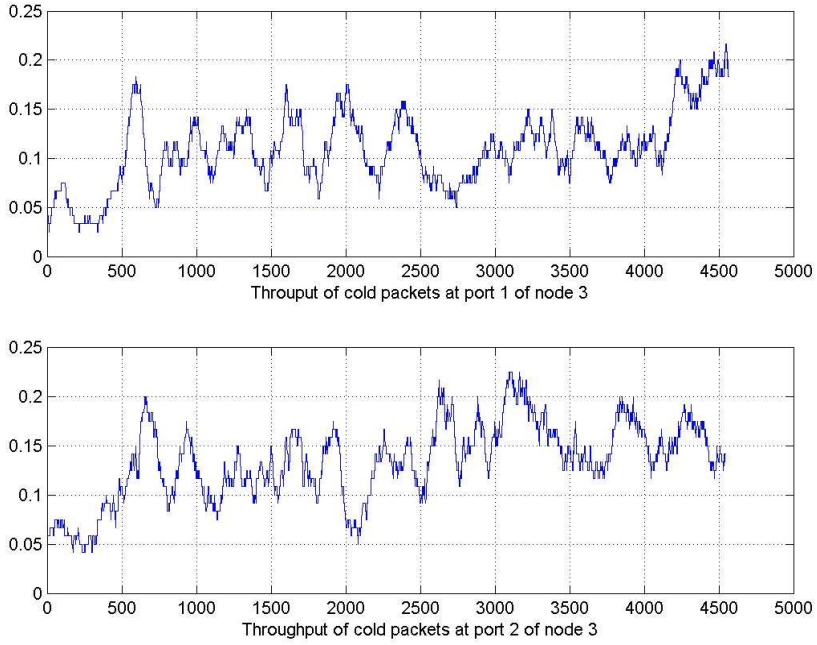

Fig. 26. Throughput with the smoothed control rule.

mine the predominating delay properties of a target network will be critical to the design of effective control algorithms. Primary factors related to oscillation include the time delay, (in)accurate representation of relative degree, noise (short bursts,) and characteristics (relative degree, minimal phase, etc.) of the network under control. Many existing flow control protocols are essentially some variations of the on-off control strategies without proper consideration of the relative degree they are susceptible to instability and breakdown.

CHAPTER IV

ANTI-EAVESDROPPING GROUP COMMUNICATION PROTOCOLS

In this chapter, we propose broadcast protocols and flow control algorithms to resist eavesdropping and traffic profiling of group communications. We adopt secret-sharing approach for exchange of shared key so that group members can use digital signatures to identify specific secret-sharing rule and determine their session keys independently. After the initiation phase to establish group memberships and exchange shared keys, group members exchange messages after they are fragmented, shuffled and encrypted into multicast packets. These packets are delivered along different paths to produce a concealment effect similar to that of the DC-net.

We consider breadth- and depth-first based packet delivery schemes. In the breadth-first scheme, packets are relayed in two stages across group members. For the depth-first approach, group members are organized into rings, each of which can be responsible for delivery of a particular type of packets. To resist traffic pattern profiling attacks, we develop a window-based adaptive traffic padding scheme for group members to balance their traffic flows. Both simulation and empirical results show that our scheme is highly efficient and robust to different traffic conditions. It has been tested on a highly secure network appliance machine to demonstrate its feasibility.

A. Introduction

Cryptographic protection can hide the contents of messages from unauthorized persons, but an eavesdropper can still eavesdrop the traffic pattern, without explicitly cracking encrypted messages, to extract some useful information, e.g. communication

peak times, sender-recipient pairs [67][68]. Such information can be critical for some applications, e.g. online bank transactions, battle field communication, etc. Traffic pattern concealment is an effective way to protect the traffic flow confidentiality [69][70][71][72]. In this chapter, we investigate sender-recipient anonymity in overlay group communications. Our ultimate goal is to hide the communication initiator among all group members so that only group members can determine the message senders.

A basic technique to counter traffic analysis attacks is traffic padding [69][70][71][72]: adding *decoy/dummy* packets to the traffic, so that the padded traffic pattern is made unrelated to the payload traffic. Raymond in [67] gives a survey on possible traffic analysis attacks on systems providing anonymous services and possible solutions. Baran [70] suggests using dummy or fraudulent traffic between fictitious users of the system to conceal traffic loading.

In [69] Chaum proposed a technique, in which users always deliver message through a proxy, called mix. The mix first collects a predefined number of fixed-size message packets from different users, and then shuffles the order of those packets and sends them out. In such a way, the sender identity is hidden among the user set. Considering that a mix cannot always get the predefined number of packets from the users, Chaum also suggests using dummy messages. In [73], Syverson etc. suggests using constant inter-arrival time (CIT) padding between the user and the proxy. CIT padding is also used in [67] to prevent packet counting attacks.

Knowing that dummy packets consume network bandwidth, it is highly desirable to use the minimal amounts of dummy packets to achieve concealment of the traffic patterns. In [71], Venkatraman and Newman-Wolfe proposed a linear programming method to optimize the bandwidth utilization while preventing traffic analysis. However, this optimization discloses the overall traffic and makes the system insecure.

Making the security mechanisms adaptive to the network conditions can reduce the performance costs, but the resulting traffic profile still needs to be made immune to statistical traffic analysis. In [74], Timmerman proposed an adaptive scheme for packet padding and rate control to reduce costs of traffic masking when the traffic workload changes. However, since dummy traffic rates are adjusted in (reverse) proportion to the real traffic, an adversary can deduce the real traffic loading through observing the padded traffic rate.

In this chapter, we propose a new technique for protection of group communications by concealment of sender-recipient pairs, and their traffic patterns. We assume that an existing key management scheme can be adopted for distribution and control of shared keys/secrets for secure data transport, and we focus on the sender-recipient anonymity, packet routing and adaptive traffic padding for a "well-known" group. Our basic approach is to fragment and disperse encrypted messages into multicast packets to be transported along different paths, creating an effect similar to that of the DC-net [75] so that the adversary cannot efficiently determine the source/recipient of a message without correct ordering of all packets, in addition to the needs for deciphering the messages. Packet flows among nodes are made balanced, to eliminate traffic patterns related to group activities. We proposed a distributed sliding window-based adaptive traffic padding scheme to control transmission of payload and dummy packets. In this scheme, we adjust the padded traffic rate instead of the decoy traffic rate to adapt to the changing of available network bandwidth. We proved that under this scheme, the padded traffic rate converges to the available bandwidth of the bottleneck link independent of the round trip time of the ring, and the feedback information does not disclose any information of sender/recipient activity. This is particularly important for an open network environment, where the users may not have full control of the network resources.

The remainder of the chapter is organized as follows: Section B describes the design of our anti-eavesdropping broadcast (AEB) protocol. Section C extends the design to provide adaptive traffic padding to improve bandwidth utilization. Section D and E provides a detailed performance evaluation and comparison. We conclude the chapter in Section F.

B. Anti-Eavesdropping Broadcasting (AEB) Protocols

In this section, we discuss our anti-eavesdropping broadcast (AEB) protocol to support dispersed transport of fragmented messages (after they are encrypted if necessary). AEB consists of two phases: initialization and operation. They exchange credentials and group keys before data exchange starts. When a legitimate new member needs to join the group, existing members must give it group keys to participate in group activities. When an active member departs, all the group keys of the group need be discarded, and the remaining members need to re-generate their group keys.

1. Design Goals

To provide sender-recipient traffic pattern anonymity for group communications, our work distinguishes itself from previous solutions, e.g. [75] by balancing of performance and anonymity requirements in an open networking environment. We assume that a public key infrastructure (PKI), e.g. X.509 or PGP, provides trust management and authentication, but the authentication authority does not necessarily possess the secrets for protected data exchanges. For group key distribution, we assume the use of a secret sharing scheme like that of Shamir [76] and Blakley [77], so that a recipient can recover the message when k-out-of-n of the shares or shadows become available. Of course, other authentication techniques like those in [78][79][80][81][82][83] are also

applicable. Nodes in the group but not involved in a transport session will only relay packets without deciphering the contents. We use a simple secret sharing based group key generation scheme, which was proposed in [76].

Given that the authentication and encryption issues are addressed, we next consider two issues related to sender-recipient anonymity: packet address headers, and packet transport patterns. It is impossible to conceal the existence of a "well known" group. However, it is relatively simple to use alias group ID headers to protect the real identities of group members. To prevent statistical analysis of the sender-recipient patterns, next we propose a dispersed packet transport scheme to conceal the group member interactions.

2. Dispersed Packet Transport

Group communication patterns can be divided into four major types: point-to-point $(1-1)$, point-to-multipoint $(1-N)$, multi-point-to-point $(N-1)$ and multipoint-to-multipoint broadcast $(N-N)$. To disassociate traffic patterns from group activities, the first obvious step is to use one single packet format to serve all types of data transport. To conceal the traffic pattern between a sender-recipient pair, their data exchange can be broken into fragments, each of which is diverted to one or more intermediate nodes before being delivered to the real destination. With sufficient amounts of fragments being emitted by group members one can generate highly symmetric packet flows, with help of padding traffic, which makes it very difficult to identify the sender-recipient pairs, not to mention how to decipher their messages. Packet fragmentation and shuffling is simple and efficient. Fig. 27 depicts the shuffling rule (3142) for mapping of message M. Let the shuffling outputs be denoted as V, we send packets in set V_i, for example, V_1 contains packets M_{33}, M_{31}, M_{34} and M_{32}, to node

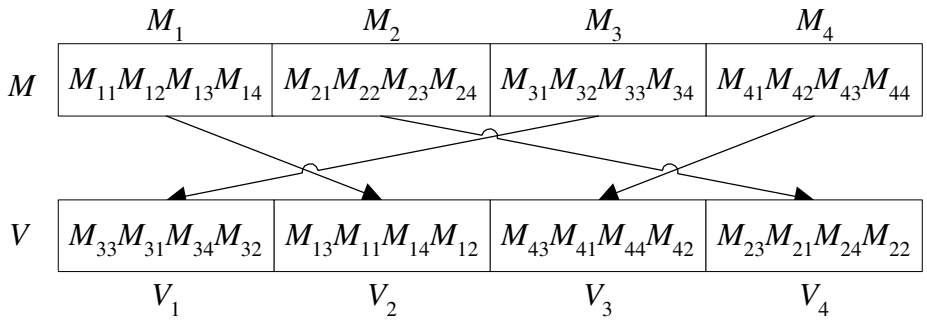

Fig. 27. An example on the double shuffling and slicing of packets ($p = 4$).

A_i respectively ($i = 1, 2, \cdots, n$). When a node S needs to broadcast a message M to n nodes, $A = \{A_1, A_2, \cdots, A_n\}$ in the group, S first sends out the digital signature of a secret-sharing rule so that only nodes in A can determine the session key. Other shuffling rules can be made a part of the shared key distribution system, so that only the chosen group members can decipher the shuffled messages.

3. Pattern-Free Packet Flows

Fragmented messages can be transmitted by breadth- and depth-first multicasts to create pattern-free packet flows. The multicast packets are routed via multiple nodes, but only nodes that have the proper session keys can decode the packets. In the breadth-first scheme, the sender fragments and disperses pieces of the message to intermediate nodes, which then relay these packets to the real destination(s). For the depth-first scheme, group nodes are organized into rings, so that data shares can be transported along them to reach destinations.

In the breadth-first scheme, broadcast traffic for the $1 - 1$ and $1 - N$ communications can be made symmetric by making the numbers of inbound and outbound messages identical at all nodes. All messages intended for specific destination are mapped into multicast packets to conceal the sender and recipient. $N - 1$ and $N - N$

are considered extension of these basic interaction patterns mentioned above. Passive eavesdroppers need to know the phase sequence, fragmentation and convergence process to decipher messages. Suppose that there are n nodes in the group, and the sender S uses i members for relay, the possibility of an eavesdropper knowing exactly the i group members is $(\sum_{i=1}^{n} \binom{n}{i})^{-1} = \frac{1}{2^n - 1}$. This simple equation does not consider the protecting effects of the dummy packets, nor that of packet ordering. Details of the breadth-first message processing routines are described below.

Node S fragments and disperses message M (through broadcast) to n participating nodes $A = \{A_1, A_2, \cdots, A_n\}$ (with a shared key K) and m intermediate relay nodes. All nodes relay message fragments in turn to other nodes to reassemble in the second phase.

Input: Message M and the shared key K.

Output: All n participating nodes with K able to decipher M.

Algorithm description at source S

o Calculate complete message encipher $EK(M)$.

o Fragment $EK(M)$ into p pieces, and do shuffling as shown in Fig. 27, $F(EK(M), n, m, p) = V_1, V_2, ..., V_p$, where $F(*)$ stands for fragmentation and shuffling operation.

o Choose an arbitrary next node A_j and choose a random V_i.

o Reliably transmit the encrypted fragment V_i to A_j. Noted as $S \rightarrow A_j : V_i$.

o Whenever the flow control scheme calls for sending of packets, send out the real or decoy packets according to the schedule.

Algorithm description at receiving nodes A_j

o For each V_i received, send an acknowledgement to the sender: $A_j \rightarrow S : ACK(V_i)$.

o For relay nodes, simply broadcast forward inbound packets, V_i or dummy packets to its real recipient.

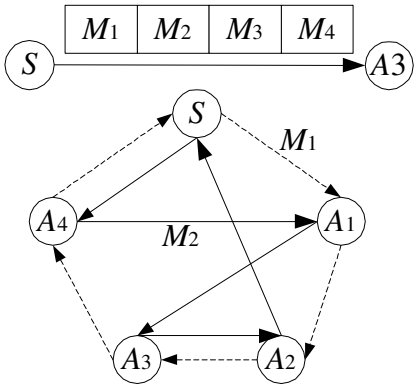

Fig. 28. Illustration of depth-first permutation ring.

o For the recipient, if the number of fragments received matches the predetermined size (as distributed by original source), an attempt is made to reorder and decrypt the fragments using the shared group key: $DK(V_1 + V_2 + \cdots + V_p)$.

The basic idea of the depth-first scheme is to organize group members into multiple overlay rings, along which packets carrying message fragments can be multicast or broadcast. In the example depicted in Fig. 28, node S intends to send a message to A_3. The message is broken into four fragments M_1, M_2, M_3, and M_4. The four encrypted fragments can be sent along four different overlay rings to reach A_3. It is easy to show that all the $1-1$, $1-N$, $N-1$ and $N-N$ interactions can be implemented using the overlay rings.

Labelling of the overlay rings among a group of nodes is equivalent to the permutation problem. That is, taking an arbitrary node as the starting point of node traversing, each overlay ring represents a unique permutation of the native node labels, and thus a unique ring ID. As a result, we adopt the notation of permutation for ring representation, called *a permutation ring*, which represents a logical interconnection path between group members. When S needs to send a message to node

A_i, it transmits each of the message fragments in a packet, along a specifically chosen ring to reach A_i. The number of rings increases at a factorial order resulting in a large search space for the eavesdroppers to crack the packets. Details of the depth-first message processing routines are described below.

Sender S disperses the packets carrying fragmented message pieces along chosen rings to reach nodes $A = A_1, A_2, \cdots, A_n$ through m intermediate nodes.

Input: Message M that the source wants to send, and a shared group key K.

Output: All n participating nodes with K able to decipher original message M.

Algorithm description at source S

o Calculate complete message encipher $EK(M)$.

o Fragment the enciphered message $EK(M)$ into p pieces, and do shuffling as shown in Fig. 27, $F(EK(M), n, m, p) = V_1, V_2, \cdots, V_p$, where $F(*)$ stands for fragmentation and shuffling operation.

o Choose a permutation pattern $\Pi_j = \pi_{j,1}, \pi_{j,2}, \cdots, \pi_{j,n}$ for $V_j, \forall j$. Calculate a ring identifier digest that uniquely identifies the hop sequence $R_j = H_R(\Pi_j), \forall j$.

o $S \rightarrow A(\pi_j, 1) : R_j + V_j, j = 1, 2, \cdots, n$. S reliably unicasts V_j along j.

o S receives V_j from the last hop of j, acknowledging the successful circulation for V_j transmitted across j.

o Send out dummy packets along different rings in accordance to the flow control scheme.

Algorithm description at receiving nodes A_i (with permutation ring identifier j)

o Use the ring digest R_j of the received packet to calculate the next hop on j.

o Unicast the packet R_j+V_i to the next hop entry $A(\pi_j, i{+}1) : A(\pi_j, i)A(\pi_j, i{+}1) : R_j + V_j$.

o For a recipient in A, order and decrypt the fragments using the shared group

key: $DK(V_1 + V_2 + \cdots + V_n)$ using the shared group key K.

Permutation rings have relatively higher latency costs because each packet needs to traverse a ring before it is removed from the network. But it has similar buffer and ordering overheads for the sender and recipients. Rings have simpler error recovery and flow control schemes to conceal the communicating parties. An adversary needs to know the rings and sequences of packet transmissions. Without having knowledge of the nodes and the rings involved, the eavesdropper search complexity grows at a factorial order. Taking the message-ordering complexity into consideration, the overall probability of message cracking becomes $\frac{1}{p![(n-1)!]^p}$.

C. Adaptive Packet Padding

In last section we discuss how to disguise group interactions via dispersed relay of packets. In this section, we further discuss techniques on how to conceal the traffic volumes, transmission rates, and their relationship with group activities. Packet padding is effective in concealing payload traffic patterns. One can easily create pattern-free (or more precisely, uni-pattern) traffic profile on a data link by using a polling technique, in which packets are transmitted at constant time intervals. Otherwise, it is easy to identify the sender-receiver pair by comparing their inbound and outbound rates. Even if all packets traverse the whole ring, without traffic padding, the outbound traffic of a source node has a large phase lag with respect to its inbound traffic. However, simple traffic padding, e.g. fully utilizing the link capacity, can be cost-prohibitive, due to excessive waste of bandwidth.

The objective of packet padding is to drive nodes on an overlay ring to reach the same traffic sending rate, so that their transmission patterns would appear to be (nearly) identical to adversaries. Equalizing the outbound rate with the inbound

rate is effective but not sufficient, because it cannot effectively handle the condition when the inbound traffic rate is higher than the available bandwidth of the outbound link. For group members on a ring to match each other's transmission rate, it is obvious that the link that has the smallest available bandwidth will determine the overall transmission rate on the ring. Sending rates of other group members should be adjusted to match the available bandwidth of the bottleneck link.

1. Distributed Sliding-Window for Packet Padding

To minimize bandwidth loss, and to balance the traffic flows between group members, we propose a distributed sliding-window-based packet padding scheme for nodes on the overlay ring. Our scheme does not require explicit coordination messages between group members. Theoretical models and simulations show that this scheme is adaptive to bandwidth variation, and greatly reduces the bandwidth waste, without compromising concealment of the traffic patterns.

The sliding-window based flow control protocol is successfully deployed to the TCP protocols [21]. A TCP sender maintains a congestion window, whose size defines the largest number of outstanding packets that are allowed to be in transit. From the flow control viewpoint, a TCP session can be viewed as a logical ring. The payload packets and ACK packets are transmitted on the forward and backward paths of a TCP session respectively, while for an overlay ring there is no obvious difference between forward and backward paths (see Fig. 29). A TCP session has an obvious sender and receiver while for a ring all nodes are virtually identical.

In summary, we can view an overlay ring as one single transmission session, and each of the intermediate overlay nodes is equivalent to an intermediate router on the forward path. We adopt a sliding-window-based scheme to throttle the traffic on the

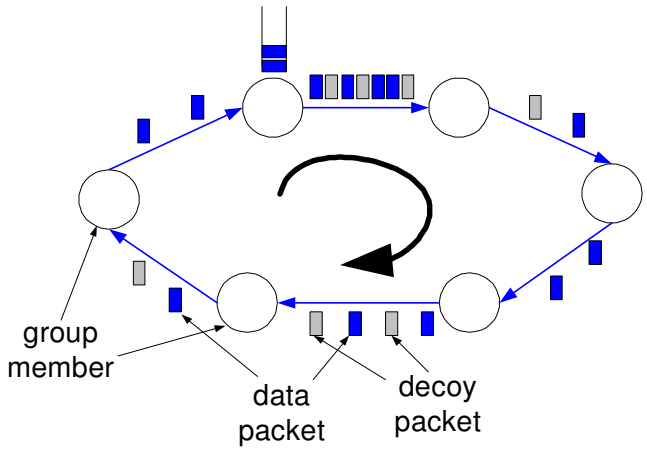

Fig. 29. An overlay ring vs. a TCP session.

overlay ring, where the congestion window of a TCP session is maintained by the TCP sender. When every node on a ring runs the sliding-window-based flow control rules, any local changes will lead to local flow adjustments, and such adjustments will lead to neighboring and global adjustments, without using explicit control messages. In this way, not only nodes keep their transmission rates identical, but also adaptive to changes of network conditions. Next we discuss the details of our algorithm.

Let us assume that there are N group members on the overlay ring, and all nodes have the same buffer size. Each node has a *sending queue* to store (local, transit and dummy) packets ready for transmission. First, each node estimates the available bandwidth of the bottleneck link and round trip transmit latency of the ring, denoted as $\bar{C}_i$ and $\bar{d}_i$ respectively, using some method developed in existing literatures, e.g. [84][85][86][87][88][89]. Here the bandwidth and delay estimations need not be very accurate. Then the node sets its initial window size as

$$W_i^0 = \frac{\bar{C}_i \bar{d}_i + Q/2}{N} \tag{4.1}$$

where Q is the buffer size. The physical meaning of (4.1) is obvious-the optimal number of packets along the ring is $\bar{C}_i \bar{d}_i + Q/2$ when the desired queue length of the bottleneck link is $Q/2$ (see page 412 in [90]). The inaccuracy of the initial windows caused by the bandwidth and delay estimation errors can be corrected by window adjustment that will be discussed in subsection 3 in section C.

Sliding window scheme for overlay rings

o When the session begins, node i sends out W_i^0 packets. The flow control scheme determines when to send out the next packet.

o When a node receives a packet m, it

1. increases its window size by one;

2. checks the packet header to find whether this packet is a decoy packet. If so, ignore it.

3. checks the packet header to find whether this packet is sent by itself earlier. If so, ignore it.

4. If m is not sent by itself earlier and m is not a decoy packet, then put packet m at the end of the sending queue.

o Only when the window size is larger than zero, the node can send out a packet.

o When sending a packet, the node sends a decoy packet to the next node if its sending queue is empty; else fetch one packet from the head of sending queue and send it to the next node.

o When an upper layer application sends a packet, the node puts the packet at the end of the sending queue.

This sliding window scheme is static because the number of packets in the pipeline is fixed. The window is different from the congestion window in TCP. In TCP, the congestion window size determines the maximum outstanding packet number, while here the window size defines the number of packets the node can send out at current time. Moreover, in TCP, congestion window is maintained in the TCP sender. In an overlay ring, there is no obvious sender. The window is distributed in all ring nodes. Next we analyze its adaptability.

2. Adaptability Analysis

An important property of the sliding window flow control protocol is self-clocking– For a connection in equilibrium, a new packet is not put into the network until an old packet leaves [21]. Packets circulating around the ring have the same round-trip/circulation time, R. Within the time period of R, nodes on the ring send out

$\sum_{i=1}^{N} W_i^0$ packets. Using the fluid dynamics model, the queuing dynamics for a sending queue can be expressed as:

$$\dot{q}(t) = \frac{\sum_{i=1}^{N} W_i^0}{R(t)} - C, \tag{4.2}$$

where

$$R(t) = d + \frac{q(t)}{C}, \tag{4.3}$$

where d is the real round trip transmit latency of the ring, $q(t)$ is the queue length of the congested link at time t, and $\dot{q}$ the first time derivative of q. The equilibrium state of (4.2) is

$$q^* = \sum_{i=1}^{N} W_i^0 - Cd. \tag{4.4}$$

If the bandwidth and delay estimations are accurate, that is $\bar{C}_i = C$ and $\bar{d}_i = d$, (4.4) means $q^* = Q/2$.

By choosing the Lyapunov function

$$v = \frac{1}{2}\dot{q}^2, \tag{4.5}$$

we have

$$\dot{v} = \dot{q}\ddot{q} = -\dot{q}\frac{C(\bar{C}d + Q/2)}{(Cd+q)^2}\dot{q} < 0, \tag{4.6}$$

except at $\dot{q} = 0$. Therefore, (4.2) asymptotically converges to its equilibrium point defined by (4.4). Thus, the sending rates of all nodes converge to the congested link capacity C. The convergence time is the round trip time R because of the self-clocking property. Even if the congested link shifts from one to another link, the sending rates of all nodes will converge to the new congested link capacity. ns-2 simulation results given in section E support this statement.

We further note that the equilibrium point of the queuing dynamics is defined by (4.2) only if $0 \leq \sum_{i=1}^{N} W_i^0 - Cd \leq Q$. If $\sum_{i=1}^{N} W_i^0 - Cd < 0$, queue starvation (low link

utilization) will occur. On the other hand, when $\sum_{i=1}^{N} W_i^0 - Cd > Q$, queue overflow (packet dropping) will occur. That is, window sizes should be adjusted according to the available bandwidth, which fluctuates with external traffic.

3. Window Adjustment

The objective of window adjustment is to keep the queue length of the bottleneck link within a reasonable range, so that the traffic flows on all nodes of the overlay ring can remain stable and balanced. We first study the queuing dynamics of a ring with sliding window-based flow control scheme. Within one control period Δt, node i gets m_i packets and increases its window size by ΔW_i(positive or negative), so that it sends out $m_i + \Delta W_i$ new packets within time period Δt. Thus, we have

$$\Delta q(t) = \sum_{i=1}^{N} [m_i(t - \tau_{Fi}) + \Delta W_i(t - \tau_{Fi})] - C\Delta t, \tag{4.7}$$

where τ_{Fi} is the forward transmission delay from node i to the bottleneck link, and N is the node number of the ring. By the self-clocking property we have

$$m_i(t) = C\Delta t, \tag{4.8}$$

where τ_{Bi} is the transmission delay from the bottleneck link to node i. Note that $\tau_{Bi} + \tau_{Fi} = R$, where R is the round trip time of the ring and R changes with time. Substituting (4.10) into (4.9), we get

$$\Delta q(t) = \sum_{i=1}^{N} \Delta W_i(t - \tau_{Fi}), \tag{4.9}$$

We choose the control period $\Delta t = R$ and adopt the following proportional feedback control rule,

$$\Delta W_i(t) = \frac{k}{N}[q_d - q(t - \tau_{Bi})], \tag{4.10}$$

where $0 < k < 1$. k is often set as 0.5 in order to balance the stability robustness and convergence rate. Under this control law, the closed loop system dynamics can be written as

$$q(t) - q(t - R) = k[q_d - q(t - R)]. \tag{4.11}$$

From (4.11), we have

$$q(t) - q_d = (1 - k)[q(t - R) - q_d]. \tag{4.12}$$

It is obvious that $q(t)$ asymptotically converges to q_d because $0 < k - 1 < 1$. This means that our window adjustment scheme is asymptotically stable even with large, time-variant delays, even when the round trip time R changes with time. The queuing dynamics under the control law (4.12) is stable.

D. Evaluation

We evaluated our models and algorithms based on simulations and implementations. For implementation, we tested the feasibility of fragmented packet transport on top of existing IP protocols. For simulations, we examined the robustness of the proposed bandwidth control schemes.

1. Implementation

We used four interconnected machines on a LAN to test the feasibility of proposed scheme using the UDP protocol. Due to resource constraints, we only tested the depth-first scheme. Here, group members on the overlay ring exchange data using UDP packets, while the inter-node packet transmissions are controlled by a sliding-window-based control algorithm in each node. Real and decoy packets shared the transmission bandwidths.

The prototype AEB was implemented based on the total ordering algorithm proposed in Isis's ABCAST [91]. That is, if a node delivers message m before it delivers n, then any other correct process that delivers n will deliver m before n. We embedded the totally ordered ring protocol into the Emcast toolkit [92]. It includes the program "emcast", a generic multicast utility (like netcat), and the library "libemcast", a generic multicast library. Emcast supports IPv4 multicast (IM) and end-host multicast (EM) protocols, such as STAR (centralized TCP), Banana Tree Protocol (BTP), and Internet Chat Relay (IRC), plus our Ring protocol. In the experiment, only one node (Node 1) broadcasts messages and the other nodes receive messages. In the four charts of Fig. 30, one can see that the traffic of each node remains virtually identical. The real traffic pattern reaches uniform and symmetric in each node through AEB.

E. Simulation

It was much more complicated to test the control strategies on the prototypes. As an alternative, we used the ns2 network simulator [93] to evaluate our adaptive packet padding scheme. In the simulation, six nodes are interconnected to form an overlay ring, and the link between nodes 2 and 3, and the link between nodes 4 and 5 are shared by outside nodes, see Fig. 31. Starting from node 1, and in the ascending order of the links between nodes, the bandwidths of links are 5, 4, 10, 5, 10, and 10 Mbs. In all following experiments, the packet sizes were set at 1k bytes, and the window sizes were initialized as 10 pkts. The latencies of the links were set at 10 ms unless explicitly stated. We used the following term, "$link_{jk}$" means the link between node j and k. We call the transmission session among group members on the ring the *internal session*, and the transmission sessions among non-group members the *exter-*

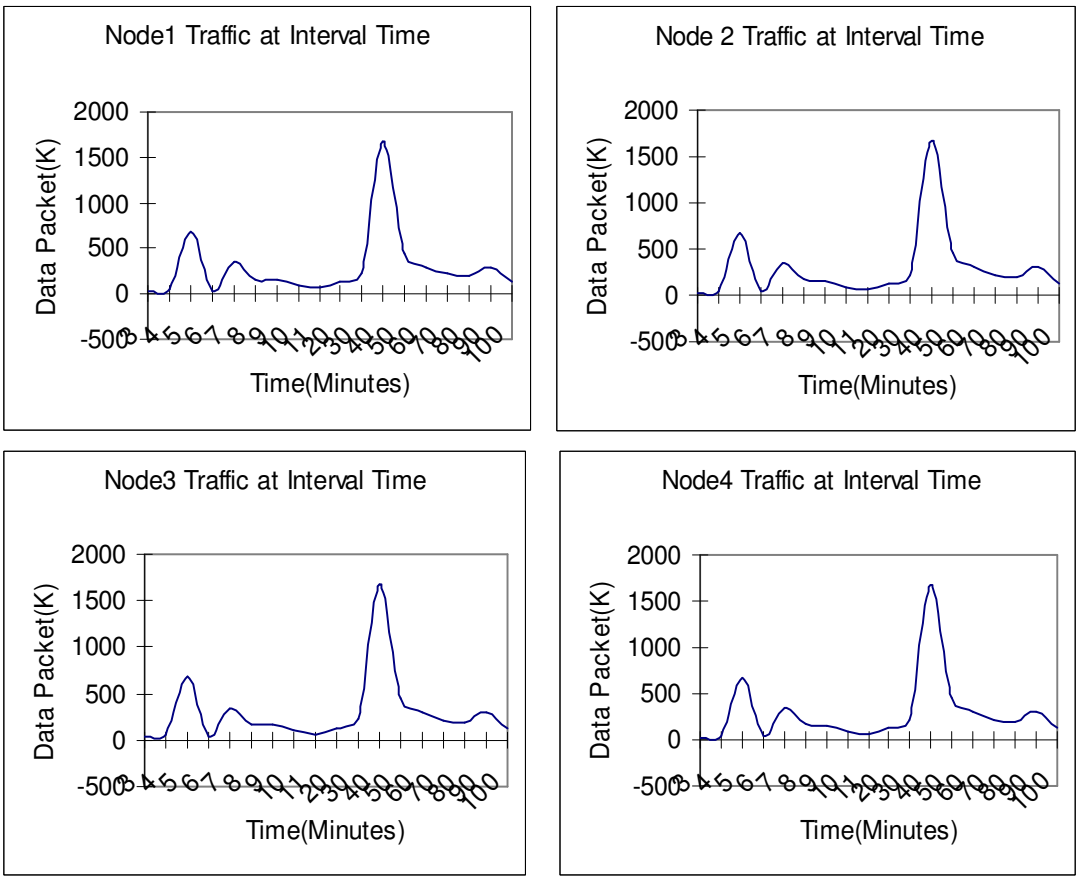

Fig. 30. Traffic volume statistics at each node.

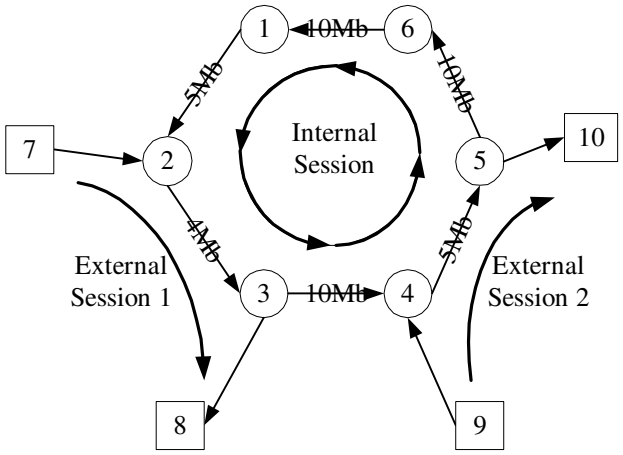

Fig. 31. Simulation topology.

nal sessions. In the simulation, we created an application-level transport protocol on the top of UDP. When the overlay ring runs on a closed environment, without the interference of external traffic, our sliding window control scheme converged rapidly and remains steady henceforth. To further test the robustness and adaptability of our scheme against uncontrolled external traffic, we created two different traffic scenarios to examine their impact on the traffic profiles on the ring.

1. Experiment 1: Workload pattern concealment

In the first experiment, we tested the effects of workload traffic changes on the overall (padded) traffic rate. We assume there is no eternal session in this experiment, so that effects of workload changes can be best observed. Between the 2nd and 10th second, there is 8Kbps workload traffic from node 1 to node 4, and between the 7th and 17th second, there is workload traffic with rate 8Kbps from node 3 to node 5.

Fig. 32 shows the measured workload traffic rate, decoy traffic rate and padded traffic rate on each link. It shows that the decoy traffic rates are automatically adjusted to make the overall padded traffic rates match the lowest available bandwidth on the ring, 4Mbps. It clearly suggests that the workload traffic changing have virtually no effects on the overall padded traffic rates.

2. Experiment 2: Performance of the static sliding window scheme

The second test was meant for simulation of static (non-adaptive) sliding window scheme. The two external sessions, sessions 1 and 2, were created to simulate a constant bit-rate (CBR) session. For the CBR transport, the time interval of packet transport is set at 0.006s. At time 0 (sec), the group session starts. Then, at time epochs 4 and 8 secs, the external session 2 between nodes 9-10, and the external session

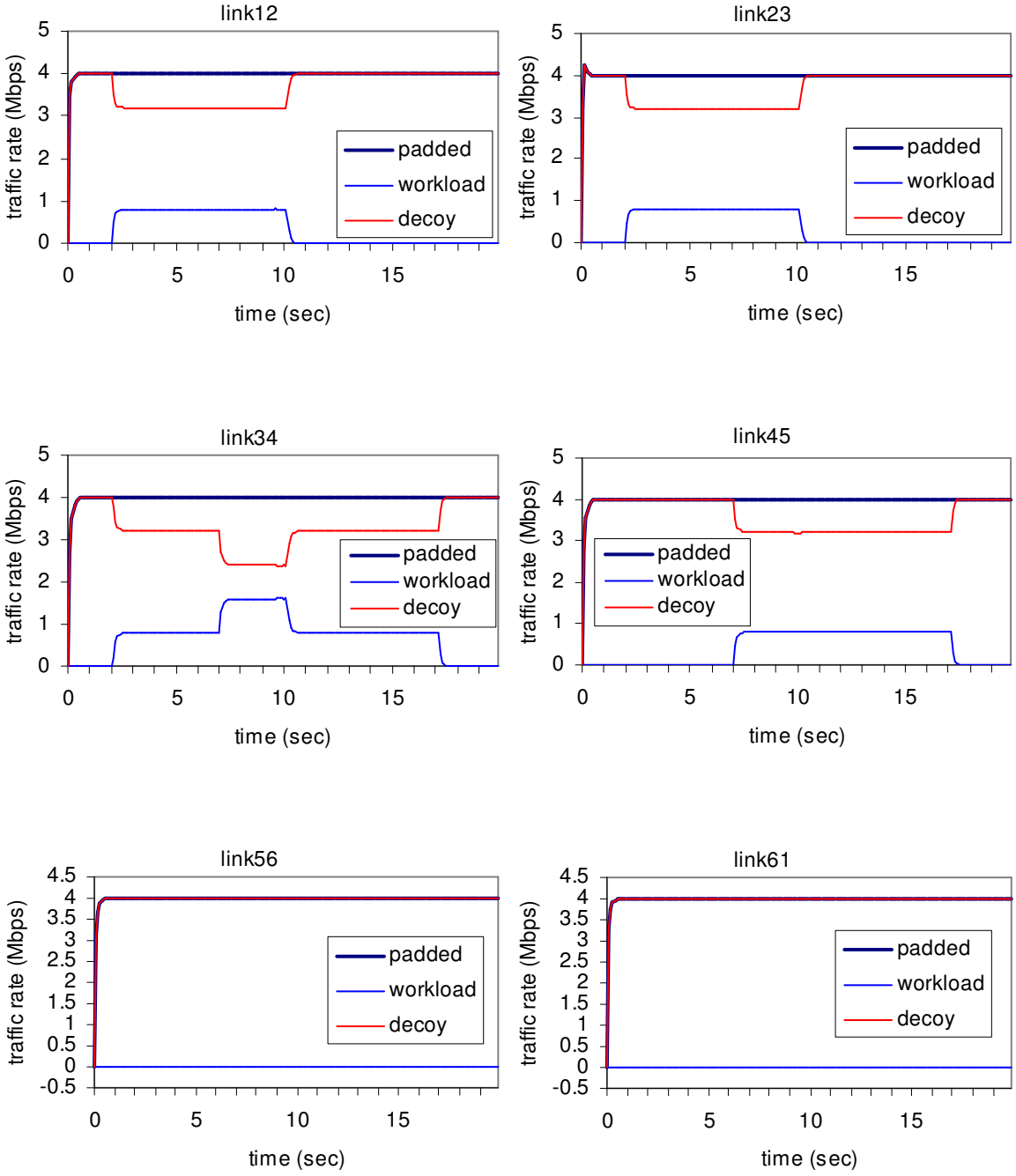

Fig. 32. Traffic rates on each link in experiment 1.

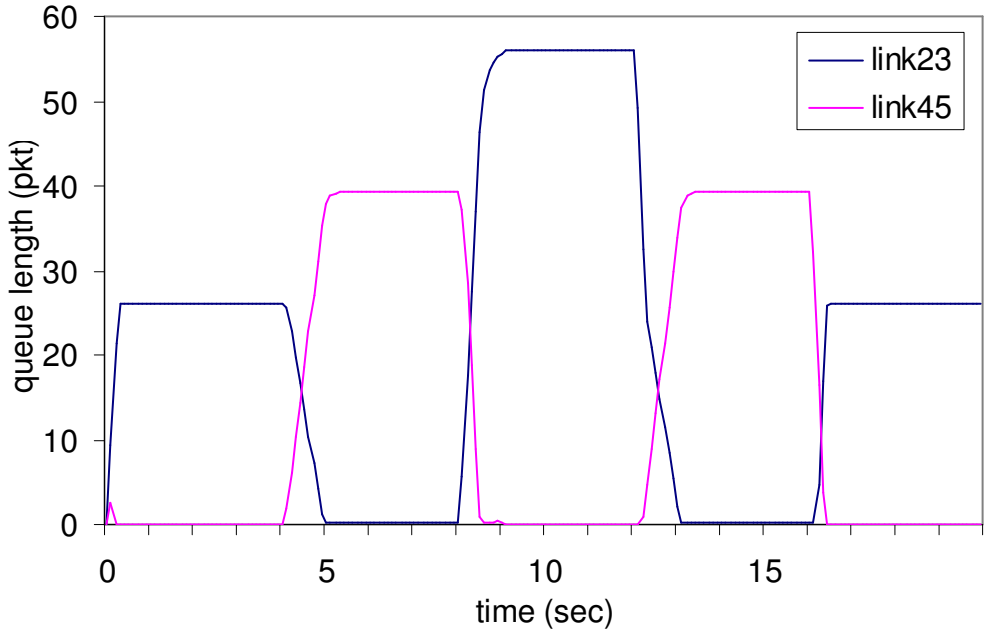

Fig. 33. Congestion queue dynamics in experiment 2.

1 between nodes 7-8, started, respectively. At time epochs 12 and 16, the external sessions 1 and 2 ended, respectively. Finally, the group internal session ended at time epoch 20 sec. Fig. 33 shows changes of queue lengths in link23 and link45 of the ring. The queue lengths in other nodes remained zero for the whole simulation period, so they are not shown here. We note that even though such changes are inevitable at presence of external sessions, it is clear that shortly after a session starts, or ends, the queue lengths converge to their steady values rapidly. Fig. 34 plots the transmission rates of group nodes on the ring for the experiment run. Fig. 35 shows the measured work traffic rate on each link. In all cases, except for the short transmission rate disturbance during state transition periods, the differences in the transmission rates of the group members are very small, on average less than 5 percent. Note that *the disturbance to transmission rates is caused by external events, it is not associated with the group activities.*

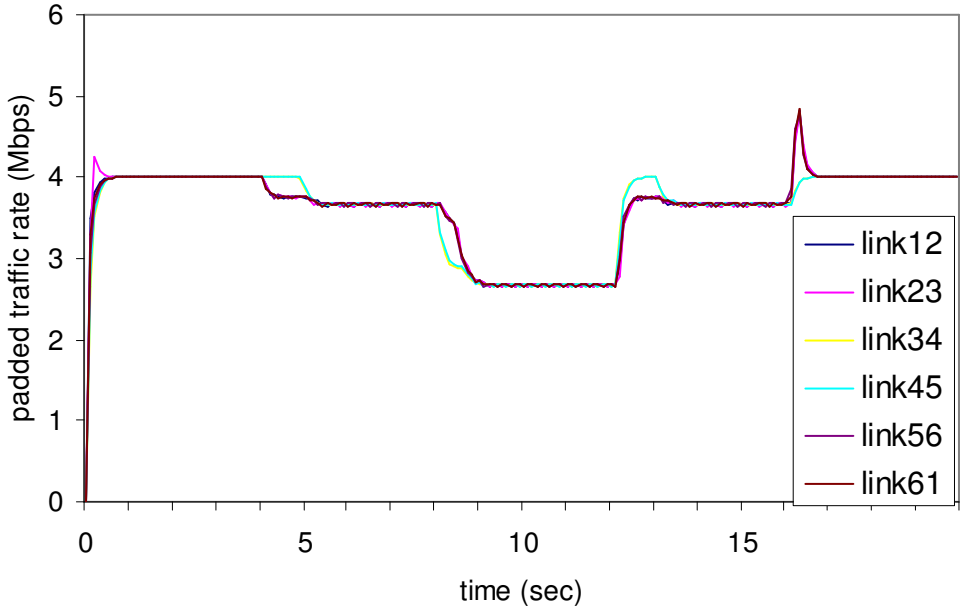

Fig. 34. Padded traffic rate at each link in experiment 2.

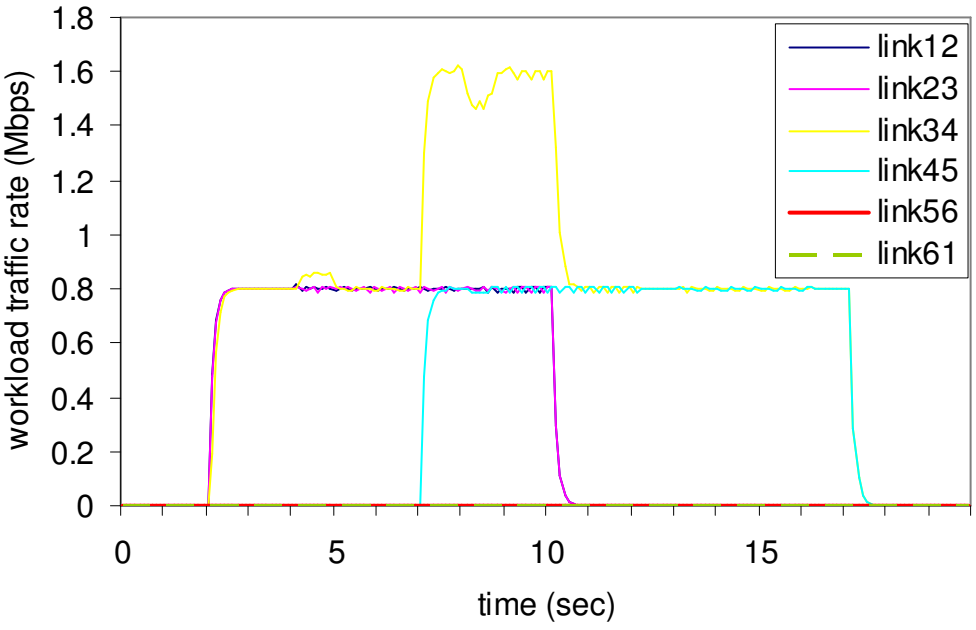

Fig. 35. Workload traffic rate on each link in experiment 2 (with eternal sessions).

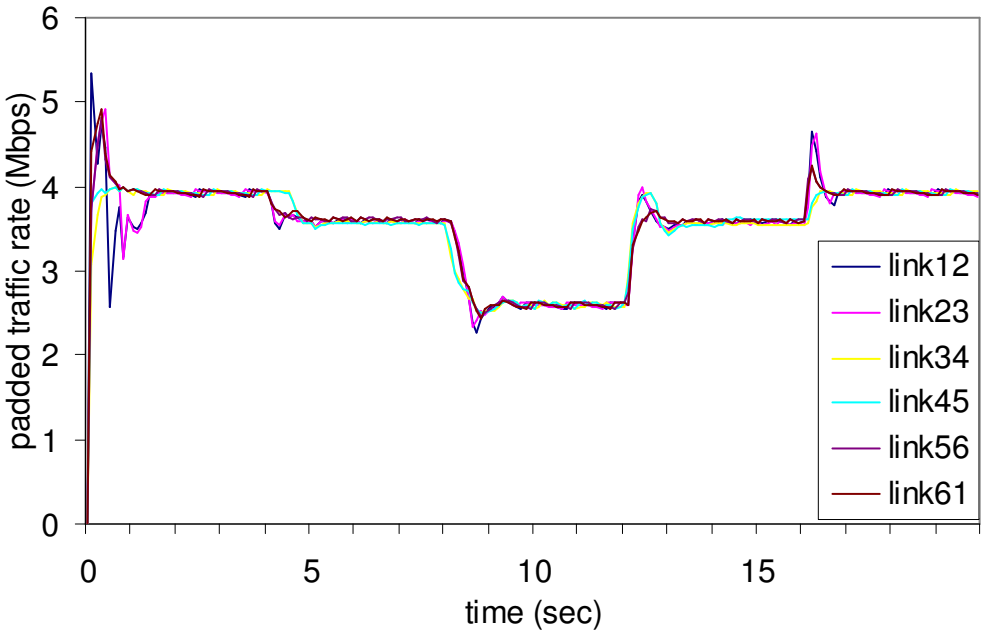

Fig. 36. Padded traffic rate on each link in experiment 3.

3. Experiment 3: Performance of the adaptive sliding window scheme

The third experiment is designed to test the effectiveness of our window adjustment scheme. All settings are the same as in experiment 2 except that our window adjustment scheme is enabled. Fig. 36 shows that the overall rates at all links are almost identical to each other. The minor rate differences are also caused by outside events, external sessions. Fig. 37 shows that the queue length of the congested link quickly converges to the proximity of the desired value, 20 packets, even if congestion shifts from link23 to link45 at the 4th and 12th sec and shifts back at the 8th and 16th sec. Thus, the link under-utilization and queue overflow are avoided.

4. Experiment 4: Performance impact of delays

The fourth experiment is used to test the effectiveness of our window adjustment scheme for the case with large round trip delay. All settings are the same as in exper-

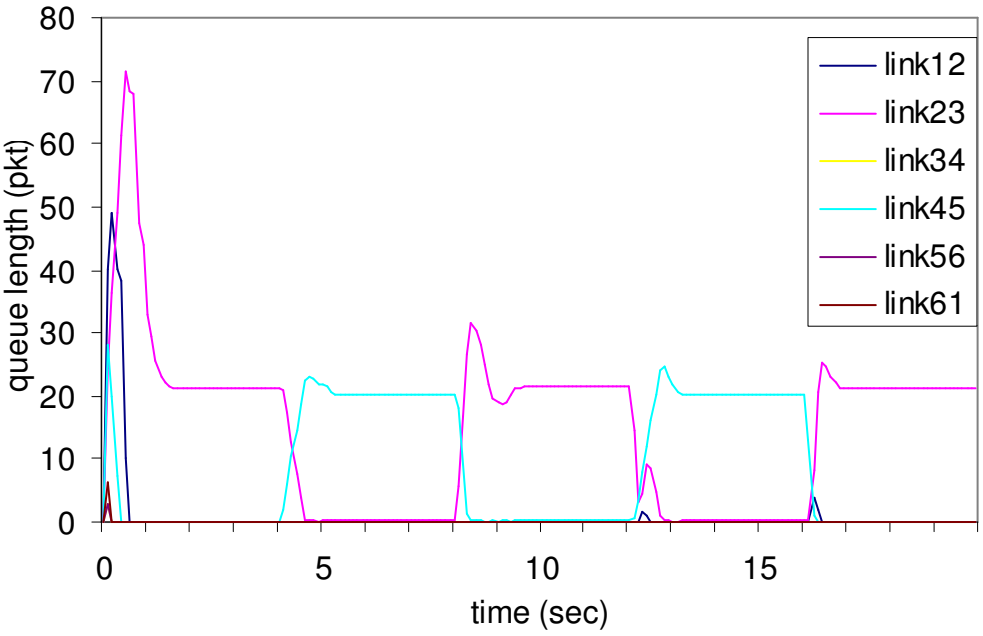

Fig. 37. Queue length of each link in experiment 3.

iment 3 except that the link transmission delays for the links are set at 50ms, 50ms, 70ms, 50ms, 40ms and 70ms respectively. Thus, the round trip delay is about 0.33 seconds. Fig. 38 and Fig. 39 clearly show the effectiveness our window adjustment scheme even with such a large round trip delay.

5. Experiment 5: Robustness of the adaptive sliding window scheme

In this experiment, we significantly increased the level of external traffic to test the robustness of the sliding window flow control algorithm. In this case, the external sessions recurrently start and close ever two seconds, starting from 0.5 sec. The composite set of starts and closes of the external sessions are collectively called the event Z in Table I.

Similar to experiment one, the transmission rates of group nodes converge to the steady state value rapidly at absence of external sessions. Fig. 40 shows that the

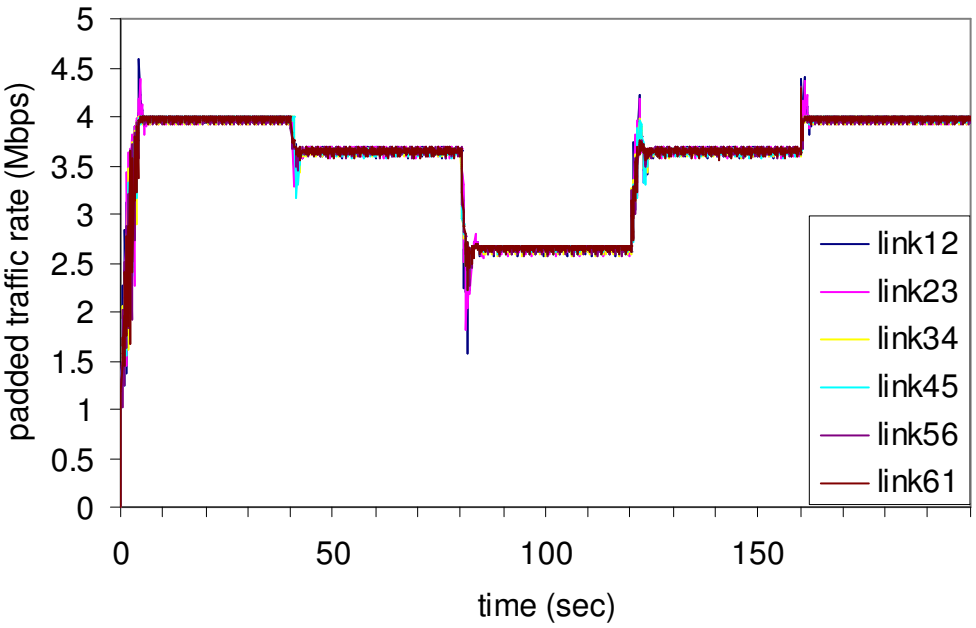

Fig. 38. Padded traffic rate on each link in experiment 4.

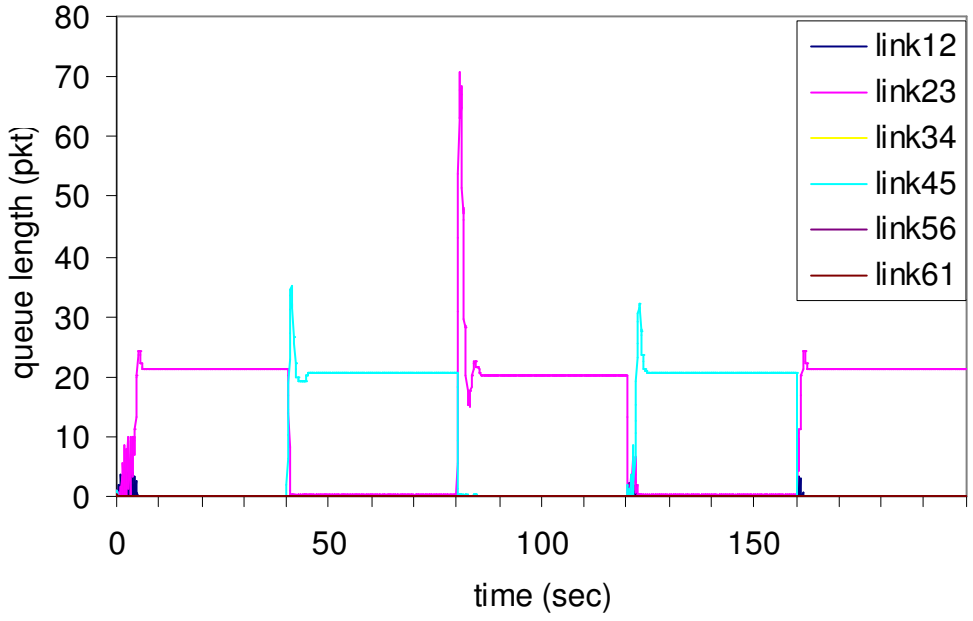

Fig. 39. Queue length of each link in experiment 4.

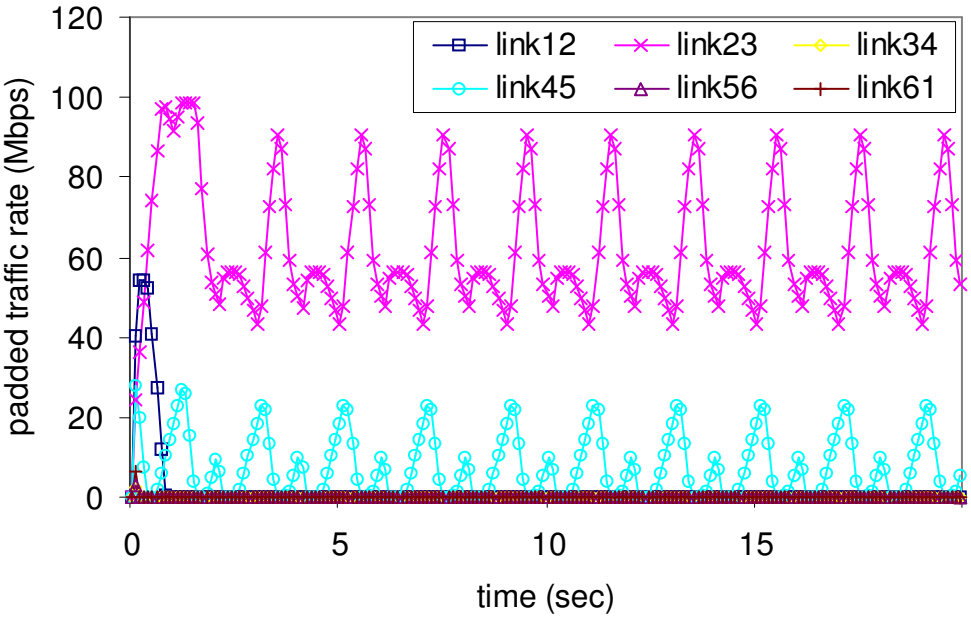

Fig. 40. Padded traffic rate on each link in experiment 5.

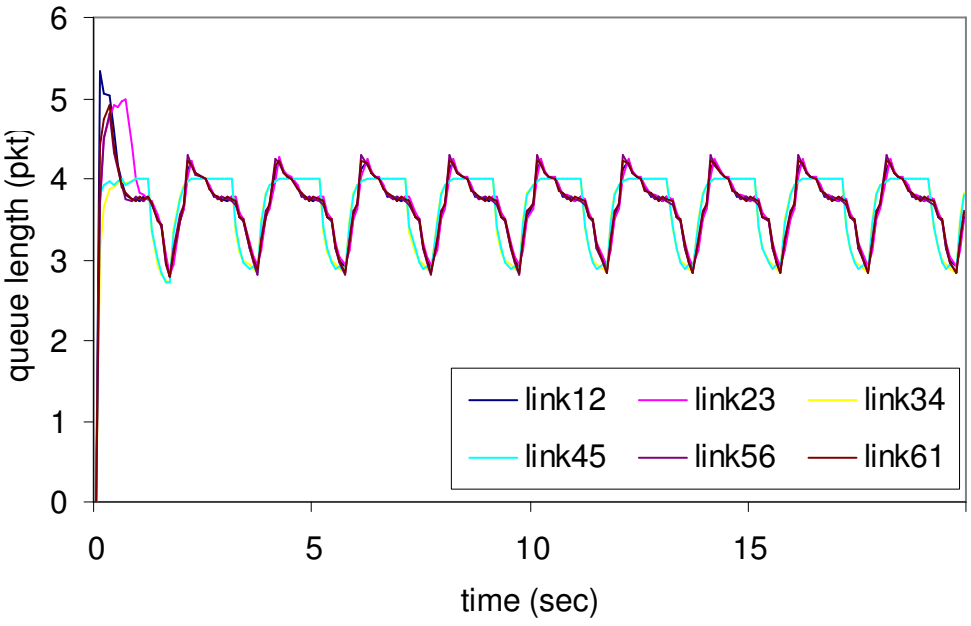

Fig. 41. Queue length of each link in experiment 5.

Table I. Events in experiment 5.

Event epoch (sec)	Actions
0	Internal session started
0.5	External session 2 between 7 and 8 started (event A)
1	External session 1 between 9 and 10 started (event B)
1.5	External session 1 closed (event C)
2	External session 2 closed (event D)
4.5	Composite event $Z : A, B, C, D$ for the 2 sec duration
$6.5, 8.5, \cdots$	Z repeated every 2 seconds
20	Internal session closed

overall rates at all links are almost identical to each other. The minor rate differences are also caused by outside events, external sessions. Obviously, the transmission rate differences increase due to the frequent "on" and "off" of the external sessions. Even though the transmission rates fluctuate at occurrences of new external events, group nodes do have consistent and similar changes, making it difficult for advisory to determine the nature of the events. Fig. 41 shows the queue length changes that reflect arrivals and departures of external sessions, and it is noted that the queue lengths do converge to the steady state values after each set of events.

F. Conclusion

In this chapter, we proposed traffic-concealing, anti-eavesdropping communication protocols for secure group communications. By using dispersed transport of message fragments, we spread the interaction communication patterns among the multicast participants to counter traffic analysis attacks. The two different types of broadcast-

based data transport primitives have been proposed to meet different performance and security requirements. The two-hop relay communication has less uniform traffic pattern, but its message delivery time is shorter. On the other hand, the permutation ring has more symmetric traffic patterns but longer latency time. Of course, to crack the message, one still must have full knowledge about the permutation and shuffling rules. It is of great interest to further expand different types of transport primitives to conceal traffic patterns of group communications.

To resist traffic pattern profiling attacks, we develop a window-based adaptive traffic padding scheme for group members to balance their traffic flows. Both simulation and empirical results show that our scheme is highly efficient and robust to different traffic conditions. It has been tested on a highly secure network appliance machine to demonstrate its feasibility.

CHAPTER V

SUMMARY

The objective of this research is to develop a generic modeling and design method of network traffic throttling. Increase and decrease (I-D) congestion control rule is widely adopted in network traffic management because of its cost-effectiveness. I proposed a modeling and design methodology for I-D congestion controllers to guarantee asymptotic stability and eliminate traffic oscillation, based on the sliding mode control (SMC) theory. My scheme addresses the discontinuous operations of I-D controller that has been largely disregarded in existing literature, and shows that discontinuity plays a crucial role in optimization of the I-D based congestion control algorithms. Increase-Decrease control laws are very simple to implement and highly robust to parameter and model uncertainties. Their stability does not require a precise network dynamics model.

On the basis of this sound system model, I also showed that one can design highly effective network protection schemes against DDoS attacks, and one apply the throttling technique to conceal the traffic patterns for protection of group communications.

DDoS management can be considered a special case of the flow control problem. Based on the theory about modeling and management of network traffic, I developed a backward-propagation feedback control strategy for DDoS defense. When a host finds itself becoming a hot spot, it informs neighboring nodes and routers to reduce influx of packets. Reduction of packet influx is propagated backward to the sources. If a source is normal, it will reduce its sending rate when backpressure is propagated to it. If a source ignores traffic backpressure and keeps infusing packets, it will be identified as an attacker and its packets will be dropped by the switches or routers

at the edge of Internet. This backward-propagation feedback control strategy adopts a simple hop-by-hop on-off control scheme. Its design is based on well-established sliding mode control theory. Thus, except prevent DDoS attacks, it also provides smooth traffic and bounded queue size. Another advantage of hop-by-hop scheme is that it responses much faster than end-to-end scheme so that delay can be ignored. This is very important when the bandwidth-delay product becomes very large.

Private group communications are critical to protection of large scale information systems. Standard encryption algorithms (e.g. IDEA, DES, RSA, AES, SHA1, etc.), together with their key management systems are necessary, yet insufficient to enable group members to conceal their interactions. In addition to cryptographic protection of information contents, concealment of the network traffic patterns (volumes, peak times, etc.) is also important to prevent unveiling the interactions between group members. A network is said to be unobservable if the volumes, times, sender-recipient pairing and other similar measures cannot be related to the underlying business activities; implying ineffectiveness of traffic analysis. Workload padding and packet routing are basic mechanisms to counter passive traffic analysis attacks. However, traffic masking is very expensive in terms of bandwidth consumption. The challenge here is how to make an optimal tradeoff between traffic pattern anonymity and network performance. Under this topic, I developed a new technique for protection of group communications by concealment of sender-recipient pairs, and their traffic patterns. Packet flows among nodes are made balanced, to eliminate traffic patterns related to group activities. I proposed a sliding window-based flow control scheme to control transmission of payload and dummy packets. Our algorithms allow flexible tradeoff between the degree of traffic uniformity, and that of the performance costs. This is particularly important for an open network environment, where the users may not have full control of the network resources.

REFERENCES

[1] J. Filipiak, *Modeling and Control of Dynamic Flows in Communication Networks*. Berlin: Springer-Verlag, 1988.

[2] V. I. Utkin, *Sliding Modes in Control Optimization*. Berlin: Springer-Verlag, 1992.

[3] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice Hall, 1991.

[4] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN*, vol. 1, no. 12, pp. 1–14, June 1989.

[5] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, October 1994.

[6] D. Sisalem and A. Wolisz, "LDA+ TCP-friendly adaptation: A measure-ment and comparison study," in *Proc. International Workshop on Network and Operating Systems Support for Digital Audio and video (NOSSDAV)*, Chapel Hill, NC, June 2000, pp. 241–251.

[7] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," in *Proc. ACM SIGCOMM*, Stanford, CA, August 1988, pp. 158–181.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[9] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001, pp. 1510–1519.

[10] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "Blue: A new class of active queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 513–528, August 2002.

[11] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001, pp. 1726–1734.

[12] K. B. Kim, A. Tang, and S. H. Low, "Design of AQM in supporting TCP based on well-known AIMD model," in *Proc. IEEE Globecom*, San Francisco, CA, December 2003, pp. 3226–3230.

[13] J. S. Sun, G. R. Chen, K. T. Ko, S. Chan, and M. Zukerman, "PD-controller: A new active queue management scheme," in *Proc. IEEE Globecom*, San Francisco, CA, December 2003, pp. 3103–3107.

[14] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *Proc. ACM SIGCOMM*, San Diego, CA, August 2001, pp. 123–134.

[15] C. L. F. Ren, X. Ying, X. Shan, and F. Wang, "A robust active queue management algorithm based on sliding mode variable structure control," in *Proc. IEEE INFOCOM*, New York, June 2002, pp. 64–79.

[16] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proc. IEEE INFOCOM*, New York, March 1999, pp. 1346–1355.

[17] D. Lin and R. Morris, "Dynamics of random early detection," in *Proc. ACM SIGCOMM*, Cannes, French Riviera, France, September 1997, pp. 127–137.

[18] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED gateway," in *Proc. IEEE INFOCOM*, New York, March 1999, pp. 1320–1328.

[19] M. May, T. Bonald, and T. Bolot, "Analytic evaluation of RED performance," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, March 2000, pp. 1415–1424.

[20] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000, pp. 139–150.

[21] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Stanford, CA, August 1988, pp. 314–329.

[22] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, no. 3, pp. 28–37, May 2001.

[23] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000, pp. 43–56.

[24] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001, pp. 631–640.

[25] X. Zhang and K. G. Shin, "Second-order rate-control based transport protocols," in *Proc. IEEE International Conference on Network Protocols*, Riverside, CA, November 2001, pp. 342–351.

[26] K. W. Lee, T. Kim, and V. Bharghavan, "A comparison of end-to-end congestion control algorithms: The case of aimd and aipd," in *Proc. IEEE GLOBECOM*, San Antonio, TX, November 2001, pp. 1580–1584.

[27] S. Athuraliya, D. E. Lapsley, and S. H. Low, "Random early marking for Internet congestion control," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 48–53, May/June 2001.

[28] R. J. Gibbens and F. P. Kelly, "Distributed connection acceptance control for a connectionless network," in *Proc. IEE International Teletraffic Congress*, Edinburgh, Scotland, June 1999, pp. 941–952.

[29] ——, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, no. 12, pp. 1969–1985, August 1999.

[30] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[31] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, December 1999.

[32] V. Misra, W. B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application on RED," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000, pp. 151–160.

[33] M. Allman, W. Stevens, and V. Paxson, "TCP congestion control," RFC 2581, April 1999.

[34] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Proc. IEEE INFOCOM*, New York, March 1999, pp. 1337–1345.

[35] T. Montgomery, "A loss tolerant rate controller for reliable multicast," West Virginia University, Tech. Rep. NASA-IVV-97-011, August 1997.

[36] L. Fridman and A. Levant, "Sliding modes of higher order as a natural phenomenon in control theory," in *Robust Control via Variable Structure and Lyapunov Techniques, Lecture Notes in Control and Information Science.* London: Springer-Verlag, 1996, no. 217, pp. 107–133.

[37] J. Bolot and A. Shankar, "Dynamical behavior of rate-based flow control mechanism," *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 4, pp. 35–49, April 1990.

[38] Headline News. (Janary 26, 2001) DDoS attacks block Microsoft web sites.

[39] ——. (Febrary 7-11, 2000) DDoS attacks on Yahoo, Buy.com, eBay, Amazon, Datek, E*Trade, CNN.

[40] L. Garber, "Denial-of-service attacks rip the Internet," *IEEE Computer*, vol. 33, no. 4, pp. 12–17, April 2000.

[41] (2000) The "mstream" distributed denial of service attack tool. Technology White Paper. [Online]. Available: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

[42] (1999) The DoS project's "trinoo" distributed denial of service attack tool. Technology White Paper. [Online]. Available: http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt

[43] (1999) The "tribe flood network" distributed denial of service attack tool. Technology White Paper. [Online]. Available: http://staff.washington.edu/dittrich/misc/tfn.analysis.txt

[44] (1999) The "stacheldraht" distributed denial of service attack tool. Technology White Paper. [Online]. Available: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt

[45] (2000) CERT® Advisory CA-99-17 denial-of-service tools. CERT/CC. [Online]. Available: http://www.cert.org/advisories/CA-1999-17.html

[46] S. Dietrich, N. Long, and D. Dittrich, "Analyzing distributed denial of service attack tools: The shaft case," in *Proc. USENIX LISA*, New Orleans, LA, December 2000, pp. 329–339.

[47] C. L. Schuba, I. Krsul, M. Kuhn, E. Sparford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, May 1997, pp. 208–223.

[48] (2000) CERT® distributed system intruder tools workshop report. CERT/CC. [Online]. Available: http://staff.washington.edu/dittrich/talks/cert/

[49] (2000) CenterTrack: An IP overlay network for tracking DoS floods. Technology White Paper. [Online]. Available: http://www.arbor.net/downloads/research51/stone00centertrack new.pdf

[50] (2001) RID (remote intrusion detector) tool 1.11. SecurityFocus Symantec Corporation. [Online]. Available: http://www.theorygroup.com/Software/RID

[51] (2000) Detecting and decoding "mstream" traffic. Technology White Paper. [Online]. Available: http://packetstorm.securify.com/distributed/Turner.mstream

[52] (2001) Purgatory 101: Learning to cope with the SYNs of the Internet. Technology White Paper. [Online]. Available: http://packetstormsecurity.com/papers/contest/RFP.doc

[53] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks," RFC 2267, January 1998.

[54] (2000) Cisco–Defining strategies to protect against TCP SYN denial of service attacks. Technology White Paper. [Online]. Available: http://www.cisco.com/warp/public/707/4.pdf

[55] (2000) CERT® Advisory CA-2000-01 denial-of-service developments. CERT/CC and FedCIRC. [Online]. Available: http://www.cert.org/advisories/CA-2000-01.html

[56] Y. W. Chen, "Study on the prevention of SYN flooding by using traffic policing," in *Proc. IEEE/IFIP NOMS*, Honolulu, HI, April 2000, pp. 593–604.

[57] X. Geng and A. B. Whinston, "Defeating distributed denial of service attacks," *IT Professional*, vol. 2, no. 4, pp. 36–42, July 2000.

[58] A. Barkley, J. C. Liu, Q. T. L. Gia, M. Dingfield, and Y. Gokhale, "A testbed for study of distributed denial of service attacks," in *Proc. IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, West Point, NY, June 2000, pp. 123–132.

[59] C. M. Ozveren, R. Simcoe, and G. Varghese, "Reliable and efficient hop-by-hop flow control," *IEEE J. Select. Areas Commun.*, vol. 13, no. 4, pp. 642–650, May 1995.

[60] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, April 1993.

[61] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE J. Select. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[62] A. Levant, "Higher order sliding: Differentiation and black-box control," in *Proc. IEEE Conference on Decision and Control*, Sydney, Australia, December 2000, pp. 1703–1708.

[63] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* New York: Springer-Verlag, 1983.

[64] J. C. Liu, K. G. Shin, and C. Chang, "Prevention of congestion in packet-switched multistage interconnection networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 5, pp. 535–541, May 1995.

[65] H. S. Ramirez, "Nonlinear variable structure systems in sliding mode: The general case," *IEEE Transactions on Automatic Control*, vol. 34, no. 11, pp. 1186–1188, November 1989.

[66] Y. Xiong, "Control of variable structure spacecraft," Master's thesis, Chinese Academy of Space Technology, Beijing, June 1995.

[67] J. F. Raymond, "Traffic analysis: Protocols, attacks, design issues and open problems," in *Proc. the Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, July 2000, pp. 7–26.

[68] O. Berthold, H. Federrath, and M. Kohntopp, "Anonymity and unobservability on the Internet," in *Proc. the Workshop on Freedom and Privacy by Design*, Toronto, Canada, April 2000, pp. 57–65.

[69] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communication of the ACM*, vol. 24, no. 2, pp. 84–90, February 1981.

[70] P. Baran, "On distributed communications: Ix security, secrecy, and tamper-free considerations," Rand Corp., Memo RM-3765-PR, August 1964.

[71] B. R. Venkatraman and R. E. Newman-Wolfe, "Performance analysis of a method for high level prevention of traffic analysis using measurements from a campus network," in *Proc. IEEE Annual Computer Security Applications Conference (ACSAC)*, Orlando, FL, December 1994, pp. 288–297.

[72] V. Voydoc and S. Kent, "Security mechanisms in high-level network protocols," *ACM Computing Surveys*, vol. 15, no. 2, pp. 135–171, June 1983.

[73] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, May 1997, pp. 44–54.

[74] B. Timmerman, "A security model for dynamic adaptive traffic masking," in *Proc. New Security Paradigms Workshop*, Langdale, Cumbria, United Kingdom, September 1997, pp. 107–116.

[75] D. Chaum, "The dining cryptographers problem: unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, August 1988.

[76] Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, November 1979.

[77] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. AFIPS National Computer Conference*, Montvale, NJ, June 1979, pp. 313–317.

[78] G. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering*, vol. 15, no. 8, pp. 929–934, August 1989.

[79] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, February 2000.

[80] O. Rodeh, K. P. Birman, and D. Dolev, "Optimized group rekey for group communication systems," in *Proc. ISOC Network and Distributed Systems Security*, San Diego, CA, February 2000, pp. 39–48.

[81] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications*, vol. 32, no. 9, pp. 33–38, September 1994.

[82] R. H. Deng and T. T. Tjhung, "Novel approach to secure broadcast in distributed systems," in *Proc. IEEE Annual International Carnahan Conference on Security Technology*, Sanderstead, United Kindom, October 1995, pp. 391–395.

[83] A. Ballardie, "Scalable multicast key distribution," RFC 1949, July 1996.

[84] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proc. ACM SIGCOMM*, Cambridge, MA, August 1999, pp. 263–274.

[85] J. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proc. ACM SIGCOMM*, San Francisco, CA, September 1993, pp. 289–298.

[86] A. B. Downey, "Using PATHCHAR to estimate internet link characteristics," in *Proc. ACM SIGCOMM*, Cambridge, MA, August 1999, pp. 241–250.

[87] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, August 2002, pp. 295–308.

[88] R. Kapoor, L. Chen, L. Lao, M. Sanadidi, and M. Gerla, "CapProbe: A simple and accurate capacity estimation technique," in *Proc. ACM SIGCOMM*, Portland, OR, August 2004, pp. 67–78.

[89] K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000, pp. 283–294.

[90] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network.* Reading, MA: Addison-Wesley, 1997.

[91] K. P. Birman and T. A. Joseph, "Reliable communication in the presence of failures," *ACM Transactions on Computer Systems*, vol. 5, no. 1, pp. 47–76, Febrary 1987.

[92] (2002) Emcast toolkit. University of Michigan. [Online]. Available: http://www.junglemonkey.net/emcast/

[93] (2002) The network wimulator - ns2. Information Sciences Institute, Univ. of Southern California. [Online]. Available: http://www.isi.edu/nsnam/ns/

APPENDIX A

RELATIVE DEGREE AND ZERO DYNAMICS

Definition A.1 (*Lie* Derivative)

Given a smooth scalar function $h(x) : R^n \to R$ and a smooth vector function $f(x) = [f_1(x), f_2(x), \cdots, f_n(x)]^T : R^n \to R^n$, where $x = [x_1, x_2, \cdots, x_n]^T \in R^n$, the *Lie* derivative of $h(x)$ with respect to $f(x)$, $L_f h(x)$, is defined as the following scalar function

$$L_f h(x) \equiv \frac{\partial h}{\partial x} f(x) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} f_i(x). \tag{A.1}$$

We also denote

$$L_g L_f \phi(x) \equiv \frac{\partial (L_f \phi)}{\partial x} g(x), \tag{A.2}$$

and

$$L_f^0 \phi(x) \equiv \phi(x), \tag{A.3}$$

$$L_f^j \phi(x) \equiv L_f L_f^{j-1} \phi(x). \tag{A.4}$$

Definition A.2 (*Relative Degree* of SISO Systems)

Considering the following Single-Input-Single-Output (SISO) nonlinear system denoted by

$$\begin{cases} \dot{x} = f(x) + g(x)u, \\ y = h(x), \end{cases} \tag{A.5}$$

where the state vector $x = [x_1, x_2, \cdots, x_n]^T \in R^n$ and the input $u \in R$, the output $y \in R$. $f(x)$, $g(x)$ and $h(x)$ are smooth functions,

$$f(x) = [f_1(x), f_2(x), \cdots, f_n(x)]^T : R^n \to R^n,$$
$$g(x) = [g_1(x), g_2(x), \cdots, g_n(x)]^T : R^n \to R^n,$$
$$h(x) : R^n \to R.$$

An SISO system described by (A.5) has a *relative degree* r at x_0 if and only if:

(1) $L_g L_f^k h(x) = 0$ for x in a neighborhood of x_0 for $\forall k < r-1$;

(2) $L_g L_f^{r-1} h(x_0) \neq 0$.

Definition A.3 (*Relative Degree* of MIMO Systems)

Consider the following Multiple-Input-Multiple-Output (MIMO) nonlinear system denoted by

$$\begin{cases} \dot{x} = f(x) + G(x)u, \\ y = h(x), \end{cases} \tag{A.6}$$

where the state vector $x = [x_1, x_2, \cdots, x_n]^T \in R^n$ and the input vector (the control variables) $u = [u_1, u_2, \cdots, u_m]^T \in R^m$, and the output vector $y = [y_1, y_2, \cdots, y_m]^T \in R^m$. $f(x)$ and $h(x)$ are smooth function vectors. $G(x)$ is a smooth function matrix.

$$f(x) = [f_1(x), f_2(x), \cdots, f_n(x)]^T : R^n \to R^n,$$
$$G(x) = [g_1(x), g_2(x), \cdots, g_m(x)]^T,$$
$$g_i(x) = [g_{i,1}(x), g_{i,2}(x), \cdots, g_{i,n}(x)]^T : R^n \to R^n, \text{ for } i = 1, 2, \cdots, m,$$
$$h = [h_1(x), h_2(x), \cdots, h_m(x)] : R^n \to R^m.$$

An MIMO system described by (A.6) has a (vector) *relative degree* $[r_1, r_2, \cdots, r_m]$ at x_0 if and only if:

(1) $L_{g_j} L_f^k h_i(x) = 0, 1 \leq i, j \leq m$ for all x in a neighborhood of x_0 for $\forall k < r_i - 1$;

(2) $L_{g_j} L_f^{r_i - 1} h_i(x_0) \neq 0$.

APPENDIX B

STABILITY THEOREMS

Given a system described by

$$\dot{x} = f(x,t), \tag{B.1}$$

where $x = [x_1, x_2, \cdots, x_n]^T \in R^n$. We make a notation that $x(t, x_0)$ is a solution of (B.1) with initial value x_0 at $t = 0$.

Definition B.1 (Equilibrium Point)

Considering a nonlinear system (B.1), x^* is called an equilibrium point of system (B.1) if and only if $f(x^*, t) = 0$ for $\forall t$.

Definition B.2 (Stability)

An equilibrium point of system (B.1), is *bounded stable* if and only if for $\forall \varepsilon > 0$, $\exists \delta > 0$, which makes $\parallel x(t) - x^* \parallel < \varepsilon$ hold for $\forall x_0$, which satisfies $\parallel x_0(t) - x^* \parallel < \varepsilon$ and $\forall t > 0$. Further if $\lim_{t \to +\infty} \parallel x(t) - x^* \parallel = 0$ also holds on, then the equilibrium point x^* is asymptotically stable. We always refer to asymptotically stable unless otherwise stated.

Definition B.3 (Positive/Negative Definite Functions)

A continuously differentiable function $f : R^n \to R_+$ is said to be *positive definite* in a region U of R that contains the origin if (1) $f(0) = 0$ and (2) $f(x) > 0$ for $x \in U$ and $x \neq 0$. $f(x)$ is said to be *positive semidefinite* if $f(x) \geq 0$ for $x \in U$ and $x \neq 0$.

Conversely, if condition (2) is replaced by $f(x) < 0$, then $f(x)$ is said to be *negative definite*. $f(x)$ is said to be *negative semidefinite* if $f(x) \leq 0$.

Theorem B.1 (Lyapunov Stability of Autonomous Systems) *Let $x = 0$ be an equilibrium point for a system described by:*

$$\dot{x} = f(x), \tag{B.2}$$

where $f : U \to R^n$ is a continuously differentiable function and $U \subset R^n$ a domain that contains the equilibrium point of (B.2), x^*. Let $V : U \to R$ be a continuously differentiable, positive definite function in U.

1. If $\dot{V}(x) = \frac{\partial V}{\partial x} f(x)$ is negative semidefinite, then $x = x^*$ is a stable equilibrium point.

2. If $\dot{V}(x)$ is negative definite, then $x = x^*$ is an asymptotically stable equilibrium point.

In both cases above V is called a Lyapunov function. *Moreover, if the conditions hold for all $x \in R^n$ and $\|x\| \to \infty$ implies that $V(x) \to \infty$, then $x = x^*$ is globally stable in case 1 and globally asymptotically stable in case 2.*

Lemma B.2 *A time-invariant linear system $\frac{d^n y}{dt^n} + \omega_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \omega_1 \dot{y} + \omega_0 y = 0$, where $\omega_i \in R$ for $i = 1, 2, \cdots, n-1$, is asymptotically stable if and only if all roots of its characteristic polynomial $p^n + \omega_{n-1} p^{n-1} + \cdots + \omega_1 p + \omega_0$ have negative real parts.*

For example, $\dot{y} + \omega y = 0$ is asymptotically stable if and only if $\omega > 0$.

Theorem B.3 (Lyapunov uniform asymptotic stability of non-autonomous systems) *Let $x = x^*$ be an equilibrium point of a system described by (B.1) and $U \subset R^n$ a domain containing it. Let $V : U \times [0, \infty] \to R$ be a continuously differentiable function that satisfies:*

$$V_1(x) \leq V(x, t) \leq V_2(x) \tag{B.3}$$

$$\dot{V}(x, t) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(x, t) \leq -V_3(x) \tag{B.4}$$

for all $t \geq t_0$, *and* $x \in U$, *where* $V_1(x)$, $V_2(x)$ *and* $V_3(x)$ *are continuous positive definite functions on* U. *Then,* $x = x^*$ *is uniformly asymptotically stable and* V *is called a Lyapunov function.*

VITA

Yong Xiong was born in Fengdu, the People's Republic of China. He received his Bachelor of Science degree from Tsinghua University, Beijing, China, in July 1992, and Master of Science degree from the Chinese Academy of Space Technology, Beijing, China, in June 1995, both in electrical engineering. His permanent address is: Dong Gao Di Mei Yuan Li ♯24-5-5, Feng Tai District, Beijing 100076, China.

The typist for this thesis was Yong Xiong.