CONGESTION CONTROL FOR NETWORKS IN CHALLENGED ENVIRONMENTS

by

GUOHUA ZHANG

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2008

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my advisor, Dr. Yonghe Liu, for his support, patience, and encouragement throughout my graduate studies. I would not finish my Ph.D. studies without his expert guidance and generous support. I wish to thank Dr. Hao Che, Dr. Sajal K. Das, Dr. Manfred Huber, Dr. Mohan Kumar for their interest in my research and for taking time to serve in my dissertation committee.

I would also like to thank Dr. Atilla Dogan at Department of Mechanical and Aeronautic Engineering, UT at Arlington, for his help in control theory.

I appreciate the friendship from my fellow students I have come to know during the past years. I want especially to thank Feng Ji, Tuli Nivas, Jing Wang, Quan Wen, Jiaxing Xue.

Finally, I am deeply indebted to my parents for their patience, endless support and encouragement.

July 24, 2008

ABSTRACT

CONGESTION CONTROL FOR NETWORKS IN CHALLENGED ENVIRONMENTS

GUOHUA ZHANG, Ph.D.

The University of Texas at Arlington, 2008

Supervising Professor: Yonghe Liu

Congestion occurs when resource demands exceed the capacity of a network. The goal of congestion control is to use the network as efficiently as possible. While extensive efforts have been devoted to providing optimization based, distributed congestion control schemes for efficient bandwidth utilization and fair allocation in the Internet and wireless networks, little consideration was given to congestion control for networks in challenged environments, specifically for networks with time-varying link capacities and networks that intermittently communicate. In this dissertation, we explore optimal congestion control strategies for such networks based on optimization techniques and repeated game model.

For networks with time varying link capacities, we explicitly model link capacities to be time varying and investigate the corresponding optimal congestion control strategies. In particular we propose a primal-dual congestion control algorithm which is proved to be trajectory stable in the absence of feedback delay. Different from system stability around a single equilibrium point, trajectory stability guarantees the system is stable around a time varying reference trajectory. Moreover, we obtain suf-

ficient conditions for the scheme to be locally stable in the presence of delay. The key technique is to model time variations of capacities as perturbations to a constant link. Furthermore, to study the robustness of the algorithm against capacity variations, we investigate the sensitivity of the control scheme and through simulations study the tradeoff between stability and sensitivity.

For a set of challenged networks where continuous end-to-end connectivity may not exist, network nodes may only communicate during opportunistic contacts (they are often referred to as delay tolerant networks or opportunistic networks). While custody transfer can provide certain reliability in delay in these networks, a custodian node cannot discard the message unless its life time expires or the custody is transferred to another node after a commitment. This creates a challenging decision making problem at a node in determining whether to accept a custody transfer: on one hand, it is beneficial to accept a large number of messages as it can potentially advance the messages toward their ultimate destinations and network utilization can be maximized; on the other hand, if the receiving node over-commits itself by accepting too many messages, it may find itself setting aside an excessive amount of storage and thereby preventing itself from receiving further potentially important, high yield (in terms of network utilization) messages. To solve this congestion control problem, we apply the concepts of revenue management, and employ dynamic programming to develop congestion control strategies. For a class of network utility functions, we show that our solution is optimal. More importantly, our solution is distributed in nature where only the local information such as available buffer of a node is required. This is particularly important given the nature of delay tolerant networks where global information is often not available and the network is inherently dynamic. Our simulation results show that the proposed congestion management scheme is effective in avoiding congestion and balancing network load among the nodes.

In the above scheme, we have assumed that the time horizon is finite in making the decision of resource allocation. However, in practice, in certain situations, it might be difficult or impossible to predict when the dynamic behavior will stop. As an alternative solution, we also employ repeated games to model the decision making for custody transfer and propose a new congestion control strategy. The repeated game based approach is particularly suitable for the situations where a node cannot be certain when a contact will occur and when the dynamic behavior is going to stop. Our simulation results show that the control strategy based on repeated games is effective in avoiding congestion and balancing network load.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

ABSTRACT iii

LIST OF FIGURES ix

LIST OF TABLES xi

Chapter

1. INTRODUCTION 1

1.1 Congestion Control in the Internet 1

1.1.1 TCP Reno 2

1.1.2 Active Queue Management (AQM) 4

1.1.3 Convex Optimization based Congestion Control Scheme 6

1.2 Congestion Control in Networks with Time-Varying Link Capacities . 14

1.3 Congestion Control in Intermittently Communicating Networks . . . 16

1.4 Repeated Game and Network Resource Allocation 21

1.4.1 Basics of Game Theory 21

1.4.2 Repeated Game and Resource Allocation 24

1.5 Motivation of Research 25

1.6 Main Contributions and Organization 27

2. CONGESTION CONTROL IN NETWORKS WITH TIME VARYING LINK CAPACITIES 29

2.1 Introduction 29

2.2 Motivation 31

2.3 Problem Formulation and Algorithm Design 33

2.3.1 Preliminary 33

2.3.2 Problem Formulation 35
2.3.3 Congestion Control Algorithm 36
2.4 Several Definitions of Stability 38
2.5 Stability Analysis without Delay 40
2.6 Stability Analysis with Delay 45
2.7 Sensitivity Analysis of Link Capacity with Perturbation 54
2.8 Simulation 57
2.8.1 System Setup 57
2.8.2 Global Stability 58
2.8.3 Local Stability and Sensitivity 60
2.9 Conclusion 68
3. CONGESTION CONTROL IN INTERMITTENTLY COMMUNICATING NETWORKS — DYNAMIC PROGRAMMING APPROACH 69
3.1 Introduction 69
3.2 Related Work 71
3.3 Preliminaries 73
3.4 Problem Formulation 74
3.4.1 System Model 75
3.4.2 State Variable and Action Variables 77
3.4.3 Opportunity Cost and Benefit Function 78
3.5 Single Node Congestion Control 79
3.5.1 Optimal Strategy for Accepting Custody Transfer 81
3.5.2 Discussion 85
3.6 Network Congestion Control 87
3.6.1 Optimal Policy with Global Information 87
3.6.2 Distributed Optimal Policy 89

3.7 Simulation 91
3.7.1 Simulation Settings 91
3.7.2 Simulation Results and Discussions 93
3.8 Conclusion 96
4. REPEATED GAME MODELING OF CONGESTION MANAGEMENT IN INTERMITTENTLY COMMUNICATING NETWORKS 98
4.1 Introduction 98
4.2 Related Work 99
4.3 Basics of Repeated Game 100
4.4 Problem Formulation 103
4.5 The Custody Transfer Game 105
4.6 Control Strategy 109
4.6.1 Control Strategy 109
4.6.2 Strategy Analysis 110
4.7 Simulation 112
4.7.1 Simulation Settings 112
4.7.2 Custodian Cost Function 114
4.7.3 Simulation Results and Discussion 115
4.8 Conclusion 118
5. CONCLUSION 122
REFERENCES 124
BIOGRAPHICAL STATEMENT 134

LIST OF FIGURES

Figure Page

1.1 Congestion Window of TCP Reno 3

1.2 RED probability profile 5

1.3 Small-scale and large-scale fading (reference [15]) 15

1.4 The overaly network approach 18

2.1 Equilibrium point stability 40

2.2 Asymptotically stable equilibrium point 41

2.3 Trajectory-stability illustration 41

2.4 Numerical study of control parameters on $\|T(s)\|$. (a) k's effect on $|T(s)|$. (b) β's effect on $\|T(s)\|$ 57

2.5 A single bottleneck link topology 59

2.6 Global stability with link perturbation. (a) Queue length at link 12. (b) Utilization of link 12. (c) Source rate of a traffic 61

2.7 Global stability with link perturbation and UDP traffic. (a) Queue length at link 12. (b) Utilization of link 12. (c) One source rate 62

2.8 A multihop network topology 63

2.9 Local stability without link perturbation. (a) Queue length at link 23. (b) Utilization of link 23. (c) Source rate of flow 2 64

2.10 Local stability with link perturbation - stable scenario. (a) Queue length at link 23. (b) Utilization of link 23. (c) Source rate of flow 2 66

2.11 Local stability with link perturbation - unstable scenario. (a) Queue length at link 23. (b) Utilization of link 23. (c) Source rate of flow 2 . 67

3.1 Simple DTN Scenario 76

3.2 State transition relation 77

3.3 Nodes' position snapshot at 600s for 40/6 node mix for case listed in

Fig. 3.5a 93

3.4 Hop count distribution 94

3.5 Load distribution in nodes - 40/6 node mix. (a) message generation rate $\lambda = 1/2$ message/second. (b) $\lambda = 5/9$ message/second 95

3.6 Load distribution in nodes - 50/8 node mix. (a) message generation rate $\lambda = 1/2$ message/second. (b) $\lambda = 5/9$ message/second 96

3.7 Load distribution in nodes - 40/6 node mix (traffic generated at constant speed 0.5 message/s) 97

4.1 Custody transfer of two players 104

4.2 Game scenario of two players 106

4.3 Load distribution in nodes - 40/20 node mix, message arrival density = 1/2 message/second. (a) Load at $t = 400s$. (b) Load at $t = 800s$ . . . 116

4.4 Load distribution in nodes - 40/20 node mix, message arrival density = 2/3 message/second. (a) Load at $t = 400s$. (b) Load at $t = 800s$ . . . 117

4.5 Load distribution in nodes - 50/20 node mix, message arrival density = 1/2 message/second. (a) Load at $t = 400s$. (b) Load at $t = 800s$ . . . 118

4.6 Load distribution in nodes - 50/20 node mix, message arrival density = 2/3 message/second. (a) Load at $t = 400s$. (b) Load at $t = 800s$ . . . 119

4.7 Load distribution in nodes - 50/30 node mix, message arrival density = 2/3 message/second. (a) Load at $t = 400s$. (b) Load at $t = 800s$ . . . 120

4.8 Load distribution in nodes - 60/30 node mix, message arrival density = 2/3 message/second. (a) Load at $t = 400s$. (b) Load at $t = 800s$ . . . 121

4.9 The throughput for several simulation scenarios at $t = 400s$ and $t = 800s$ (message arrival density = 2/3 message/second) 121

LIST OF TABLES

Table		Page
1.1	Payoff matrix of Prisoners' Dilemma	22
3.1	Simulation Parameters	92
4.1	Payoffs for the prisoners' dilemma	101
4.2	Payoff matrix of a game scenario between node i and node N_i^t	106
4.3	General form of a payoff matrix in a stage game for custody transfer	107
4.4	A specific payoff matrix of a stage game between node i and node N_i^t	108
4.5	Simulation parameters	113

CHAPTER 1

INTRODUCTION

1.1 Congestion Control in the Internet

The Internet is a worldwide-interconnected computer network that transmits data by packet switching based on the TCP/IP suite. Originated from the modest research network ARPANET, the Internet experienced exponential growth in the past three decades. Today, it connects hundreds of millions of machines and end systems. It is generally believed that the great success of the Internet should be attributed to the success of its protocols [1].

In the Internet protocol architecture, two protocols are defined for data transmission right above the IP layer. One is User Datagram Protocol (UDP), a simple and minimal protocol to send messages over the IP layer. It is transaction oriented, without guarantee of delivery and duplication protection. Another one is Transmission Control Protocol (TCP). TCP is primarily used by file transfer applications which need reliable, in-sequence delivery of packets from the source to the destination. Congestion control is implemented within the transport layer protocol TCP.

Congestion occurs when resource demands exceed the available capacity. Early in the Internet evolution, it was recognized that unrestricted access to the Internet resulted in poor performance in the form of low network utilization and high packet loss rates. This phenomenon known as congestion collapse, led to the development of the first congestion control algorithm for the Internet [18]. The basic idea behind the algorithm was to detect congestion in the network through packet losses. Upon detecting a packet loss, the source reduces its transmission rate; otherwise, it increases

the transmission rate. The original algorithm has undergone many minor, but important changes, but the essential features of the algorithm used for the increase and decrease phases of the algorithm have not changed through the various versions of TCP, such as TCP-Tahoe, Reno, NewReno, SACK. We next use TCP Reno to explain the mechanism of congestion control.

1.1.1 TCP Reno

TCP Reno has performed remarkably well and has prevented severe congestion as the Internet expanded by five orders of magnitude in size, speed, load, and connectivity. TCP Reno is the only deployed congestion control scheme in the current Internet.

A TCP Reno source sends packets using a sliding window algorithm. Its sending rate is controlled by the congestion control window size, which is the maximum number of packets that have been sent, yet not acknowledged. When the congestion window size becomes 0, the source must wait for an acknowledgement before sending a new packet. This is the "self-clocking " feature, which automatically slows down the source when a network becomes congested and round-trip time (RTT) increases. Since the number of packets sent every RTT is determined by the window size, the source rate is controlled by the window size divided by RTT. The key idea of TCP Reno is to additively increase congestion window size for additional bandwidth and multiplicatively decrease it while network congestion is detected.

A connection starts with a small window size of one packet, and the source increments its window by one every time it receives an acknowledgement. This doubles the window size every RTT and is called *slow start*. In this stage, the source exponentially increases its rate and can catch the available bandwidth quickly. When the window size reaches the slow start threshold (`ssThreshold`), the source enters

the *congestion avoidance* stage, where it increases its windows by the reciprocal of the current window size for each acknowledgement (ACK). This increases the window size by one in each RTT and is called as additive increase. When a loss is detected through duplicate ACKs, the source halves its window size, updates the value of the `ssThreshold`, and performs a *fast recovery* by retransmitting the lost packets. When a loss is detected through timeout expiration, the congestion window is set to one, and the source reenters the *slow start* stage. The whole stages of TCP Reno has been shown in Figure 1.1.

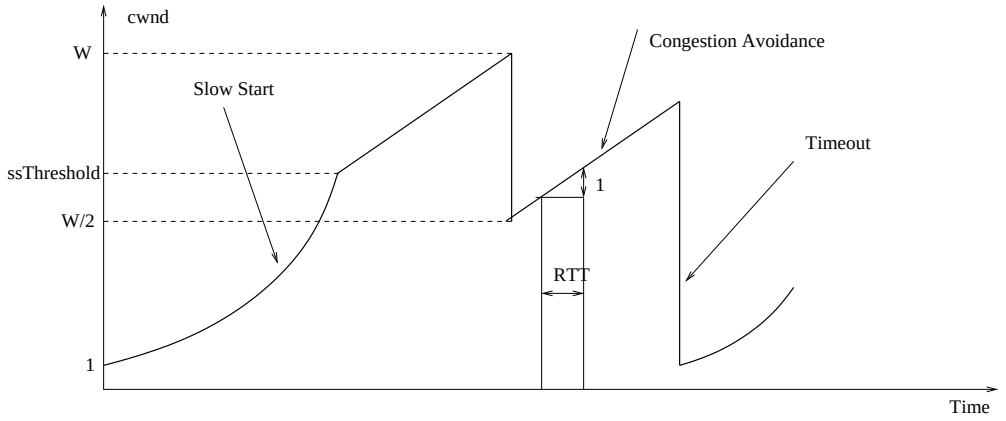


Figure 1.1: Congestion Window of TCP Reno

There are some drawbacks in using packet loss as an indiction of congestion. First, high utilization of bandwidth can be achieved only with a full queue. This is ill-suited to heavy-tailed TCP traffic. Second, the loss-based TCP will be degraded if it is used in wireless environments, where losses can be due to other effects.

It is worth noting the variant of TCP congestion control such as TCP Vegas algorithm uses queueing delay in the network as the indictor of congestion instead of packet loss. TCP Vegas updates its congestion window size based on end-to-end delay.

In the current Internet, the source algorithm is carried out by TCP, and the link algorithm is carried out by active queue management schemes to be discussed below.

1.1.2 Active Queue Management (AQM)

The AQM algorithm runs on a router, which updates and feedbacks congestion information to end-users. The feedback is in the form of packet loss, delay, and marking.

1.1.2.1 Droptail

Droptail is the simplest AQM scheme in the current Internet. It is just a first-in-first-out (FIFO) queue with limited capacity, and it simply drops any coming packets when the queue is full. Since it is simple and easy to implement, Droptail is the dominant AQM in the current Internet.

The congestion information in a Droptail queue is updated by the queueing process and is represented by the size of backlog buffer. The delay-based TCP algorithms such as TCP Vegas receive this information by sensing the changes in the round-trip delay.

For loss based TCP algorithm such as TCP Reno, the Droptail queue sends back one bit of information by a packet drop, which indicates that the router buffer is full and the network is congested.

1.1.2.2 Random Early Detection (RED)

RED was introduced as a mechanism to break synchronization among TCP flow [32]. Currently, it is primarily used as a mechanism to maintain small queue lengths in the Internet. Under RED, a packet is dropped or marked with a certain

probability which depends on the queue length. Instead of using the current queue length, RED maintains an exponentially-average estimate of the queue length and uses this to determine the marking probability. The basic idea is that, if the average queue length is large, then the packets should be marked with a high probability to let the source know that the level of congestion at the link is high, otherwise the marking probability is low.

Let b be the average queue length at a link. The marking probability at a link is determined according to the following profile:

$$f(b) = \begin{cases} 0 & \text{if } b \leq B_{\min}, \\ K(b - B_{\min}) & \text{if } B_{\min} < b < B_{\max}, \\ 1 & \text{if } b > B_{\max} \end{cases} \tag{1.1}$$

where K is some constant, and $B_{\min}$ and $B_{\max}$ are some thresholds such that the marking probability is 0 if the average queue length is below $B_{\min}$, and is equal to 1 if the average queue length is above $B_{\max}$.

The RED marking or dropping probability profile is shown in Fig 1.2.

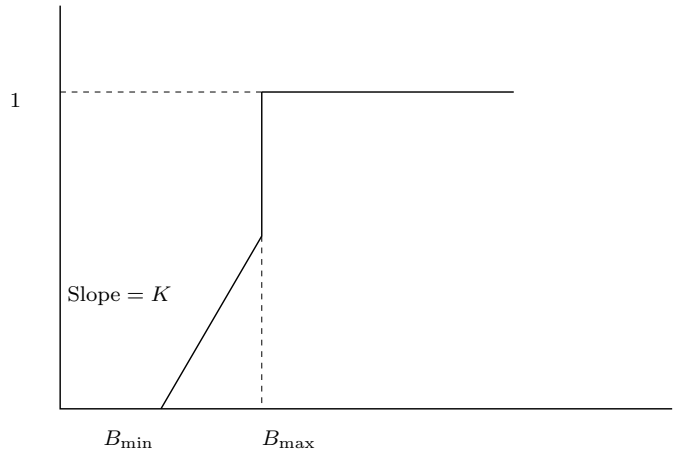


Figure 1.2: RED probability profile

1.1.2.3 Explicit Congestion Notification (ECN)

The Internet as it is today doesn't guarantee a high quality-of-service to its users. One of the reasons is that users in the Internet estimate the level of congestion by measuring packet loss or delay. Thus, a bad event (either high loss or high delay) has to occur before users can infer network congestion. To counter this problem, a protocol called ECN has been proposed for congestion indiction at the links [10, 18]. ECN marks allow routers to notify users about incipient congestion. A packet is said to be marked if a particular bit in its header is set to one. When the destination receives a packet with the ECN bit set equal to one, it conveys this information back to the source in the ACK packet. The TCP congestion control algorithm at the source can then treat this information in a manner similar to packet loss and cut its transmission rate. By providing early congestion notification, the ECN protocol can significantly reduce queueing delays and packet loss rates.

1.1.3 Convex Optimization based Congestion Control Scheme

Designs of congestion control started from intuition with little preliminary theoretical support, and were validated by simulations under simple network scenarios before deployment. Approximate mathematical models, whether stochastic or deterministic, continuous or discrete, were set up later to study their behaviors and possible refinements, but usually in a very small scales.

Over the past decade, theoretical research on congestion control of the Internet has been widely studied [8, 10–12, 18, 20–22]. Large strides have been taken in bringing analytical model into Internet congestion control. Key to these advances has been the explicit modeling of the congestion measure that communicates back to data sources the information on congestion in network resources being used. It is assumed that

each network link measures its congestion by a scalar variable (termed shadow price) and that sources have access to the aggregate price of links in their path [5]. The shadow price can be packet loss probability, queueing delay etc. From the discussion in Section 1.1, we can know that these assumptions are implicitly present in many variants of today's TCP protocols.

In [5, 6], Kelly et al studied the dynamic pricing and congestion control of the Internet by applying the supply and demand principle in economics. To provide multiple services, congestion control problems are modeled to price the resources and optimize the aggregate utility of all users. In kelly's seminal work, fairness and utilization issues on resource allocation can be integrated into a unified dynamic and distributed congestion system.

1.1.3.1 Resource Allocation and Congestion Control

Consider a large network shared by many users, where the goal is to share the network resource in a fair manner. The network resources that we consider here are the link bandwidths. There is no universally accepted definition of fairness. We usually associate a utility function with each user in the network, and refer to a resource allocation scheme as being fair if it maximizes the sum of utilities of all the users in the networks.

Fair Resource Allocation

A network is modeled as a set of resources indexed by l, called links, with finite capacities c_l. It is shared by a set of sources, indexed by i. Let $U_i(x_i)$ be the utility of source i as a function of its rate x_i. Associated with each source is a route which is a collection of links in the networks. Let R be a routing matrix whose (l, i) entry is 1 if source i's route includes link l and is 0 otherwise. $U_i(x_i)$ should be an

increasing, strictly concave, continuously differential function of the nonnegative rate. The resource allocation problem can be formulated as in [5], that is,

$$\max_{x \geq 0} \sum_i U_i(x_i) \quad \text{s.t.} \quad Rx \leq c, \tag{1.2}$$

where x is the vector of source rates and c is the vector of link capacities. The constraint says that, at each link l, the aggregate source rate $\sum_i R_{li} x_i$ does not exceed the capacity c_l. Since we assume that the utility functions are strictly concave, then the above convex optimization problem has a unique optimal solution.

The utility functions under this framework are closely related with the fairness criteria. It has been argued in [7] that

$$U(x) = w \log(x) \tag{1.3}$$

can lead to weighted proportional resource allocation. Intuitively, for a feasible small perturbation $x^* + \delta x$ of the optimum x^*, the decrease of the objective function should be

$$\sum_i U_i'(x_i^*) \delta x_i = \sum_i w_i \frac{\delta x_i}{x_i^*} \leq 0$$

The discussion of other types of fairness is referred to [7].

To solve the problem (1.2), we have to know the utility functions and routes of all the sources in the networks. In a large network such as the Internet, it is impractical to get global information of the network, we have to devise distributed solutions, where each source adapts its transmission rate based on local information.

Primal Algorithm

The function $p_l(x)$ is assumed to be nonnegative and denotes the penalty function corresponding to the capacity constraint at link l. This is commonly referred to as price at link l. The price at link l is a function of the total arrival rate $\sum_{i:l\in i} x_i$ at link l, and can be interpreted as a measure of congestion at link l. Consider the following problem where the constraints are embedded into the objective (1.2) by using penalty function,

$$\max_{x\geq 0} \sum_i U_i(x_i) - \sum_l \int_0^{\sum_{i:l\in i} x_i} p_l(x)dx \tag{1.4}$$

where $l \in i$ means that link l belongs to source i's route.

A first-order necessary condition for the optimum follows, for each i,

$$U_i'(x_i) - \sum_{l\in i} p_l\left(\sum_{j:l\in j} x_j\right) = 0$$

In continuous time, we have the following gradient-ascent algorithm that can be used to solve (1.4), and be used as source controller.

$$\dot{x}_i = k_i\left(1 - \frac{1}{U_i'(x_i)} \sum_{l\in i} p_l\left(\sum_{j:l\in j} x_j\right)\right) \tag{1.5}$$

where k_i is a positive function of x_i and $p_l\big(\sum_{j:l\in j} x_j\big)$. The algorithm (1.5), with appropriate price function $p_l(\cdot)$ (a static function), is referred to as the *primal algorithm*.

The most important feature of (1.5) is that source i's congestion controller only depends on the sum of the link prices along its route. Therefore, if there is a protocol that can compute the sum of link prices along its route, then it can implement its congestion controller in a distributed manner, without requiring any coordination with

other sources in the network. The price of $p_l(\cdot)$ will be computed by the routers in the network, it only depends on the total arrival rate at link l, each link's computation can be performed without requiring any coordination with other links. Thus, the source controller (1.5) is completely decentralized, except for the requirement of a protocol to communicate the link prices to the sources.

The primal algorithm also bears the following properties:

- The utility functions of the sources are not necessarily the same throughout the networks. This allows complete freedom of fairness control associated with utility function. It can also be applied to model networks with different types of applications [19].
- Note that current congestion control schemes hold dynamics at sources. One attractive feature of the primal algorithm is that it can be applied to approximate current TCP schemes with appropriate utility functions.
- As the penalty function of the capacity constraints, $p_l(\cdot)$ may have different choices for different purposes. With appropriate choice of such function, the optimal value of the primal control can approximate that of (1.2) arbitrarily closely.

Dual Algorithm

Consider the Lagrangian [53] of the problem (1.2),

$$\begin{aligned} L(x;p) &= \sum_i U_i(x_i) - p^T(Rx - c) \\ &= \sum_i \left(U_i(x_i) - x_i \sum_{l \in i} p_l \right) + p^T c \end{aligned} \tag{1.6}$$

where p is the vector of the Lagrange multipliers, or *shadow prices*. Then we have the dual problem as in [12]

$$\min_{p_l \geq 0} D(p) := \sum_i \max_{x_i \geq 0} \left(U_i(x_i) - x_i \sum_{l \in i} p_l \right) + \sum_l p_l c_l \tag{1.7}$$

The maximization over x_i can be carried out by individual sources based only on the aggregate price of its route as follows:

$$x_i = U_i'^{-1} \left(\sum_{l \in i} p_l \right) \tag{1.8}$$

where $U_i'^{-1}(\cdot)$ denotes the inverse of the derivative of U_i. Note that when the utility function U_i is convex, x_i is a decreasing function of the aggregate price $\sum_{l \in i} p_l$.

To solve (1.7), consider the gradient vector of the objective function $D(p)$, for each l,

$$\frac{\partial D}{\partial p_l} = c_l - \sum_{i: l \in i} x_i$$

Then one candidate for the link control could be

$$\dot{p}_l = -\gamma_l \frac{\partial D}{\partial p_l}$$

Thus, an algorithm to compute the shadow price at link l becomes

$$\dot{p}_l = \gamma_l \left(\sum_{i: l \in i} x_i - c_l \right) \tag{1.9}$$

where γ_l should be positive function. Since p_l has to be nonnegative, it is usually taken as

$$\dot{p}_l = \gamma_l \left(\sum_{i: l \in i} x_i - c_l \right)^+_{p_l} \tag{1.10}$$

(1.10) is simply the law of supply and demand. If the demand $\sum_{i:l\in i} x_i$ exceeds the supply c_l, increase price; otherwise, decrease it. Since each link only uses the total arrival rate into it to compute its price, it is a distributed algorithm.

The algorithm (1.8) and (1.10) is referred to as *dual algorithm*.

Primal-Dual Algorithm

Another approach for solving the resource allocation is to use the primal algorithm at the sources and the dual algorithm at the links. Thus, we continue to use the same algorithm at the source as in the primal algorithm:

$$\dot{x}_i = k_i\left(1 - \frac{1}{U_i'(x_i)}\sum_{l\in i} p_l\left(\sum_{j:l\in j} x_j\right)\right) \tag{1.11}$$

The link prices are generated according to the dual algorithm at links:

$$\dot{p}_l = \gamma_l\left(\sum_{i:l\in i} x_i - c_l\right)_{p_l}^{+} \tag{1.12}$$

It is worth noting that we only use general the utility function to explain the congestion control algorithms. If a specific utility function is chosen, the format of the congestion control algorithms may have the corresponding change, but the property of congestion control algorithm does not change.

1.1.3.2 Stability Analysis of Congestion Control Algorithm

For congestion control in the Internet, we are also concerned about the dynamics of the congestion control protocols in the domain of control theory. In particular, we are interested in the stability of the equilibrium point, especially in the presence of feedback delay, and in performance metrics such as speed of convergence, capacity

tracking, etc. The property of stability of congestion control is critical because our ultimate goal is to use the mechanism in a rather uncontrolled environment, where users come and go at will. It is therefore clear that any control that is designed for use in a real environment should be stable, that is, it should not exhibit the behavior that a slight deviation from the equilibrium point (also called optimum) will lead the control away from this point.

Stability of congestion control mechanisms can be qualitatively evaluated by the Lyapunov stability theorem [42]. There is no general principle to construct Lyapunov functions. Sometimes it is not easy to construct Lyapunov function. Whether or not the Lyapunov stability theorem can be employed to prove the stability of congestion control mechanisms lies in constructing an ideal Lyapunov function. Fortunately, stability can be established by showing that, with an appropriate formulation of an overall optimization problem, the network's implicit objective function provides a Lyapunov function for congestion control mechanisms [6].

In the primal algorithm, the penalty function is used as Lyapunov function, that is,

$$V(x) = \sum_i U_i(x_i) - \sum_l \int_0^{\sum_{i:l\in i} x_i} p_l(x)dx \tag{1.13}$$

Then it is easy to verify that

$$\frac{dV}{dt} = \sum_i \frac{\partial V}{\partial x_i}\dot{x}_i$$

is strictly positive when x is not an equilibrium point of (1.5), and zero otherwise. Therefore, we can conclude that the primal algorithm is stable. The detailed discussion on the stability of the primal algorithm is referred to [5].

For the dual algorithm and primal-dual algorithm, we can also construct Lyapunnov functions to prove that the dual algorithm and the primal-dual algorithm are stable. The detailed discussion of the stability analysis is referred to [18].

In the above discussion on stability analysis, we assume that there is no delay in the control system. In such cases, we can construct a Lyapunov function to prove that a congestion control algorithm is globally stable.

If there exists delay in the congestion control systems, it is difficult to establish global stability. Therefore, we turn to look for local stability. The technique in dealing with local stability is as follows: we linearize the system around its equilibrium point, employ *Laplace* transform to transform it from the time domain to the frequency domain, and derive sufficient conditions for local stability. The derived conditions limit the choice of the parameters of congestion control algorithms.

In this thesis, different from existing work, we focus on congestion control in challenged environments, particularly, networks with varying link capacity and networks that intermittently communicate. Below, we first examine these two types of challenged networks, then present the motivation of our research.

1.2 Congestion Control in Networks with Time-Varying Link Capacities

In this section, we first examine the factors to cause the link capacity to vary, then investigate the impact of link capacity variation on congestion control.

In wireless environments, there are many factors that affect the wireless links, and cause the link capacities to vary. We here list some important differences between a wired link and wireless link [16, 17]:

- *Decreasing signal strength.* Electromagnetic radiation attenuates as it passes through matter. Even in free space, the signal will disperse as the distance between sender and receiver increases (Path loss).
- *Interference from other sources.* Radio sources transmitting in the same frequency band will interference with each other.

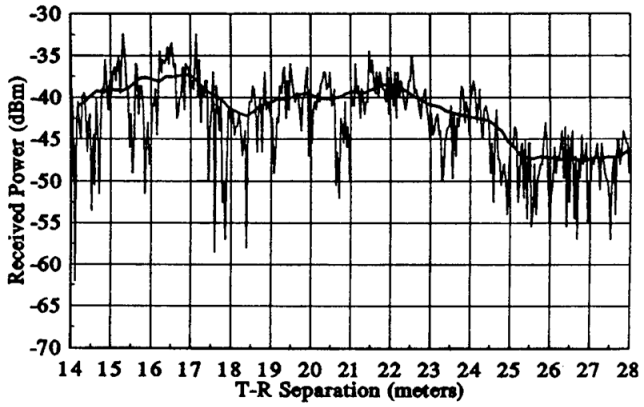


Figure 1.3: Small-scale and large-scale fading (reference [15])

- *Multipath propagation.* Multipath propagation occurs when portions of the electromagnetic wave reflect off objects and the ground, taking paths of different lengths between a sender and receiver. Moving objects between the sender and receiver can cause multipath propagation to change over time.
- *Shadow Fading.* A signal transmitted through a wireless channel will typically experience random variation due to blockage from objects in the signal path, giving rise to random variation of the received power at a given distance.

  Figure 1.3 shows the time varying link capacity caused by link fading.

In multihop wireless networks such as wireless mesh networks, radio-equipped nodes can communicate with their neighbors directly when they are within the radio transmission ranges. Two nodes that are away from each other may rely on intermediate nodes to relay traffic. In such networks, there are unique challenges in the wireless contexts. In particular, the wireless channel is a spatially shared resource. Wireless nodes within the interference range compete for the same wireless channel. Wireless link contention in such networks causes the link capacities to be time varying [9]. In order to achieve high end-to-end throughput in an efficient manner, network resources such as power can sometimes be allocated to change link capacities [25].

It is well known that the presence of wireless links can significantly affect the performance of end-to-end transport protocols [31]. Congestion control in today's Internet is based on an assumption that almost all packet losses result from congestion. Packet losses on wireless links that are from corruption rather than congestion violate this assumption. In order to effectively handle this situation, several schemes suitable for wireless networks have been proposed to either provide the sender with explicit information about congestion such as Explicit Congestion Notification (ECN) or shield effect of wireless losses on congestion control indication [14] such as Indirect-TCP, MTCP, WTCP, and Snoop.

A time varying link capacity can cause variation of inter-packet delay [31]. Those transport protocols that consider increased delay as an indication of congestion will be affected. More important, time-varying link capacities change the TCP dynamics, and the optimization solution to network utility maximization [25, 26].

1.3 Congestion Control in Intermittently Communicating Networks

In this section, we first compare the TCP based Internet with an important class of challenged networks (also termed delay tolerant networks or disruption tolerant networks), then explain why TCP breaks in these challenged networks, and finally describe the structure of delay tolerant networks and congestion control in such networks.

For the existing TCP/IP based Internet service model, a number of key assumptions, although often not explicitly stated, are made: an end-to-end path exists between a data source and its peer(s), the maximum round-trip time between any node pairs in the network is not excessive, and the end-to-end packet drop probability is small [47]. Unfortunately, a class of challenged networks, which may violate one or more of the assumptions, are becoming important and may not be well served by

the current end-to-end TCP/IP model. This class of challenged networks are qualitatively characterized by intermittent connectivity, long or variable delay, asymmetric data rates, and high error rates.

Challenged networks arise primarily as a result of various forms of host and node mobility, but may also come into being as a result of disconnection due to power management or interference [47, 69]. Examples of such networks include terrestrial mobile networks, military ad-hoc networks, sensor/actuator networks, deep space communication etc.

TCP is ill suited for operation over a path characterized by extremely long propagation delay, particularly if the path contains intermittent links. TCP communication requires that the sender and receiver negotiate a connection that will regulate the flow of data. Establishment of TCP connection typically entails at least one round-trip time (RTT) before any application data can flow. If transmission latency exceeds the duration of the communication opportunity, no application data will flow at all [68, 70]. There is a generic, two-minute timeout implemented in most TCP stacks: if no data is sent or received for two minutes, the connection breaks. However, in interplanetary networks such as Earth-Mars communication, the RTT is roughly eight minutes at Mars' closest approach to Earth, with a worst-case RTT of approximately 40 minutes. Thus, normal TCP can not work at all for Earth-Mars communication. The high delay also exists in some sparse sensor networking, where sensor readings aren't needed in real time.

Delay tolerant networks overcome the problems associated with intermittent connectivity, long or variable delay, asymmetric data rates, and high error rates by using *store-and-forward message switching*. Whole messages or pieces (fragments) of such messages are moved forward from a storage place on one node to a storage place on another node along a path that eventually reaches the destination. Store-

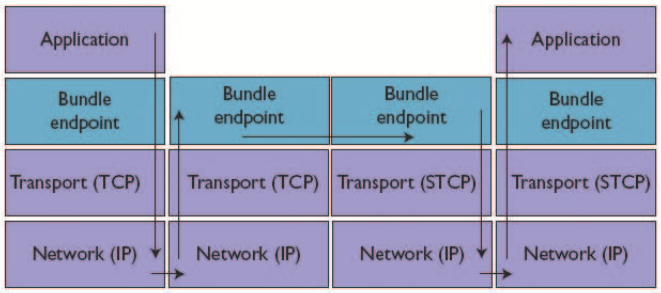


Figure 1.4: The overaly network approach

and-forward methods are similar to those methods used in today's voice mail and email systems. Storage places on nodes can hold messages indefinitely. These storage places are called persistent storage, opposed to the very short-term storage provided by memory chips in Internet routers. Internet routers use memory chips to store (queue) incoming packets for a few milliseconds while they are waiting for an available outgoing router port. The nodes in delay tolerant networks need persistent storage to store messages because a communication link to the next hop may not be available for a long time; a message, once transmitted, may need to be retransmitted if an error occurs at a downstream (toward the destination) node or link, or if a downstream node declines acceptance of a forwarded message. In order to move messages (or fragments thereof) forward, the message switching techniques provide the network nodes with immediate knowledge of the size of messages, and therefore the requirements for immediate storage space and transmission bandwidth.

Delay tolerant networks implement store-and-forward message switching by overlaying a new protocol layer, called the bundle layer, at the application layer or at least above the transport layer. Figure 1.4 shows the overlay network approach.

The bundle protocol is an example of what is generally called an overlay network, and can run on top of the current Internet protocol suite as well as the more

esoteric protocols for complex sensor networks, and other challenging environments. The protocol packages a unit of application data along with any required control information into a "bundle" that is similar to an email message. Nodes then forward this bundle along a path consisting of several intermediate nodes that each can store it for significant periods. Thus, the bundle protocol is an overlay network store-and-forward protocol. The bundle layer ties together low layers among networks so that application programs can communicate across networks.

The bundle protocol includes a way to transfer responsibility for retransmissions to another node. The node that is currently responsible for handling retransmission of some fragment of a bundle is called the *custodian* for that fragment. The bundle layer supports hop-by-hop retransmission by means of *custody transfers*. Such transfers are arranged between the bundles of successive nodes, at the initial request of the source application. When the current bundle layer custodian sends a bundle to the next node, it requests a custody transfer and starts a time-to-acknowledge retransmission timer. If the next hop bundle layer accepts custody, it returns an acknowledgement to the sender. If no acknowledgement is returned before the sender's time-to-acknowledge expires, the sender retransmits the bundle. A bundle custodian must store a bundle until either another node accepts custody or the bundle's time-to-live expires. Custody transfers don't provide guaranteed end-to-end reliability. The bundle layer uses a reliable transport layer protocol together with custody transfer to move points of retransmission progressively forward toward the destination. The advance of retransmission points minimizes the number of potential retransmissions, and the total time to convey a bundle reliably to its destination.

Since bundles have to traverse lower-layer networks, they are ultimately subject to whatever restrictions exist on those networks in terms of maximum packet sizes.

For example, on most IP networks it is safest to assume that single packets should be less than 1,500 bytes long.

Congestion control in delay tolerant networks refers to the handling of contention for the persistent storage at nodes of such networks. It is difficult to be implemented because contacts may not arrive for some time in the future, accumulated data may not have an opportunity to drain in the immediate future; received messages for which custody has been accepted can't be discarded except under extreme circumstances or on expiration. The current approach uses a priority queue for allocating custody storage. First, messages that are too large are denied custody transfer. Next, messages are spooled FCFS based on priority. The potential problem arising from this approach is that arriving higher priority messages may not have custody storage available if lower priority messages arriving earlier have been custodially received.

In [47], Fall proposed two potential mechanisms to deal with congestion, i.e., proactive or reactive methods. Proactive methods generally involve some form of admission control to avoid the onset of congestion in the first place. In many cases, a single region may be under the administrative control of a single entity. For the reactive method, the possible schemes include reserving buffer space as a function of custody space, rejecting incoming connections for new messages when buffer space is full, arranging for custody transfers to other potential custodians that may not be the most desirable next hop and discarding non-custody bundles in favor of any bundles requiring custody transfers. The reactive method will result in degraded performance.

In all, the above delay tolerant network architecture aims to provide interoperable communications between and among a wide range of networks which may have exceptionally poor and disparate performance characteristics. The design embraces message switching similar to today's voicemail and email systems. Interestingly, delay

tolerant networks can be overlaid upon the TCP/IP based Internet easily, and tie together dramatically different types of networks with unusual connectivity properties [47].

1.4 Repeated Game and Network Resource Allocation

In this section, we first introduce game theory, then briefly describe repeated game theory and its applications in network resource allocation.

1.4.1 Basics of Game Theory

Game theory aims to model situations in which multiple players interact or affect each other's outcomes [84]. It provides a rich set of tools for understanding how such players may desire to act in practice and how the rules and structure of the environment can impact their behaviors, and has been applied to study networking problems in an attempt to build more efficient and robust systems [84]. In order to introduce game theory, we start by describing what is perhaps the most well known and well-studied game below [83, 84].

Prisoners' Dilemma Two prisoners are on trial for a crime and each one faces a choice of confessing to the crime or remaining silent. If they both remain silent, the judge will not able to charge them and they will get short term penalties, say 2 years, for minor offenses. If only one of them confesses, his term will be 1 year and he will be used as a witness against the other, who in turn will get a sentence of 5 years. If they both confess, they both will get a sentence of 4 years.

Clearly, there are a total of four outcomes depending on the choices made by each of the two prisoners. We can succinctly summarize the payoff incurred in these outcomes via Table 1.1. Each of the two prisoners has two possible strategies "Confess" or "Silent". The two strategies of prisoner P1 correspond to the two rows

Table 1.1: Payoff matrix of Prisoners' Dilemma

		P2	
P1		Confess	Silent
	Confess	$(4, 4)$	$(1, 5)$
	Silent	$(5, 1)$	$(2, 2)$

and the two strategies of P2 correspond to two columns of the matrix. The entries of the matrix are the payoffs incurred by the players in each situation. Such a matrix is called payoff matrix because it contains the payoff incurred by the players for each choice of their strategies. The only stable solution in this game is that both prisoners confess.

Formally, a game consists of a set of players, $\{1, 2, \ldots, n\}$. Each player i has his own *set of possible strategies*, say S_i. To play the game, each player i selects a strategy $s_i \in S_i$. We will use $s = (s_1, \ldots, s_n)$ to denote the *vector of strategies* selected by the players and $S = \Pi_i S_i$ to denote the set of all possible ways in which players can pick strategies.

The vector of strategies $s \in S$ selected by the players determines the outcome for each player; in general, the outcome will be different for different players. To specify the game, we need to give, for each player, a preference over the set of all strategy vectors S. The simple way to specify a preference is by assigning, for each player, a value to each outcome. In some games values will be payoffs to the players and in others the costs incurred by the players. We can denote these functions by $u_i : S \to \mathbf{R}$ and $c_i : S \to \mathbf{R}$, respectively. Clearly, costs and payoffs can be used interchangeably, since $u_i(s) = -c_i(s)$.

The Prisoners' Dilemma game has a very special property: each player has a unique best strategy, independent of the strategies played by the other players. We say that a game has a *dominant strategy solution* if it has this property.

More formally, for a strategy vector $s \in S$ we use s_i to denote the strategy played by player i and s_{-i} to denote the $(n-1)-$ dimensional vector of the strategies played by all other players. We use $u_i(s)$ to denote the payoff (or cost) incurred by player i. We also use the notation $u_i(s_i, s_{-i})$ when it is more convenient. Using this notation, a strategy vector $s \in S$ is a *dominant strategy solution*, if for each player i, and each alternate strategy vector $s' \in S$, we have that

$$u_i(s_i, s'_{-i}) \geq u_i(s'_i, s'_{-i})$$

Having a single dominant strategy for each player is an extremely stringent requirement for a game and very few games satisfy it.

Since games rarely possess dominant strategy solutions, we need to seek a less stringent and more widely applicable solution concept. A desirable game-theoretic solution is one in which individual players act in accordance with their incentives, maximizing their payoff. This idea is best captured by the notion of a Nash equilibrium, which has emerged as the central solution concept in game theory.

A strategy vector $s \in S$ is said to be a *Nash equilibrium* if for all players i and each alternate strategy $s'_i \in S_i$, we have that

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}).$$

In other words, no player i can change his chosen strategy from s_i to s'_i and thereby improve his payoff, assuming that all other players stick to the strategies they have chosen in s. Observe that such a solution is self-enforcing in the sense that once the players are playing such a solution, it is in every player's best interest to stick to his or her strategy.

Clearly, a dominant strategy solution is a Nash equilibrium. Moreover, if the solution is strictly dominating, it is also the unique Nash equilibrium.

The detailed discussion of game theory is referred to the excellent reference [83].

1.4.2 Repeated Game and Resource Allocation

Informally, repeated game theory considers an interaction not as a single play but rather a sequence of similar plays occurring over time, with the actions in one period potentially impacting the state of the world and the actions of players in future periods.

Repeated game theory is an appropriate tool for modeling network resource allocation based on the following reasons [83, 85].

1. **Repetition is an inherent aspect of almost all networked problems.** Routing and congestion control are examples of plays which are constantly repeated in similar environments. Individual players repeatedly interact with the same networks, often to accomplish the same or a similar set of tasks. Further, in P2P networks, individuals repeatedly interact with the same or behaviorally similar individuals to share files.
2. **Repeated game theory is a well understood dynamic game theory.** The best-understood class of dynamic games is that of repeated game. The previous research provides appropriate tools and concepts for the analysis of repeated game. These concepts have been shown to be robust to a wide array of practical assumptions in networking and these dynamics have been observed and documented in practice.
3. **Repetition can significantly alter the outcome of a game.** The outcome of a repeated game can differ from the outcome of the particular stage game. If the players' actions are observed at the end of each period, it becomes possible

for players to condition their play on the past play of their opponents, which can lead to equilibrium outcomes that don't arise when the game is played only once. One example of this in the repeated prisoner's dilemma is the "unrelenting" strategy — cooperate until the opponent defects; if ever the opponent defects, then defect in every subsequent period.

The prevalence of repetition in networking suggests that repeated games are appropriate models for networked applications.

1.5 Motivation of Research

In this dissertation, we explore congestion control strategies in networks with challenging environments, i.e., networks with time varying link capacity and intermittently communicating networks.

Optimization based congestion control approaches have been extensively used over the past several years to study resource allocation problems in wireline networks, and such approaches have resulted in a deep understanding of the ubiquitous Transmission Control Protocol (TCP) and resulted in improved solutions for congestion control mechanisms in wireline networks. Unfortunately, since there exist fundamental differences between wireless networks with time varying link capacity and traditional wireline networks, the significant differences prevent verbatim applications of the existing congestion control mechanisms in wireline networks to the situations in wireless networks [9, 30].

In wireline networks, the link capacity is usually known and fixed, the equilibrium of the distributed primal or dual solutions to a convex optimization problem that maximizes the aggregate system performance (or utility) is fixed. The analysis of stability of optimal solution focuses on the unique equilibrium point. However wireless environments can change the link capacities, therefore change the dynamics

of congestion control mechanisms, and the optimal solution to network utility maximization. In this regard, the approaches developed for traditional wireline networks can not be directly applied to these challenging problems. We have to seek t solutions suitable for the ever changing environments, i.e., the formulation of network utility maximization developed for traditional wireline networks should be adapted to time varying link capacities, the control scheme should be able to adapt to the changing conditions, stability of congestion control algorithms should be guaranteed even if there exists perturbation of link capacity on the congestion control systems.

In intermittently communicating networks, while custody transfers provide a technique to handle congestion, it is still very unclear how one can set up a network so that custody transfers will actually solve the likely congestion. In [47], Fall proposed two potential congestion control schemes, unfortunately, they are only in the conceptual stage; how to develop applicable congestion control algorithms based on these concepts needs a lot of research.

For intermittently communicating networks, there doesn't exist an end-to-end path, the networks are inherently dynamic, the global information is not available, two nodes can only communicate during opportunistic contacts. The convex optimization based congestion control algorithms developed for the Internet can not be applied to such networks. Therefore, an alternative optimization approach is necessary to effectively avoid congestion and optimize the network resources.

Whether or not delay tolerant networks can handle congestion as applications are deployed will be a good indicator for the success of this technology. Thus, it is imperative that effective congestion control algorithms be developed to optimize the network resources.

Game theory has long been used to model the routing decisions of networks. However, once we move to dynamic and resource constrained settings, such as delay

tolerant networks, traditional models are no longer sufficient. Instead, new models that capture the dynamic nature of the decisions and the resource constraints of the networks are needed. In intermittently communicating networks like delay tolerant networks, there exist situations in which the number of rounds is finite, but there is no knowledge when the game is going to stop. Every node can not be sure that it is going to play the next round with different opponents since the communication is intermittent. Since the repeated game can capture this kind of dynamic behaviors, it will be beneficial to apply repeated game theory to study the congestion control problem in delay tolerant networks.

1.6 Main Contributions and Organization

Chapter 2 explores congestion control mechanisms for a class of networks with time varying link capacities. We explicitly model link capacities to be time varying and investigate congestion control problems based on the convex optimization approach. Since the link capacity is time varying, the optimization solution of the congestion control algorithm is not a unique equilibrium point, it is a reference trajectory. Correspondingly, we introduce trajectory stability instead of stability around a single equilibrium point, and prove that the proposed primal-dual congestion control algorithm is trajectory stable in the absence of feedback delay. Moreover, we obtain sufficient conditions for the scheme to be locally stable in the presence of delay. Our key technique is to model time variations of capacities as perturbations to a constant link. Furthermore, to study the robustness of the algorithm against capacity variations, we investigate the sensitivity of the control scheme and through simulations study the tradeoff between stability and sensitivity.

In Chapter 3, we study congestion control mechanism for intermittently communicating networks, where end-to-end path may not exist, propagation delay is

excessively large, TCP breaks. We apply the concept of revenue management, and employ dynamic programming to develop a congestion management strategy for this class of networks. For a class of network utility functions, we show that our solution is optimal. More importantly, our solution is distributed where only the local information such as available buffer of a node is required. This is particularly important given the nature of the intermittently communicating networks where global information is often not available and the network is inherently dynamic. Our simulation results show that the proposed congestion management scheme is effective in avoiding congestion and balancing network load among the nodes.

In Chapter 4, we employ repeated games to study congestion control mechanism for the intermittently communicating networks such as delay tolerant networks. Repeated game model can effectively capture the dynamic behaviors happening in delay tolerant networks, especially the situations where there is no the knowledge when the dynamic behavior is going to stop. The proposed congestion control strategy is completely distributed where only the local information of a node is required. The control strategy can avoid complicated computation when dynamic programming techniques are applied to infinite time horizon.

Chapter 5 concludes this dissertation.

CHAPTER 2
CONGESTION CONTROL IN NETWORKS WITH TIME VARYING LINK CAPACITIES

2.1 Introduction

Edging toward wide deployment, a critical challenge facing networks with time varying link capacities such as wireless multihop networks is the design and development of transport protocols that can simultaneously guarantee high bandwidth utilization and fairness across multiple users [2]. The unique characteristics underlying such networks are essentially two-fold: the potentially high bandwidth and larger number of flows, and the time varying link capacities over the multi-hop environment.

Indeed, extensive related work exists in the literature. First, the inadequacy of TCP when facing the exploding Internet bandwidth has promoted extensive research toward new congestion control schemes targeted at high utilization, low queueing delay, and fairness [18]. Owing to the seminal work by Kelly [5], congestion control has mainly been formulated as utility maximization problems and distributed control theory based solutions have been devised accordingly [8, 10, 18, 23, 24, 36]. The proposed schemes generally consist of two components: a source algorithm that adjusts sending rate in response to congestion in its path, and a link algorithm that updates a congestion measure and feeds it back, implicitly or explicitly, to the sources utilizing the link.

In parallel, intensive research efforts have been devoted to designing congestion control algorithms capable of accommodating error-prone and time varying wireless links, which have been demonstrated to be well beyond normal TCP's reach [31]. While earlier approaches, such as I-TCP and Snoop-TCP [14], have mainly been

engineered based on empirical techniques, recent efforts have embraced the above optimization and control theory based approach [25, 27–29]. For example, a hop-by-hop congestion control scheme is proposed specifically for wireless networks in [28], and various performance metrics to be maximized in wireless networks are studied in [27].

Surprisingly, these congestion schemes developed for wireless networks often have assumed constant link capacities (or a fixed portion of a certain constant bandwidth). While such assumptions suit wireline networks comfortably, it certainly will limit the application scopes of the proposed algorithms in the wireless domain. It is well known that wireless channels are characterized by inherent time-varying capacities that have been shown to significantly reduce the throughput of TCP [14, 15]. The failure of TCP in such a scenario is the consequence of its inability to distinguish packet loss caused by flow congestion or link errors (or equivalently, reduced link bandwidth). Using constant capacities to model wireless links can not fully capture this effect and in particular that on congestion control algorithms. While an optimal congestion control scheme in conjunction with power control has been developed for multi-hop wireless networks considering time-varying link capacities, the requirement of knowledge of network-wide interference limits its practicability [25].

In this chapter, we explore congestion control algorithms in networks with time varying link capacities, and use multi-hop wireless networks such as wireless mesh networks as application examples. We propose a primal-dual congestion control algorithm and prove it to be trajectory stable in the absence of feedback delay. Different from existing works that can only guarantee system equilibrium at a single point, we show that the system is stable around a sequence of time-indexed equilibrium points that in turn jointly form a time varying reference trajectory. Moreover, by modeling capacity variation as perturbation to a fixed channel, we further obtain sufficient

conditions for the primal-dual scheme to be locally stable. Furthermore, to study the robustness of the algorithm against capacity variations, we investigate the sensitivity of the control scheme. Using tractable scenarios, we demonstrate that local stability and system sensitivity in the presence of feedback delay can achieve a balancing tradeoff by tuning the gain of source controllers. Through extensive experimental studies, we show that the algorithm excels in a wide variety of system setups and investigate the effects of different parameters on system stability and sensitivity.

This chapter is organized as follows. In Section 2.2, we motivate our work. In Section 2.3, we further examine the convex optimization based framework of the congestion control mechanism. In Section 2.4, we present several definitions of stability as background. In Section 2.5, we prove that in the absence of feedback delay, the algorithm is trajectory stable. When feedback delay is present, we derive in Section 2.6 sufficient conditions for the system to be locally stable. Subsequently, system sensitivity with respect to link capacity perturbation is analyzed in Section 2.7. Section 2.8 describes our experimental studies. Finally, we conclude in Section 2.9.

2.2 Motivation

Congestion control in the wireline domain has attracted tremendous research interests owing to the inadequacy of TCP when facing high bandwidth-delay product. While extensive work [5, 8, 10–12, 18, 20, 21, 23] has been done on congestion control thereafter, a common assumption is that the capacity of a link is fixed, i.e., the link capacity is not time varying. This assumption surely is valid for wireline networks. Wireless links, on the contrary, are characterized by time variations. Such variations are direct results of changes in signal to noise ratio (SNR), which in turn are caused by the mobility of wireless devices and/or fluctuations of the surrounding physical environment [15]. While adaptive modulation and coding schemes can be employed

at the physical layer, they only target a desired bit/frame error probability at the cost of varying transmission rate. Indeed, even for wireline networks, bandwidth can vary due to various reasons. For example, due to link sharing, a router may only have access to a portion of the bandwidth which can fluctuate over time [46].

It is pointed out in [31] that variable bandwidth is one of the intrinsic characteristics that affect the performance of transport protocols in wireless networks. However, the authors only qualitatively address how bandwidth variations affect the system performance, no formal analysis regarding this was presented. A distributed hop-by-hop congestion control scheme is developed for multi-hop wireless networks in [28]. The scheme is shown to be stable in the absence of round trip propagation delay. However, the authors assume that channel variations can be effectively masked by physical layer coding and modulation schemes and hence can be considered as a "constant channel" at higher layers.

For research on active queue management, the authors proposed an Adaptive Virtual Queue(AVQ) algorithm [21]. Although the AVQ scheme can adaptively change the virtual link capacity to get high utilization via a predefined dynamic law, its dynamic behavior is different from that owing to time-varying wireless channels. The time varying link capacity in wireless networks can not be described by a deterministic differential equation and hence the research results for AVQ cannot be directly applied to design congestion control mechanisms for networks with time varying link capacities.

Time varying link capacities can change the TCP dynamics, and the optimal solution to network utility optimization. Unfortunately, little consideration is given to congestion control dynamics caused by link capacity variations. Thus, we have to seek the solutions suitable for the ever changing environments. Specifically, the formulation of network utility maximization developed for traditional wireline networks should

be adapted to time varying link capacities, the control scheme should be able to adapt to the changing conditions, stability of congestion control algorithms should be guaranteed even if there exist perturbations of link capacities on the congestion control systems.

2.3 Problem Formulation and Algorithm Design

2.3.1 Preliminary

Triggered by the seminal work by Kelly, congestion control has then been developed and analyzed as distributed algorithms solving appropriately formulated utility maximization problems [5]. Intuitively, consider a wireline communication network with L links, each with a fixed capacity of c_l, and S sources with transmission rates of x_s for $s \in S$. Assume that each source s uses a fixed set $L(s)$ of links to route its traffic through and possesses an increasing, strictly concave, and twice differential utility function $U_s(x_s)$. A congestion control scheme can be formulated as to maximize the total utility $\sum_{s \in S} U_s(x_s)$ over the source rates $\{x_s, s \in S\}$, subject to the constraints of total link capacity, i.e., $\sum_{s:l \in L(s)} x_s \leq c_l$ for all links. Formally, the optimization problem is

$$\max \quad \sum_{s \in S} U_s(x_s) \tag{2.1}$$

$$\text{s.t.} \quad \sum_{s:l \in L(s)} x_s \leq c_l, \forall\, l \in L \tag{2.2}$$

$$x_s \geq 0, s \in S. \tag{2.3}$$

Corresponding to the above objective, distributed solutions allowing individual sources to adjust their transmitting rates generally have taken the following form:

$$\dot{x}_s(t) = f\left(x_s(t), \sum_{l \in L(s)} p_l(t)\right) \tag{2.4}$$

$$\dot{p}_l(t) = g\left(p_l(t), \sum_{s \in S(l)} x_s(t)\right). \tag{2.5}$$

Here $p_l(t)$ denotes the price of link l that may correspond to link congestion, loss probability, etc. $f(\cdot)$ and $g(\cdot)$ are the control functions for updating the transmission rate and price at the source and link, respectively. A unique equilibrium can be derived as the solution for the utility maximization problem for Equations (2.4) and (2.5) under certain conditions. Actually the above control based solutions are termed primal-dual algorithm, as differential equations are used at both the sources and links for updating. On the contrary, if a static function is employed at the links to generate congestion signal, it is termed primal algorithm and if the sources use static functions to regulate packet rates, it is termed dual algorithm.

Before formally formulating our problem, we remark that we assume that link capacities in networks with time varying link capacities are independent of each other. Indeed this assumption represents a certain degree of simplification of reality. However, eliminating the interference allows us to much more clearly understand the effect of the link variations on the congestion control scheme. Indeed this assumption is not without its own real applications. For example, by using different codes (CDMA), frequencies (FDMA), or time allocation (TDMA), non-interfering wireless links can be achieved in multihop wireless networks such as mesh networks. In reality, OFDM and MIMO systems, such as Nortel's wireless mesh network products, have enabled wireless nodes to engage in simultaneous communication on multiple channels at the

same time. Furthermore, with the increasing application of directional antenna, non-interfering links can also be created in the spatial domain. For works employing similar assumptions, we refer to [15].

2.3.2 Problem Formulation

For completeness, we rephrase congestion control for utility maximization but in an all wireless network (wireline links with fixed capacities can be considered a special case). Given the set of wireless links L and the set of traffic sources $\mathcal{S}$. Each source $s \in \mathcal{S}$ identifies a unique source-destination pair and correspondingly a flow between them. Associated with each source is a route r composed of a subset $\{l\} \subset L$ of links. If route r uses link l, we write $l \in r$. Let R be the set of routes. The routing matrix $\mathcal{R}$, of dimension $|L| \times |\mathcal{S}|$, is thus defined by

$$\mathcal{R}_{lr} = \begin{cases} 1 & \text{if route } r \text{ uses link } l \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Let $x_r \geq 0$ be the flow (source sending) rate associated with route $r \in R$ and $c_l(t)$ be the *time-varying "capacity"* of link $l \in L$. As each link l may be used by several routes, let y_l be the total arrival rate of traffic on logical link l. Then, the vector of link rates $\mathbf{y}$ is given by the relationship $\mathbf{y} = \mathcal{R}\mathbf{x}$, where $\mathbf{y} = (y_l, l \in L)$ and source rate $\mathbf{x} = (x_r, r \in R)$ are both column vectors. For the utility function associated with each source, we restrict ourselves to weighted proportionally fair utility functions of the form $U_r(\cdot) = w_r \log(\cdot)$, where w_r is the weight for flow r. This function has been

shown to be particularly suitable for wireless networks [27]. The congestion control problem for utility maximization can then be summarized as

$$\max \quad \sum_{r \in R} w_r \log x_r \tag{2.7}$$

$$s.t. \quad \sum_{r:l \in r} x_r \leq c_l(t), \quad \forall\, l \in L, x_r \geq 0, r \in R. \tag{2.8}$$

Again, the key difference in our problem formulation lies in the right side of Equation (2.8), namely the time varying channel capacity. Correspondingly, the congestion control algorithm must be capable of accommodating the fluctuations while maintaining system stability and optimality. Towards this end, we define a capable primal-dual algorithm below.

2.3.3 Congestion Control Algorithm

Define Lagrangian function

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}) &= \sum_{r \in R} w_r \log x_r - \sum_{l \in L} \lambda_l \left(\sum_{r:l \in r} x_r - c_l(t) \right) \\ &= \sum_{r \in R} \left(w_r \log x_r - x_r \sum_{l \in r} \lambda_l \right) + \sum_{l \in L} \lambda_l c_l(t) \end{aligned} \tag{2.9}$$

where $\lambda_l (l \in L)$ is the Lagrangian multiplier. By differentiating Equation (2.9) with respect to x_r, we have

$$\frac{\partial L}{\partial x_r} = \frac{w_r}{x_r} - \sum_{l \in r} \lambda_l = 0 \tag{2.10}$$

From the above equation, we get

$$x_r = \frac{w_r}{\sum\limits_{l \in r} \lambda_l} \tag{2.11}$$

We remark that in Equations (2.9)-(2.11), if t is given, then $c_l(t)$ is fixed and bounded. The problem will degenerate to the one presented in [5] and solutions proposed therein hence can be employed. However, as we will show later, the time-varying characteristic of the link capacity will challenge us to explore new techniques for a stability proof and furthermore, sensitivity study.

Let $x_r(t)$ denote the flow rate of route r at time t. We define the source rate controller (primal algorithm) that adapts its rate according to the following differential equation

$$\dot{x}_r(t) = k_r \left(w_r - x_r(t) \sum_{l \in r} \lambda_l \right) \tag{2.12}$$

where λ_l, the Lagrangian multiplier, can also be considered as the link price of link l. Although Equation (2.12) is analogous in shape to those developed for wireline networks [5], the key difference dwells in λ_l, which now is not only determined by the aggregate rate $y_l(t)$ on link l, but also affected by the variations of link capacity $c_l(t)$. This is further elaborated by the dual algorithm for price updating on each link given by [18]

$$\dot{\lambda}_l(t) = h_l(\lambda_l(t)) \left[y_l(t) - c_l(t) \right]^+_{\lambda_l} \tag{2.13}$$

Here, $h_l(\lambda_l(t))$ is a non-decreasing continuous function in a generic form used for price updating. Specific functions can be determined for different purposes. $[y_l(t) - c_l(t)]^+_{\lambda_l}$ is defined as

$$[y_l(t) - c_l(t)]^+_{\lambda_l} = \begin{cases} y_l(t) - c_l(t) & \text{if } \lambda_l > 0, \\ \max(y_l(t) - c_l(t), 0) & \text{if } \lambda_l = 0. \end{cases}$$

Before proceeding to the analysis of this congestion control scheme, we recapitulate our motivation. If t is fixed, $c_l(t)$ is a constant and our model can be reduced to the standard model as stated in (2.1), which has been shown to possess a unique

optimum to the optimization problem. However, since $c_l(t)$ is time-varying, the optimum to (2.7) is not unique. Instead, the optimum is time varying as well. Our key objective in this chapter is thus to prove the stability and optimality of the above proposed congestion algorithm, even when coping with time varying link capacities.

2.4 Several Definitions of Stability

In order to better understand the analysis of stability in Section 2.5, we briefly present several definitions of stability in this section. The detailed discussion of stability of dynamic systems is referred to [39, 40, 42].

This section is concerned with differential equations of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t), \;\; \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.14}$$

where $\mathbf{x} \in \mathbb{R}^n$, $t \geq 0$.

The system defined by (2.14) is said to be *autonomous*, or *time-invariant*, if f does not depend on t, and non autonomous, or *time-varying*, otherwise.

Without loss of generality, we will always assume that $\mathbf{f}(\mathbf{x}(t), t)$ satisfies $\mathbf{f}(\mathbf{0}, t) = \mathbf{0}$ and study the stability of the origin $\mathbf{x} = \mathbf{0}$. $\|\cdot\|$ stands for the Euclidean norm on a real field with appropriate dimension.

Definition 2.1 Stability in the sense of Lyapunov

$\mathbf{x} = \mathbf{0}$ *is called a* stable *equilibrium point of* (2.14)*, if, for all* $t_0 \geq 0$ *and* $\varepsilon > 0$*, there exists* $\delta(t_0, \varepsilon) > 0$ *such that*

$$\|\mathbf{x}_0\| < \delta(t_0, \varepsilon) > 0 \Rightarrow \|\mathbf{x}(t)\| < \varepsilon, \;\; \textit{for all } t \geq t_0$$

where $\mathbf{x}(t)$ *is the solution of* (2.14) *starting from* $\mathbf{x}_0$ *at* t_0*.*

Definition 2.2 Uniform Stability

$\mathbf{x} = \mathbf{0}$ *is called a* uniformly stable *equilibrium point of* (2.14) *if, in the preceding definition,* δ *can be chosen independent of* t_0.

Definition 2.3 Asymptotic Stability

$\mathbf{x} = \mathbf{0}$ *is called an* asymptotically stable *equilibrium point of* (2.14)*, if*

1. $\mathbf{x} = \mathbf{0}$ *is a stable equilibrium point of* (2.14),
2. $\mathbf{x} = \mathbf{0}$ *is attractive, that is, for all* $t_0 \geq 0$*, there exists* $\delta(t_0)$*, such that*

$$\|\mathbf{x}_0\| < \delta \Rightarrow \lim_{t\to\infty} \|\mathbf{x}(t)\| = 0$$

that is, there exists $\delta > 0$*, and for every* $\varepsilon > 0$ *there exists a* $T(\varepsilon) > 0$ *such that* $\|\mathbf{x}(t_0)\| < \delta$ *implies that* $\|\mathbf{x}(t)\| < \varepsilon$ *for all* $t \geq T + t_0$

Definition 2.4 Global Asymptotic Stability

$\mathbf{x} = \mathbf{0}$ *is called a* globally asymptotically stable *equilibrium point of* (2.14)*, if it is asymptotically stable and* $\lim_{t\to\infty} \|\mathbf{x}\| = 0$ *for all* $\mathbf{x}_0 \in \mathbb{R}^n$.

It is worth noting that the difference between asymptotic stability and globally asymptotic stability lies in how to choose the initial value of $\mathbf{x}_0$. For asymptotic stability, $\mathbf{x}_0$ can only be in a closed ball of radius h centered at $\mathbf{0} \in \mathbb{R}^n$ (h is a small positive real number). For globally asymptotic stability, $\mathbf{x}_0$ can be in anywhere in $\mathbb{R}^n$, that is, $\mathbf{x}_0 \in \mathbb{R}^n$.

The above definitions of stability of dynamic system (2.14) focus on a unique equilibrium point. We now turn to the trajectory stability.

In trajectory stability, a reference trajectory $\mathbf{x}^*(t)$ is employed instead of a single equilibrium point $\mathbf{0}$. $\mathbf{x}^*(t; \mathbf{x}_0^*, t_0)$ is deemed trajectory stable if for all t_0 and $\varepsilon > 0$, there exists $\delta(\varepsilon, t_0) > 0$ such that $\|\mathbf{x}(t; \mathbf{x}_0, t_0) - \mathbf{x}^*(t; \mathbf{x}_0^*, t_0)\| < \varepsilon$ for all $t \geq t_0$ if $\|\mathbf{x}_0 - \mathbf{x}_0^*\| < \delta$. $\mathbf{x}^*(t; \mathbf{x}_0^*, t_0)$ is said to be asymptotically stable if it is stable and

convergent, where convergence roughly requires that for any t_0 there exists a $\delta_1(t_0)$ such that $\|\mathbf{x_0} - \mathbf{x}_0^*\| < \delta_1$ implies that $\lim_{t\to\infty} \|\mathbf{x}(t; \mathbf{x}_0, t_0) - \mathbf{x}^*(t; \mathbf{x}_0^*, t_0)\| = 0$.

Figure 2.1 illustrates the equilibrium point (x_e) stability in plane space. Figure 2.2 illustrates asymptotically stable equilibrium point. Figure 2.3 illustrates the trajectory stability. The major difference between equilibrium point stability and trajectory stability is that for trajectory stability, equilibrium point is time varying, the equilibrium points form a reference trajectory.

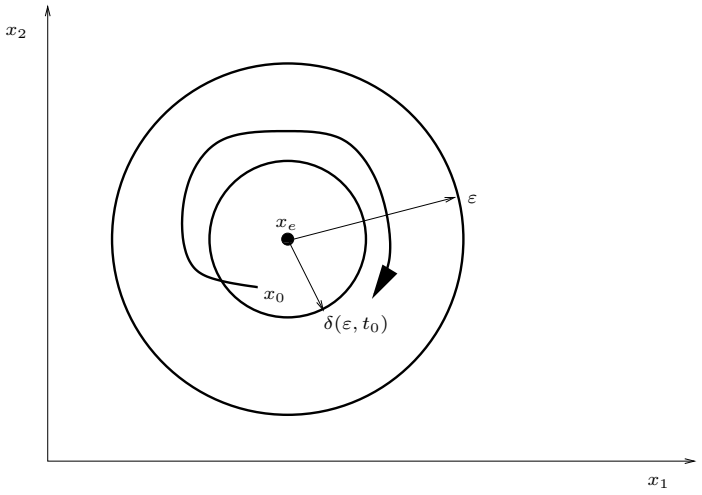


Figure 2.1: Equilibrium point stability

Interested readers are referred to [40] for detailed discussions on trajectory stability.

2.5 Stability Analysis without Delay

If link capacities are constant, the unique system equilibrium resides on a single point, which indeed is guaranteed by extensive designs [8, 18]. On the contrary, if the link capacities are time varying, the equilibrium of the system becomes dependent on time t. In other words, the equilibrium $\bar{x}_r(t)$ of the system is a curve rather than

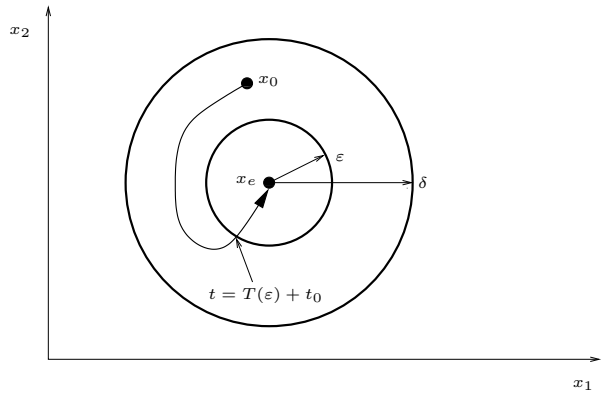


Figure 2.2: Asymptotically stable equilibrium point

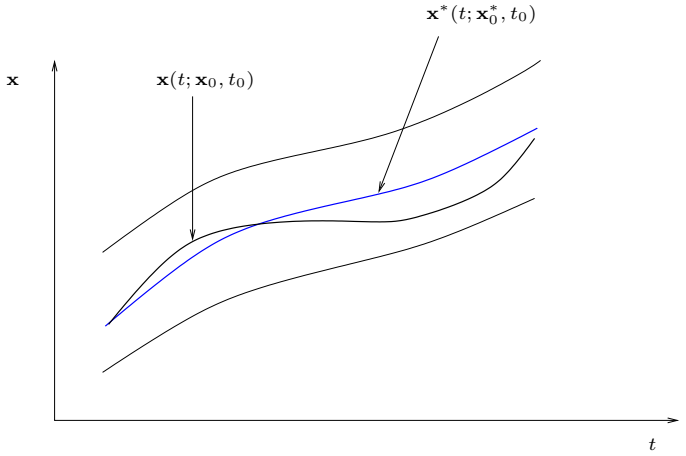


Figure 2.3: Trajectory-stability illustration

a point, as it is varying with t. In this section, we adopt the concept of trajectory stability to investigate this time varying system stability without considering propagation time delay, and show that our proposed congestion control scheme is trajectory stable.

The goal of the congestion control algorithm is to force the actual trajectory of the system to follow the reference one. In other words, the actual trajectory should stay near the reference trajectory at all times. Since the link capacities are time-varying, their values can not be known in advance. In this case, an open-loop control law will never reach the desired goal. Fortunately, the congestion control

scheme introduces feedback into the control law. This feedback can guarantee that the actual trajectory converges to the reference trajectory provided that the primal-dual algorithm is properly designed. Notably, if the link capacities are constant, the reference trajectory degenerates into a single equilibrium point. In this case, the primal-dual algorithm is asymptotically stable around the equilibrium point. This degenerated case actually corresponds to the wireline case where link capacities are fixed.

Following the above argument, when the system delay is zero, we have the following theorem.

Theorem 2.1 *The proposed primal-dual algorithm is asymptotically stable in the name of trajectory-stability.*

Proof: Denote the optimal transmission rate of user r as $\bar{x}_r(t)$. Let $x_r(t)$ be any other rate that satisfies the constraints in (2.7) at time t. Let $\bar{\lambda}_l(t)$ and $\lambda_l(t)$ be the corresponding link prices for $\sum_{r:l\in r} \bar{x}_r(t)$ and $\sum_{r:l\in r} x_r(t)$ respectively. Let $\mathbf{x}(t) = (x_r(t), r \in R)$. For simplification, define $q_r(t) = \sum_{l\in r} \lambda_l(t)$, i.e., $q_r(t)$ is the sum of the prices of all links on route r at time t. Let $\mathbf{q}(t) = (q_r(t), r \in R)$, and $\boldsymbol{\lambda}(t) = (\lambda_l(t), l \in L)$. From the definition of the routing matrix, we have $\mathbf{q}(t) = \mathcal{R}^T\boldsymbol{\lambda}(t)$, where T stands for the transpose of a matrix. As routing matrix $\mathcal{R}$ is usually required to be of full row rank, given $\mathbf{q}(t)$, there exists a unique $\boldsymbol{\lambda}(t)$ such that $\mathbf{q}(t) = \mathcal{R}^T\boldsymbol{\lambda}(t)$.

We construct the following Lyapunov function,

$$\begin{aligned} V(\mathbf{x}(t), \boldsymbol{\lambda}(t); t) = &\sum_{r\in R} \int_{\bar{x}_r(t)}^{x_r(t)} \frac{1}{k_r\sigma}(\sigma - \bar{x}_r(t))d\sigma \\ &+ \sum_{l\in L} \int_{\bar{\lambda}_l(t)}^{\lambda_l(t)} \frac{1}{h_l(\beta)}(\beta - \bar{\lambda}_l(t))d\beta \end{aligned} \tag{2.15}$$

We remark that the Lyapunov function we have chosen is very similar to the one presented in [11]. However, the key difference from [18] is that, the constructed Lyapunov function is an explicitly time dependent function. The integral scope is time dependent in our case.

Taking the derivative on both sides of Equation (2.15), we have Equation (2.16) and subsequently (2.17) and (2.18).

$$\begin{aligned}
\frac{dV}{dt} &= \sum_{r\in R}\frac{1}{k_r x_r(t)}(x_r(t)-\bar{x}_r(t))\dot{x}_r(t) + \sum_{l\in L}\frac{1}{h_l(\lambda_l(t))}(\lambda_l(t)-\bar{\lambda}_l(t))\dot{\lambda}_l(t) \\
&= \sum_{r\in R}(x_r(t)-\bar{x}_r(t))\left(\frac{w_r}{x_r(t)} - \sum_{l\in r}\lambda_l(t)\right) + \sum_{l\in L}(\lambda_l(t)-\bar{\lambda}_l(t))(y_l(t)-c_l(t))^+_{\lambda_l(t)} \\
&= \sum_{r\in R}(x_r(t)-\bar{x}_r(t))\left(\frac{w_r}{x_r(t)} - q_r(t)\right) + \sum_{l\in L}(\lambda_l(t)-\bar{\lambda}_l(t))(y_l(t)-c_l(t))^+_{\lambda_l(t)} \\
&\le \sum_{r\in R}(x_r(t)-\bar{x}_r(t))\left(\frac{w_r}{x_r(t)} - q_r(t)\right) + \sum_{l\in L}(\lambda_l(t)-\bar{\lambda}_l(t))(y_l(t)-c_l(t)) \\
&= \underbrace{\sum_{r\in R}(x_r(t)-\bar{x}_r(t))(\bar{q}_r(t)-q_r(t)) + \sum_{l\in L}(\lambda_l(t)-\bar{\lambda}_l(t))(y_l(t)-\bar{y}_l(t))}_{\text{Sum 1}} \\
&\quad + \underbrace{\sum_{r\in R}(x_r(t)-\bar{x}_r(t))\left(\frac{w_r}{x_r(t)} - \bar{q}_r\right) + \sum_{l\in L}(\lambda_l(t)-\bar{\lambda}_l(t))(\bar{y}_l(t)-c_l(t))}_{\text{Sum 2}}
\end{aligned} \tag{2.16}$$

$$\begin{aligned}
\text{Sum 1} &= \sum_{r\in R}(x_r(t)-\bar{x}_r(t))(\bar{q}_r(t)-q_r(t)) + \sum_{l\in L}(\lambda_l(t)-\bar{\lambda}_l(t))(y_l(t)-\bar{y}_l(t)) \\
&= (\bar{\mathbf{q}}(t)-\mathbf{q}(t))^T(\mathbf{x}(t)-\bar{\mathbf{x}}(t)) + (\boldsymbol{\lambda}(t)-\bar{\boldsymbol{\lambda}}(t))^T(\mathbf{y}(t)-\bar{\mathbf{y}}(t)) \\
&= (\bar{\boldsymbol{\lambda}}(t)-\boldsymbol{\lambda}(t))^T\mathcal{R}(\mathbf{x}(t)-\bar{\mathbf{x}}(t)) + (\boldsymbol{\lambda}(t)-\bar{\boldsymbol{\lambda}}(t))^T(\mathbf{y}(t)-\bar{\mathbf{y}}(t)) \\
&= (\bar{\boldsymbol{\lambda}}(t)-\boldsymbol{\lambda}(t))^T(\mathbf{y}(t)-\bar{\mathbf{y}}(t)) + (\boldsymbol{\lambda}(t)-\bar{\boldsymbol{\lambda}}(t))^T(\mathbf{y}(t)-\bar{\mathbf{y}}(t)) \\
&= 0
\end{aligned} \tag{2.17}$$

$$\text{Sum } 2 = \sum_{r \in R} (x_r(t) - \bar{x}_r(t)) \left(\frac{w_r}{x_r(t)} - \bar{q}_r \right) + \sum_{l \in L} (\lambda_l(t) - \bar{\lambda}_l(t))(\bar{y}_l(t) - c_l(t)) \quad (2.18)$$

In (2.18), the first term $\sum_{r \in R} (x_r(t) - \bar{x}_r(t)) \left(\frac{w_r}{x_r(t)} - \bar{q}_r \right) \leq 0$ since $\frac{w_r}{x_r(t)} \downarrow$ as $x_r(t) \uparrow$. The second term $\sum_{l \in L} (\lambda_l(t) - \bar{\lambda}_l(t))(\bar{y}_l(t) - c_l(t)) \leq 0$ as $\bar{\lambda}_l(t) = 0$ if $\bar{y}_l(t) < c_l(t)$. Therefore, from (2.16), (2.17), (2.18) and the above argument, we have $\frac{dV}{dt} \leq 0$. The equality only holds when $x_r(t) = \bar{x}_r(t)$ and for each link $\lambda_l(t) = \bar{\lambda}_l(t)$ or $\bar{y}_l(t) = c_l(t)$. Consequently, we conclude that the primal-dual algorithm is asymptotically stable in the name of trajectory-stability.

Notice that we have chosen to employ a time-variant Lyapunov function. In the remainder of this section, we first investigate the possible application of Lasalle's invariance principle, and then explain the reasons that Lasalle's invariance principle is not applicable to our scenario.

Consider the autonomous system

$$\dot{x} = f(x). \quad (2.19)$$

In a domain about an equilibrium point (without losing generality, let $x = 0$ be an equilibrium point for (2.19)), if we can find a Lyapunov function $V(x)$ whose derivative along the path of the system is negative semidefinite, and if we can establish that no path can stay identically at a point or points where $\dot{V}(x) = 0$, except at the equilibrium point, then the equilibrium point is asymptotically stable. This follows from LaSalle's invariance principle. That is, LaSalle's theorem enables us to conclude asymptotic stability of an equilibrium point even when $-\dot{V}(x)$ is not positive definite.

Let us also take equation (2.19) as an example to explain the largest invariant set in LaSalle's invarinace principle. The largest invariant set can be defined as $S = \{x \in D | \dot{V}(x) = 0\}$, where D is the domain of $V(x)$. Suppose that no solution can stay identically in S, other than the trivial solution $x = 0$. The equilibrium point $x = 0$ is then asymptotically stable. LaSalle's invariance principle applies only to autonomous or periodic systems [41, 42].

In the proof of the above theorem, we prove that $\dot{V}(\mathbf{x}(t), \boldsymbol{\lambda}(t); t) < 0$ except when $x_r(t) = \bar{x}_r(t)$ and for each link $\lambda_l(t) = \bar{\lambda}_l(t)$ or $\bar{y}_l(t) = c_l(t)$, that is, if it is on reference trajectory $\mathbf{x}^*(t)$. The Lyapunov function allows us to conclude asymptotic stability. Lasalle's invariance principle is not applicable for this scenario. Furthermore, since the system is time-varying, we can not apply Lasalle's invariance principle [41].

LaSalle's invariance principle can relax the negative definite requirement of Lyapunov's theorem. It also extends Lyapunov's theorem in another direction. LaSalle's invariance principle can be used where the system has an equilibrium set, rather than an isolated equilibrium point. For our scenario, since the system is time-varying, the equilibrium points form a reference trajectory, they are dependent on time t, thus a time-variant Lyapunov function $V(\mathbf{x}(t), \boldsymbol{\lambda}(t); t)$ is required [40, 41].

2.6 Stability Analysis with Delay

In the previous section, the trajectory stability of the primal-dual approach is analyzed in the absence of delay. When delay is considered, generally global stability is hard to obtain [10]. Instead, in this section, we will focus on local stability of the congestion control scheme in the presence of round trip time delay. Our focus is to obtain the sufficient conditions on system parameters for the system to be locally stable. As it is reasonable to assume that the trajectories of nonlinear systems in

a small neighborhood of an equilibrium point to be "close" to the trajectories of its linearization near that point [42], we still use the linearization technique to study local stability of congestion control with time delay, an approach also adopted by [18, 20–22].

We remark that linearization is a widely used technique in analyzing local stability in congestion control, which is actually borrowed from non-linear control theory. With no exception, we have followed the same line. The basic limitation of this local linearization approach though is the fact that the controller is guaranteed to work only in some neighborhood of a single operating (equilibrium) point [42]. In our case, to handle the varying link capacities, we have chosen to linearize around its nominal value.

Let τ_{lr}^f be the "forward" propagation delay from r's source $s(r)$ to link l and τ_{lr}^b be the "reverse" propagation delay from link l to the source $s(r)$ of route r. Then, $T_r = \tau_{lr}^f + \tau_{lr}^b$ for $l \in r$ is defined as the round trip delay (or round trip time, RTT) from source $s(r)$ to l. Therefore, with this propagation delay considered, the source rate controller becomes

$$\dot{x}_r(t) = k_r \left(w_r - x_r(t - T_r)q_r(t)\right) \tag{2.20}$$

where $q_r(t)$ is the route price given by

$$q_r(t) = \sum_{l \in r} \lambda_l(t - \tau_{lr}^b) \tag{2.21}$$

Our analysis will follow these steps. We will first linearize (2.20) around its local equilibrium point. Then we will transform the system to the frequency domain and

employ the Nyquist stability criterion to derive sufficient conditions for the system to be locally stable.

Introducing $\delta x_r(t) = x_r(t) - \bar{x}_r$ and $\delta q_r(t) = q_r(t) - \bar{q}_r$, where $\bar{x}_r$ and $\bar{q}_r$ are corresponding local equilibrium points of $x_r(t)$ and $q_r(t)$ respectively. By linearizing (2.20) around the equilibrium points $\bar{x}_r$ and $\bar{q}_r$, we have

$$\delta \dot{x}_r(t) = -k_r(\bar{q}_r \delta x_r(t - T_r) + \bar{x}_r \delta q_r(t)) \tag{2.22}$$

Taking the *Laplace* transform of the above equation and noting the fact that $\bar{q}_r = w_r/\bar{x}_r$ yields

$$\left(s + \frac{k_r w_r}{\bar{x}_r} e^{-sT_r}\right) x_r(s) = -k_r \bar{x}_r q_r(s) + x_r(0) \tag{2.23}$$

Let $D(s)$ be the diagonal matrix of RTTs in the Laplace domain, i.e., $D(s) = \text{diag}\{e^{-sT_r}\}$. Let $X = \text{diag}\{\bar{x}_r\}$, $W = \text{diag}\{w_r\}$ and $K = \text{diag}\{k_r\}$. The above Laplace transform equation can be rewritten as

$$\left(sI + KWX^{-1}D(s)\right)\mathbf{x}(s) + KX\mathbf{q}(s) = \mathbf{x}_0, \tag{2.24}$$

where $\mathbf{x}_0$ is the column vector of initial states given by $\mathbf{x}_0 = (x_{0r}, r \in R)$ and I is the identity matrix given by $I = \text{diag}\{1\}$.

For the price updating procedure depicted in (2.21), we take the Laplace transform and obtain

$$q_r(s) = \sum_{l \in r} e^{-s\tau_{lr}^b} \lambda_l(s) = e^{-sT_r} \sum_{l \in r} e^{s\tau_{lr}^f} \lambda_l(s) \tag{2.25}$$

Let $\mathcal{R}(s)$ denote the $|L| \times |\mathcal{S}|$ Laplace domain routing matrix that includes both routing and delay information, whose (l, r) entry is defined as

$$\mathcal{R}_{lr} = \begin{cases} e^{-s\tau_{lr}^f} & \text{if } l \in r, \\ 0 & \text{otherwise.} \end{cases} \tag{2.26}$$

Thus, from (2.25), we have

$$\mathbf{q}(s) = D(s)\mathcal{R}^T(-s)\boldsymbol{\lambda}(s) \tag{2.27}$$

where $\mathbf{q}(s) = (q_r(s), r \in R)$, $\boldsymbol{\lambda}(s) = (\lambda_l(s), l \in L)$.

Now we need to compute $\lambda_l(s)$ based on the price updating algorithm given in Equation (2.13). In order to keep this mathematical tractability, instead of taking the Laplace transform directly over $\lambda_l(t)$, we first assume that at equilibrium status, $c_l(t) = \bar{c}_l$ is a constant and introduce perturbations around $\bar{c}_l$ to "emulate" the effects of a time varying $c_l(t)$. By investigating the consequences of perturbation on the source controller and link dynamics, the effects of a time varying equilibrium point can be obtained. This technique has been well applied in the control domain and proven to be effective [42].

Following this approach, for the link price updating procedure given in (2.13), we introduce $\delta\lambda_l(t) = \lambda_l(t) - \bar{\lambda}_l$, $\delta z_l(t) = c_l(t) - \bar{c}_l$, and $\delta y_l(t) = y_l(t) - \bar{y}_l$. Linearizing (2.13) around equilibrium $\bar{\lambda}_l$ gives

$$\delta\dot{\lambda}_l(t) = h_l(\bar{\lambda}_l)[\delta y_l(t) - \delta z_l(t)] \tag{2.28}$$

Taking the Laplace transform for the above equation, we have

$$\lambda_l(s) = \frac{1}{s} h_l(\bar{\lambda}_l)[y_l(s) - z_l(s)] \tag{2.29}$$

As a result, we have

$$\boldsymbol{\lambda}(s) = \frac{1}{s} \text{diag}\left\{h_l(\bar{\lambda}_l)\right\}[\mathbf{y}(s) - \mathbf{z}(s)], \tag{2.30}$$

where $\mathbf{y}(s)$ and $\mathbf{z}(s)$ are column vectors and $\mathbf{y}(s) = (y_l(s), l \in L)$, $\mathbf{z}(s) = (z_l(s), l \in L)$.

In order to continue our discussion, a specific form of $h_l(\bar{\lambda}_l)$ is needed. For this purpose, we choose $h_l(\bar{\lambda}_l) = \beta_l \bar{\lambda}_l / \bar{c}_l$ as in [18]. Here β is often termed damping factor. In the remainder of this chapter, we will employ this function for price adjustment at the links. Based on this, from equations (2.27) and (2.30), we have

$$\mathbf{q}(s) = \frac{1}{s} D(s)\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l \bar{\lambda}_l}{\bar{c}_l}\right\}[\mathbf{y}(s) - \mathbf{z}(s)] \tag{2.31}$$

Hence,

$$\left[sI + KWX^{-1}D(s) + \frac{1}{s}KXD(s)\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l \bar{\lambda}_l}{\bar{c}_l}\right\}\mathcal{R}(s)\right]X(s) = \mathcal{Z}(s) \tag{2.32}$$

where $\mathcal{Z}(s) = X_0 + (1/s)KXD(s)\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l \bar{\lambda}_l}{\bar{c}_l}\right\}\mathbf{z}(s)$.

We remark that Equation (2.32) is different from corresponding equations in [18] in that, in (2.32), the perturbation of the link capacity results in the existence of the second term in $\mathcal{Z}(s)$. This term depicts the effect of capacity variations on the system and for the purpose of stability analysis must be investigated.

From control theory, the system described by (2.32) is stable if all its poles lie in the left-half of the complex plane. In other words, the solutions to

$$\det(sI + Q(s)) = 0 \tag{2.33}$$

should only have negative real parts, where

$$Q(s) = KWX^{-1}D(s) + \frac{1}{s}KXD(s)\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l\bar{\lambda}_l}{\bar{c}_l}\right\}\mathcal{R}(s) \tag{2.34}$$

We now will determine the conditions for the system to satisfy this requirement. Let $G(s) = (1/s)Q(s)$, then we can write

$$G(s) = \frac{1}{s}\left[KWX^{-1}D(s) + \frac{1}{s}KXD(s)\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l\bar{\lambda}_l}{\bar{c}_l}\right\}\mathcal{R}(s)\right] \tag{2.35}$$

Or equivalently,

$$G(s) = \text{diag}\left\{\frac{e^{-sT_r}}{sT_r}\right\}\text{diag}\{k_rT_r\}X\left[WX^{-2} + \frac{1}{s}\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l\bar{\lambda}_l}{\bar{c}_l}\right\}\mathcal{R}(s)\right] \tag{2.36}$$

From the generalized Nyquist Criterion [43], the stability condition of the proposed congestion control scheme is equivalent to the following statement: the eigenvalues of $G(j\omega)$ should not encircle the point -1. Therefore to guarantee local stability, we have to find conditions when the eigenvalues of $G(j\omega)$ do not encircle -1 for all values of ω. We will closely follow the line of analysis in [20]. From *Lemma* 3.1 in [18], we know that there exists an ω^* such that no eigenvalues of $G(j\omega)$ is real for all $\omega < \omega^*$. Therefore, we only need to prove that under some constraints, the eigenvalues of $G(j\omega)$ don't enclose -1 for $\omega > \omega^*$.

For ease of exposition, we define

$$G_1 = \text{diag}\left\{\sqrt{k_r T_r \bar{x}_r}\right\} W X^{-2} \text{diag}\left\{\sqrt{k_r T_r \bar{x}_r}\right\}$$
$$G_2 = \text{diag}\left\{\sqrt{k_r T_r \bar{x}_r}\right\} \mathcal{R}^T(-j\omega) \text{diag}\left\{\frac{\beta_l \bar{\lambda}_l}{\bar{c}_l}\right\} \mathcal{R}(j\omega) \text{diag}\left\{\sqrt{k_r T_r \bar{x}_r}\right\}$$
$$L = \text{diag}\left\{\frac{e^{-j\omega T_r}}{j\omega T_r}\right\}$$

Therefore,

$$G(j\omega) = \left(G_1 + \frac{1}{j\omega} G_2\right) L \tag{2.37}$$

In the following, we will derive certain conditions under which the eigenvalues of $G_1 L$ do not encircle $-\varepsilon$ and conditions for the eigenvalues of $\frac{1}{j\omega} G_2 L$ to be bounded by $(1-\varepsilon)$. Hence, if the conditions are jointly satisfied, the eigenvalues of $G(j\omega)$ will not encircle -1.

Note that

$$\sigma(G(j\omega)) = \sigma\left(\left(G_1 + \frac{1}{j\omega} G_2\right) L\right)$$

where $\sigma(\cdot)$ denotes the spectrum of a square matrix. Let λ be an eigenvalue of $G(j\omega)$ and $\mathbf{v}$ be the corresponding normalized eigenvector, i.e. $\|\mathbf{v}\|^2 = \mathbf{v}^*\mathbf{v} = 1$. Note that G_1 is a positive definite matrix and G_2 is a Hermitian matrix. According to the matrix theory, we have

$$\lambda \mathbf{v} = \left(G_1 + \frac{1}{j\omega} G_2\right) L\mathbf{v} \tag{2.38}$$

As G_1 is positive definite, the inverse G_1^{-1} exists. Therefore, from (2.38) we have

$$\lambda \mathbf{v}^* G_1^{-1} \mathbf{v} = \mathbf{v}^* \left(I + \frac{1}{j\omega} G_1^{-1} G_2\right) L\mathbf{v} \tag{2.39}$$

Hence,

$$\lambda = \frac{\mathbf{v}^* L\mathbf{v}}{\mathbf{v}^* G_1^{-1}\mathbf{v}} + \frac{1}{j\omega}\frac{\mathbf{v}^* G_1^{-1} G_2 L\mathbf{v}}{\mathbf{v}^* G_1^{-1}\mathbf{v}} \tag{2.40}$$

For the first term in (2.40), we note that

$$\mathbf{v}^* L\mathbf{v} = \sum_r |v_r|^2 \frac{e^{-j\omega T_r}}{j\omega T_r},$$

and

$$\mathbf{v}^* G_1^{-1}\mathbf{v} \geq \lambda_{\min}(G_1^{-1}) = \frac{1}{\lambda_{\max}(G_1)}$$

Thus,

$$\sigma(G_1 L) \subset \lambda_{\max}(G_1) \sum_r |v_r|^2 \frac{e^{-j\omega T_r}}{j\omega T_r} \tag{2.41}$$

Therefore, based on the fact that as ω is varied from $-\infty$ to ∞, $\frac{\pi}{2}\frac{e^{-j\omega T_r}}{j\omega T_r}$ does not encircle the point -1(see [18] for its proof), we can choose k_r such that $\lambda_{\max}(G_1) < \varepsilon\frac{\pi}{2}$ and consequently force $\sigma(G_1 L)$ not to encircle $-\varepsilon$, $\forall\, 0 < \varepsilon < 1$. Specifically, the condition to satisfy the above requirement for G_1 is given by

$$k_r \bar{q}_r < \varepsilon \frac{\pi}{2T_r}, \quad \forall r \in R \tag{2.42}$$

Next, we will analyze the second term in (2.40). Our goal is to prove

$$\left| \frac{1}{j\omega} \frac{\mathbf{v}^* G_1^{-1} G_2 L\mathbf{v}}{\mathbf{v}^* G_1^{-1}\mathbf{v}} \right| < 1 - \varepsilon$$

Note that

$$\left| \frac{1}{j\omega} \frac{\mathbf{v}^* G_1^{-1} G_2 L\mathbf{v}}{\mathbf{v}^* G_1^{-1}\mathbf{v}} \right| \leq \frac{1}{\omega^*} \frac{\|G_1^{-1}\|_2 \|G_2\|_2 \|L\|_2}{\lambda_{\min}(G_1^{-1})} \tag{2.43}$$

where $\|\cdot\|_2$ is the matrix norm given by $\|A\|_2 = \sqrt{\lambda_{\max}(A^* A)}$ [44]. The above inequality follows directly from the Cauchy-Schwartz inequality and $\lambda_{\min}(A) \leq \mathbf{x}^T A\mathbf{x} \leq \lambda_{\max}(A)$ for any vector $\mathbf{x}$ satisfying $\mathbf{x}^T\mathbf{x} = 1$ and any positive-definite matrix A.

Now, we investigate the RHS of inequality (2.43)

$$\begin{aligned}
&\frac{1}{\omega^*}\frac{\|G_1^{-1}\|_2\|G_2\|_2\|L\|_2}{\lambda_{\min}(G_1^{-1})} \\
&\quad\le \frac{1}{(\omega^*)^2}\max_r\left(\frac{1}{T_r}\right)\times\frac{\lambda_{\max}(G_1)\lambda_{\max}(G_2)}{\lambda_{\min}(G_1)} \\
&\quad\le \frac{1}{(\omega^*)^2}\max_r\left(\frac{1}{T_r}\right)\lambda_{\max}(G_2)
\end{aligned} \tag{2.44}$$

Let $\rho(\cdot)$ denote spectral radius of a square matrix, and

$$\hat{\mathcal{R}}(j\omega) = \mathrm{diag}\left\{\sqrt{\beta_l\bar{\lambda}_l/\bar{c}_l}\right\}\mathcal{R}(j\omega)\mathrm{diag}\left\{\sqrt{k_rT_r\bar{x}_r}\right\},$$

then we have

$$\begin{aligned}
\lambda_{\max}(G_2) &= \rho\left(\hat{\mathcal{R}}^T(-j\omega)\hat{\mathcal{R}}(j\omega)\right) \\
&= \rho(\mathrm{diag}\{k_rT_r\bar{x}_r\}\mathcal{R}^T(-j\omega)\mathrm{diag}\left\{\frac{\beta_l\bar{\lambda}_l}{\bar{c}_l}\right\}\mathcal{R}(j\omega)) \\
&\le \|\mathrm{diag}\{k_rT_r\}\mathcal{R}^T(-j\omega)\mathrm{diag}\{\beta_l\bar{\lambda}_l\}\| \\
&\quad\times\|\mathrm{diag}\left\{\frac{1}{\bar{c}_l}\right\}\mathcal{R}(j\omega)\mathrm{diag}\{\bar{x}_r\}\| \\
&\le \|\mathrm{diag}\{k_rT_r\}\mathcal{R}^T(-j\omega)\mathrm{diag}\{\beta_l\bar{\lambda}_l\}\|\times 1
\end{aligned} \tag{2.45}$$

The last inequality above uses the fact that at equilibrium point

$$\sum_{r:l\in r}\bar{x}_r = \bar{y}_l \le \bar{c}_l,\ \ \forall l\in L$$

For the first term in the last inequality in (2.45), if we set

$$\beta_l < \frac{\min_r(T_r)}{k_rT_r\bar{q}_r}(\omega^*)^2\times(1-\varepsilon), \tag{2.46}$$

we have

$$\|\text{diag}\{k_r T_r\}\mathcal{R}^T(-j\omega)\text{diag}\{\beta_l \bar{\lambda}_l\}\| \leq \min_r(T_r)(\omega^*)^2(1-\varepsilon)$$

Therefore,

$$\lambda_{\max}(G_2) < \min_r(T_r)(\omega^*)^2(1-\varepsilon) \tag{2.47}$$

Combining (2.44) and (2.47), we have

$$\frac{1}{(\omega^*)}\frac{\|G_1^{-1}\|_2\|G_2\|_2\|L\|_2}{\lambda_{\min}(G_1^{-1})} < 1-\varepsilon \tag{2.48}$$

Till now, we have proven that the first term of (2.40) does not encircle ε given the condition in (2.42) is satisfied and the second term (2.40) does not encircle $(1-\varepsilon)$ as shown in (2.48) and (2.43) as long as (2.46) is satisfied. Therefore, for any $\omega > \omega^*$, if the two conditions are satisfied, the eigenvalues of $G(j\omega)$ will not encircle -1 and hence the system is locally stable.

Therefore, sufficient conditions for guaranteeing the system with time delay to be locally stable can be summarized as

$$\begin{cases} k_r \bar{q}_r \leq \varepsilon \frac{\pi}{2T_r}, & \forall\, r \in R \quad \text{and} \\ \beta_l < \frac{\min_r(T_r)}{k_r T_r \bar{q}_r}(\omega^*)^2 \times (1-\varepsilon), & \forall\, r \in R:\ l \in r. \end{cases} \tag{2.49}$$

2.7 Sensitivity Analysis of Link Capacity with Perturbation

In this section, we will study how to reduce the effects owing to link capacity perturbation. Equivalently, our goal is to study proper system parameters so that the system is robust to the perturbations. In other words, if the system is insensitive to capacity perturbations, rate oscillations will be kept at a low level even in the presence

of large capacity changes. Such a feature is much desired by wireless networks for example, in order to provide quality of service to higher layer applications.

Before going to the details, let us first refresh system sensitivity using a system characterized by linear equation $Ax = b$. For this example system, when A and b are subjected to small order perturbation ΔA and Δb, respectively, the problem becomes $(A + \Delta A)(x + \Delta x) = b + \Delta b$. Our main concern is the deviation Δx of the solution with respect to the perturbation of ΔA and Δb. The system sensitivity is thus defined as the extent of the deviation of Δx relative to ΔA and Δb [45]. Putting this into the congestion control scheme we are concerned, our main target is to investigate the deviation of system equilibrium relative to the link capacity change.

Let us consider Equation (2.32). We remark that as X_0 is the initial condition vector, its existence will not affect our sensitivity analysis and hence can be ignored. Towards this end, we define

$$\mathcal{Z}'(s) = (1/s)KXD(s)\mathcal{R}^T(-s)\text{diag}\left\{\frac{\beta_l\bar{\lambda}_l}{\bar{c}_l}\right\}\mathbf{z}(s), \tag{2.50}$$

and our focus becomes

$$[sI + Q(s)]X(s) = \mathcal{Z}'(s). \tag{2.51}$$

General sensitivity analysis will involve matrices (in Laplace domain) $\mathcal{R}(s)$, $\mathcal{R}^T(-s)$, and $[sI + Q(s)]^{-1}$. Computation of $[sI + Q(s)]^{-1}$ is often dependent on the specific system setup and becomes intractable for complex systems. To illustrate the idea and avoid tedious matrix manipulations, we consider a simple scenario of one source transmitting over one link in the following discussion. Then (2.51) can be expressed as

$$\left(s + k\bar{\lambda}e^{-sT} + \frac{1}{s}\frac{k\beta\bar{\lambda}\bar{x}e^{-sT}}{\bar{c}}\right)x(s) = \frac{1}{\bar{c}s}k\beta\bar{\lambda}\bar{x}e^{-sT}e^{s\tau^f}z \tag{2.52}$$

At equilibrium point, $\bar{c} = \bar{x}$ and $\bar{\lambda} = w/\bar{x}$, the above equation yields

$$\left(s + \frac{kw}{\bar{c}}e^{-sT} + \frac{1}{s}\frac{k\beta we^{-sT}}{\bar{c}}\right)x(s) = \frac{k\beta w}{\bar{c}s}e^{-sT}e^{s\tau^f}z \tag{2.53}$$

and it follows that

$$x(s) = \frac{k\beta we^{-sT}e^{s\tau^f}}{\bar{c}s^2 + kwe^{-sT}s + k\beta we^{-sT}}z \tag{2.54}$$

Define

$$T(s) = \frac{k\beta we^{-sT}e^{s\tau^f}}{\bar{c}s^2 + kwe^{-sT}s + k\beta we^{-sT}} \tag{2.55}$$

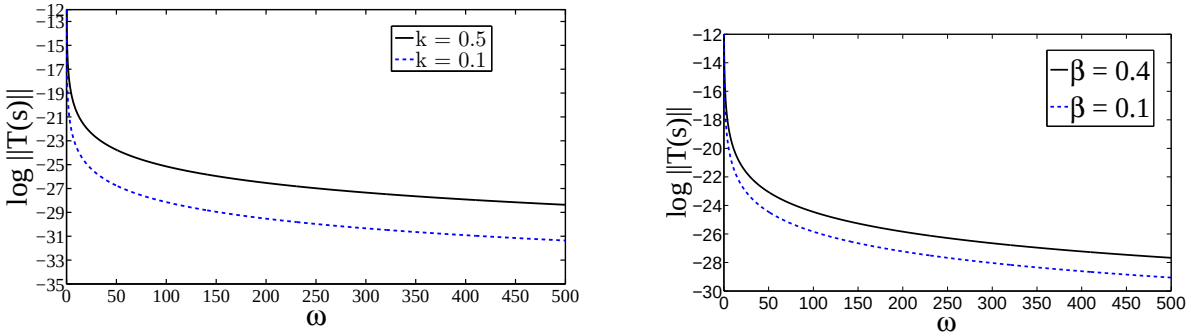
In order to minimize the effect of link perturbation on source rate, we need to minimize $|T(s)|^2_{s=j\omega}$, that is,

$$\min_{k,\beta} \quad \|T(s)\|^2_{s=j\omega} = \min_{k,\beta}\left\|\frac{k\beta w}{\bar{c}s^2 + kwe^{-sT}s + k\beta we^{-sT}}\right\|^2_{s=j\omega} \tag{2.56}$$

Evidently, the above optimization depends on delay T, the source controller gain k, and link damping factor β. Given the time delay T on the link, the range of k and β for guaranteeing the system is local stability can be obtained through Inequality (2.42) and (2.46).

Notice that $\|T(s)\|$ is a strongly nonlinear function. Closed form expression of the optimal solution is hard, if not impossible, to obtain. Instead, we have to rely on numerical tools to study this function. Fig. 2.4(a) provides an illustrative result on the dependence of $\|T(s)\|$ on k given that β and T are fixed. Notice that, $\|T(s)\|$ decreases as k decreases, which means the sensitivity of the system also decreases. However, even though we can reduce the effect of link perturbation on source rates by

reducing k, we still have to balance the tradeoff among system stability, sensitivity and convergence rate of congestion algorithms, as a small k will result in slow convergence rate. On the other hand, given k and delay T, as shown in Fig. 2.4(b), $\|T(s)\|$ decreases as β decreases. However, at the same time, convergence rate will also decrease.



(a) (b)

Figure 2.4: Numerical study of $\|T(s)\|$. (a) k's effect on $\|T(s)\|$. (b) β's effect on $\|T(s)\|$.

2.8 Simulation

In this section, we perform a broad set of simulations to validate our theoretical results and study the effects of different control parameters, for example, source controller gain k and damping factor β of the proposed algorithm.

2.8.1 System Setup

If the stability conditions are satisfied, then the system is stable. The dual algorithm at a link for price updating in (2.13) is implemented by marking packets

with probability (the link price) λ_l as an exponential function of the queue length b_l [18], given by

$$\lambda_l = \begin{cases} 0 & \text{if} \quad 0 \leq b_l < th_{\min,l}, \\ p_{\min} e^{\frac{\beta_l}{c_l}(b_l - th_{\min,l})} & \text{if} \quad th_{\min,l} < b_l < th_{\max,l}, \\ 1 & \text{if} \quad b_l \geq th_{\max,l}. \end{cases} \tag{2.57}$$

where $th_{\min,l}, th_{\max,l}$ $(th_{\min,l} < th_{\max,l})$ are the two user defined queue length thresholds and $p_{\min}$ is the marking probability when $b_l = th_{\min,l}$. In our simulation, $th_{\min,l} = 4$, $th_{\max,l} = 10$ packets, and $p_{\min} = 0.002$. This scheme is termed E-RED and details are given by [20, 38]. Again, the primal algorithm to perform rate update is given by

$$\dot{x}_r(t) = k_r \left(w_r - x_r(t) \sum_{l \in r} \lambda_l \right) \tag{2.58}$$

The above primal algorithm is implemented as a rate-based control scheme. The reason is that rate-based control is less sensitive to round time variation than window-based transport protocols [37].

The simulations are carried out on the *ns*-2 platform. In the simulations, the buffer limit of the bottleneck link is set to be 500 packets and all packets are fixed to 512 bytes. We set up different network topologies for global stability, and local stability and sensitivity.

2.8.2 Global Stability

For fixed link capacities, extensive results have been shown in the literature on global stability in the absence of delay, and hence we focus on scenarios with time varying capacities.

We first set up a topology with single bottleneck link as shown in Fig. 2.5. The link capacity between node 1 and 2 is set to be 10 Mbps with a variation simulated by a sine function, whose amplitude and period are 5 Mbps and 30 seconds, respectively.

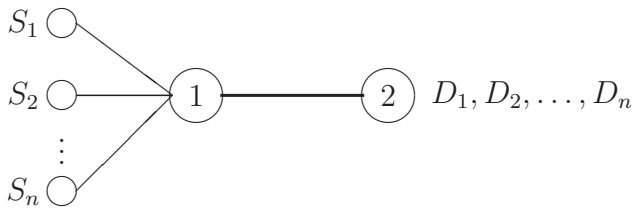


Figure 2.5: A single bottleneck link topology

In this simulation, there are 30 traffic flows, 10 of whom stop at 100 seconds, and resume at 250 seconds. The results are depicted in Fig. 2.6(a) and (b) and (c). From the simulation, we observe that the trajectory stability can be guaranteed under various chosen parameters in the sense that link utilization and source rates can closely follow the link capacity oscillation after short transient phases, the choice of parameters can affect the utilization of link bandwidth and oscillation amplitude. We further notice that the stability is actually not affected by damping factor β of the link algorithm.

We next introduce a constant rate UDP traffic flow of 1 Mbps and study the performance on the same topology. This UDP flow arrives at 200 seconds and stops at 250 seconds. From Fig. 2.7, we observe that as this UDP traffic flow cannot adjust its traffic rate, its existence reduces the link utilization. We also notice that the fluctuation of the available bandwidth introduced by the UDP flow can be effectively absorbed by the congestion control algorithm, as the system can quickly re-stabilize when the UDP enters and leaves the network.

From both Fig. 2.6 and 2.7, we can see that even though smaller source controller gains can reduce the oscillation amplitudes of source rates and link utilization, they also negatively affect the transient phases. In other words, the source controllers with large controller gains can finish the transient phase in shorter time than the ones with small controller gains.

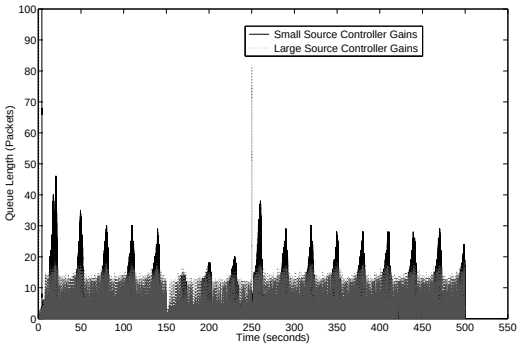
2.8.3 Local Stability and Sensitivity

As shown in our analysis, the local stability when delay is present significantly depends on control parameters such as k, β, and RTTs of the system. In this set of simulations, we set up a multi-hop network topology as shown in Fig. 2.8, and study the effects of those control parameters on the utilization of link bandwidth, queue length and source rates when the capacity of the bottleneck link is time varying. We assume that the capacity of the bottleneck link 23 is 2 Mbps. The link perturbation, if present, is simulated by a sine function whose amplitude and period are half of the fixed link capacity and 30 seconds, respectively. The capacities of all other links are set to be 5 Mbps.

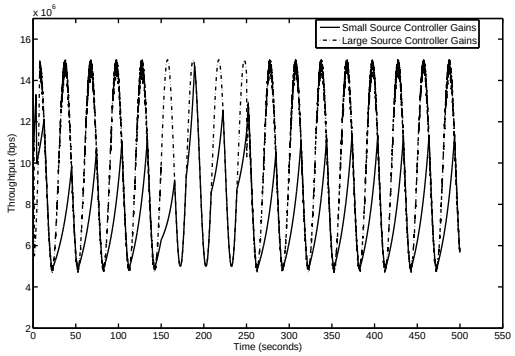
We assume that there are 5 traffic flows in the network as shown in Fig. 2.8. Traffic flow 5 comes at 250 seconds, and other traffic flows start at 0 second. All the traffic flows stop at 500 seconds. Notice that flow 5 introduces dynamic traffic load into the network. The behavior of the network (for example, time to re-stabilize) during this transient phase is one of the key criteria measure the performance of the congestion control algorithm.

2.8.3.1 Local Stability and Sensitivity Without Link Perturbation

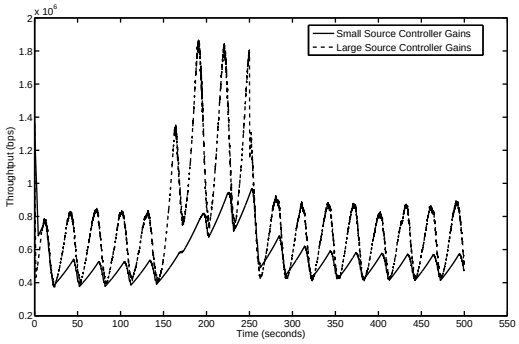
In this case, we study the system performance in the absence of link perturbation. We first set source controller gains in the system as $[k_1\ k_2\ k_3\ k_4\ k_5] =$



(a)

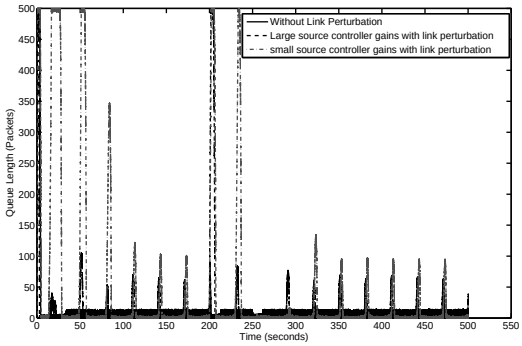


(b)

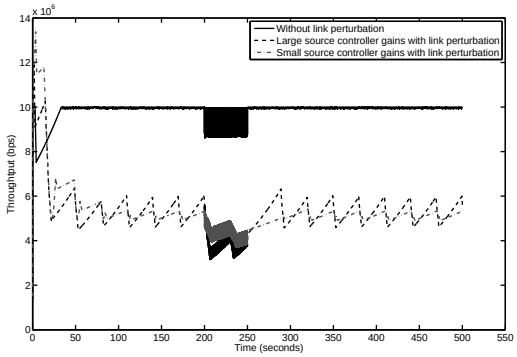


(c)

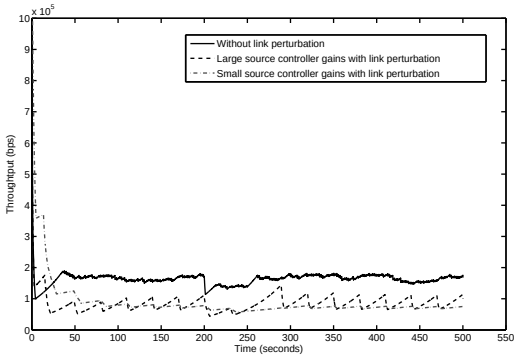
Figure 2.6: Global stability with link perturbation. (a) Queue length at link 12. (b) Utilization link 12. (c) Source rate of a traffic.



(a)



(b)



(c)

Figure 2.7: Global stability with link perturbation and UDP traffic. (a) Queue length at link 12. (b) Utilization of link 12. (c) One source rate.

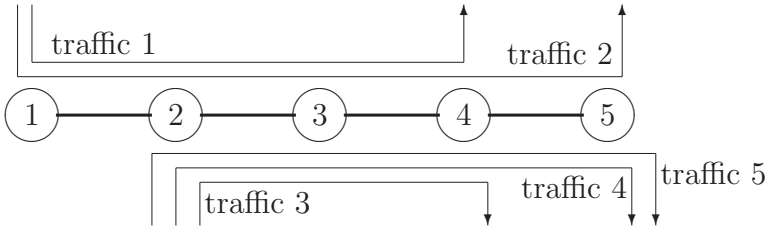
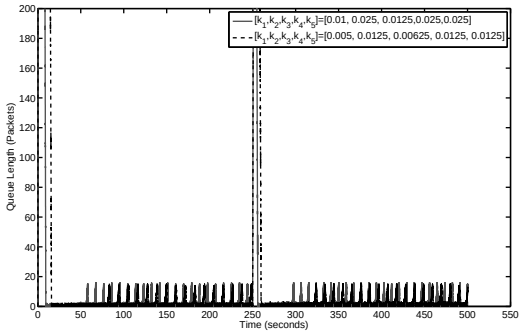


Figure 2.8: A multihop network topology

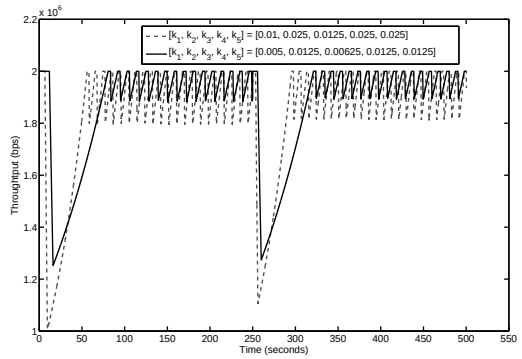
[0.01 0.025 0.0125 0.025 0.025] where k_i is the gain for source i, then we reduce all the gains by half while keeping other parameters fixed and study their differences. The results for these two cases are shown in Fig. 2.9. From Fig. 2.9(a), we can see that the queue length is being kept between 0 and 20 packets after the transition phase, denoting a stable condition. Correspondingly, Fig. 2.9(b) shows that the throughput of the bottleneck link oscillates around the equilibriums slightly. Fig. 2.9(c) shows one of the source rates around its equilibrium. By comparing the results with different controller gains, we observe that adjusting source controller gains can affect the throughput oscillation magnitude around the equilibrium. The throughput oscillation magnitude in the transition phase (when flow 5 enters the network) can also be reduced by adjusting the source controller gains. We also observe that the transient phase can end faster with large source controller gains than with small controller gains. In the simulations, we also find that our algorithm is not sensitive to the damping factor β for price updating at the link and RTTs, which concur with the conclusion from [37].

2.8.3.2 Local Stability and Sensitivity with Link Perturbation

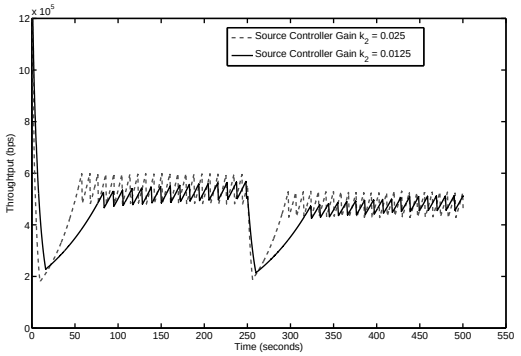
In this case, we still use the scenario described in Fig. 2.8 but introduce the link perturbation simulated by a sine wave. Before going into the details, we first remark that in the presence of capacity variation and system delay, we observe in our simula-



(a)



(b)



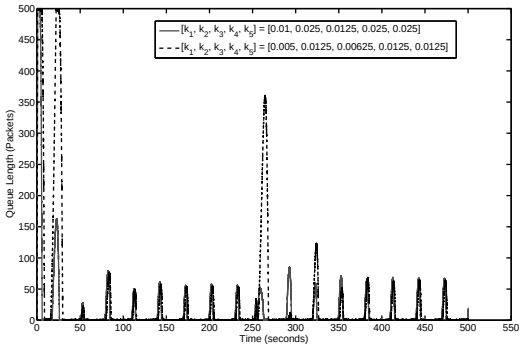
(c)

Figure 2.9: Local stability without link perturbation. (a) Queue length at link 23. (b) Utilization of link 23. (c) Source rate of flow 2.

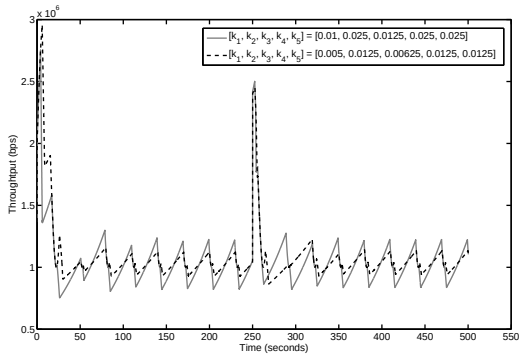
tion that if perturbation is significant (large amplitude of the sine wave), the system is unable to stabilize regardless of the parameters chosen. This actually concurs with our analytical results as given a certain perturbation the sufficient conditions may not be able to be satisfied by adjusting the system parameters. For a reasonable perturbation, however, suitable parameters chosen according to the derived conditions can stabilize the system.

As in the case without link perturbation, we first set source controller gains in the system as $[k_1\ k_2\ k_3\ k_4\ k_5] = [0.01\ 0.025\ 0.0125\ 0.025\ 0.025]$ where k_i is the gain for source i, then we reduce all the gains by half while keeping other parameters fixed and study their differences. The stable scenarios with the amplitude of the sine wave set to be 1 Mbps are shown in Fig. 2.10. We can see that the system can effectively adapt to the capacity change and achieve stabilized queue length and link utilization. We also observe in our simulation that the utilization of the bottleneck link depends on values of k while maintaining this local stability. It is a balance to be carefully tuned between the utilization of the bottleneck link and the robustness of the control algorithm against link capacity variation.

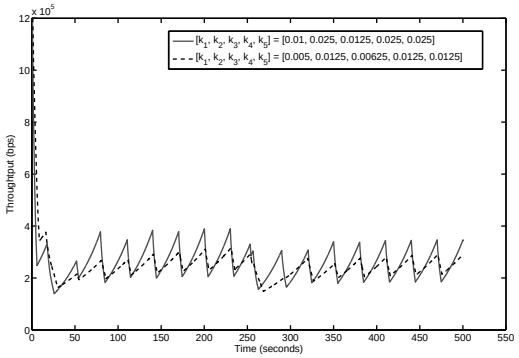
On the other hand, if the parameters are not chosen properly according to the conditions derived, that is, if the sufficient condition for stability is violated, the system may not be able to stabilize either. For a large source controller gain, the results are depicted in Fig. 2.11(a) and (b) and (c), where the large oscillations for the source rate and the link utilization reveal an unstable system. Similar results are observed for unsatisfactory values of the damping gain and different RTTs which are omitted here.



(a)

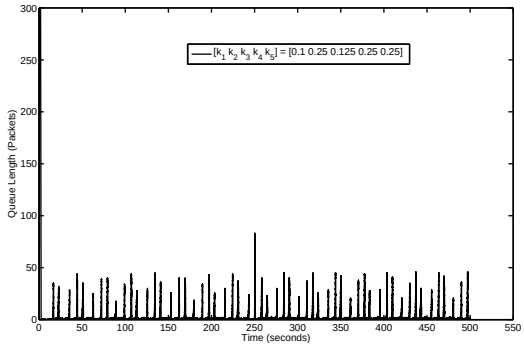


(b)

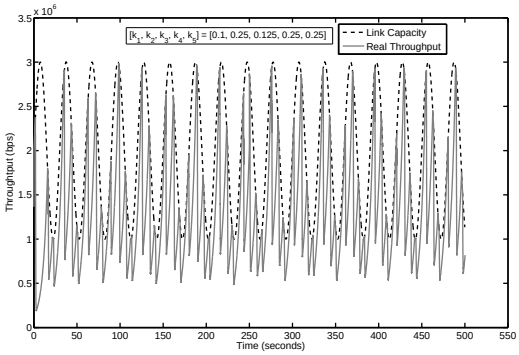


(c)

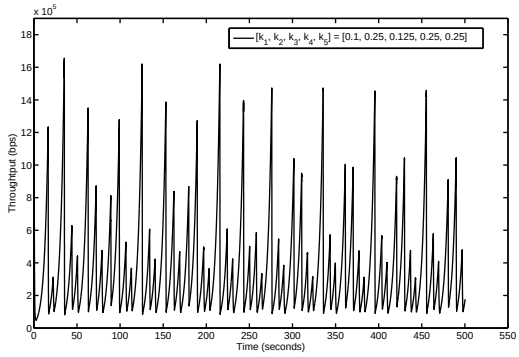
Figure 2.10: Local stability with link perturbation - stable scenario. (a) Queue length at link 23. (b) Utilization of link 23. (c) Source rate of flow 2.



(a)



(b)



(c)

Figure 2.11: Local stability with link perturbation — unstable scenario. (a) Queue length at link 23. (b) Utilization of link 23. (c) Source rate of flow 2.

2.9 Conclusion

By explicitly introducing time varying channel capacity into the utility maximization problem, we investigate congestion control in networks with time varying link capacities, following Kelly's seminal work. Different from conventional system stability around an equilibrium point, we employ trajectory stability and prove that the proposed prime-dual algorithm is stable around a time varying trajectory without considering system delay. In the presence of system delay, we derive sufficient conditions for the system to be locally stable. Furthermore, by modeling link variations as perturbations to constant capacities, we investigate system sensitivity and provide insightful experimental studies regarding its tradeoff with system stability.

CHAPTER 3
CONGESTION CONTROL IN INTERMITTENTLY COMMUNICATING NETWORKS — DYNAMIC PROGRAMMING APPROACH

3.1 Introduction

Different from conventional networks exemplified by the mighty Internet, an important class of intermittently communicating networks (also called delay tolerant networks or disruption tolerant networks) often faces long round trip delay, intermittent connectivity, and opportunistic contacts among nodes, which can be traced back to its origin of deep space communication [47, 50]. Correspondingly, store-and-forward, message-oriented architectures, in contrast with the dominating end-to-end architecture of the current Internet, are often adopted to cope with the challenged environments [48]. In particular, to enhance end-to-end reliability, *custody transfer*, is proposed where the responsibility for reliable delivery of a message (*often termed a bundle*) is gradually moved toward its ultimate destination in a hop-by-hop fashion [48].

Unfortunately, by accepting the custody of a bundle (in other words, the responsibility of reliable delivery of a bundle), a node may have to store the bundle for a significant period of time before being able to hand it over to another node during opportunistic contact, as discarding a bundle before its life time expires is generally prohibited. As a result, precious storage space is committed by accepting the custody of a bundle which may likely hinder acceptance of future custody transfer requests. Therefore, it may not be entirely wise for a node to openly and widely declare oneself to be a willing storage device for any set of bundles requested. Rather, a carefully crafted congestion management strategy is desired in order to effectively

manage the storage capacity on a node in the intermittently communicating networks so that the network utilization can be maximized. It is this problem that this chapter aims to address. For convenience, thereafter, we use "delay tolerant networks" and "intermittently communicating networks" interchangeably.

Although bundle handling protocols and routing schemes specific to delay tolerant networks have been extensively proposed [47, 61, 64, 69, 71, 73], little results, as to our best knowledge, exist on how to handle the above discussed congestion/resource allocation problem. Given the plethora of potential applications of delay tolerant networks in various domains, congestion will become imminent when the network technology is successfully applied and proper management schemes are demanded.

Fortunately, a node cannot discard the bundle unless its life time expires or the custody is transferred to another node after a commitment, the node indeed has the freedom in deciding whether to accept the custody in the first place. Consequently, two conflicting forces can be considered that are governing the receiving node's actions: on one hand, it is beneficial to accept a large number of messages as it can potentially advance the messages toward their ultimate destinations and network utilization can be maximized; on the other hand, if the receiving node over-commits itself by accepting too many messages, it may find itself setting aside an excessive amount of storage and thereby preventing itself from receiving further potentially important, high yield (in terms of network utilization) messages.

We apply the concept of revenue management, and employ dynamic programming to develop a congestion management strategy for delay tolerant networks to solve the above problem. For a class of network utility functions, we show that our solution is optimal. More importantly, our solution is distributed in nature where only the local information such as available buffer of a node is required. This is particularly important given the nature of delay tolerant networks where global in-

formation is often not available and the network is inherently dynamic. Extensive simulation results show that the proposed congestion management scheme is effective in avoiding congestion and balancing network load among the nodes.

This chapter is organized as follows. In Section 3.2, we investigate the related work for delay tolerant networks. Section 3.3 briefly presents some concepts of dynamic programming and delay tolerant networks as basis for further investigation. The congestion management problem is formulated in Section 3.4. In Section 3.5, we establish the dynamic programming formulation for the defined congestion management problem in Section 3.4. Section 3.6 studies the optimal strategies for congestion management in delay tolerant networks with multiple nodes. In Section 3.7, we use several simulation scenarios to show the performance of the derived optimal control policies. Section 3.8 concludes this chapter.

3.2 Related Work

Delay tolerant in general has been the subject of a wide range of research [47, 48, 55, 60, 61, 64, 65, 67, 69, 71, 73–75]. Among those, a set of papers have focused on the routing problems [67, 71, 73–75]. In these works, usually buffer space in each node is assumed to be unlimited or treated in an ad-hoc manner, as resource allocation is not the key focus there. For example, in [66], it is assumed that the send buffer and receive buffer have limited space in designing message ferry route, but the congestion issue is not addressed.

While the bundle layer employs reliable transport layer protocols together with custody transfers to offer hop-by-hop reliability, no end-to-end reliability can be guaranteed. In [55], the authors have developed active receipt, passive receipt, and network-bridged receipt approaches to offer end-to-end reliability by delivering an

active end-to-end acknowledgment over the delay tolerant network itself or another network.

It is worth noting that the storage congestion mitigation problem is handled in [49]. When a custodian node becomes congested, the authors propose to migrate its stored messages to alternative storage (usually neighboring nodes) locations to avoid losses. The problem of storage congestion on a node is approached by migrating the stored data to neighbors. The proposed solution includes a set of algorithms to determine which messages should be migrated to which neighbors and when. However, this approach is a passive approach. In contrast, our approach is proactive in that each node actively makes decisions on whether to accept a custody request in order to avoid congestion. In [50], a financial model based approach is adopted in addressing the congestion problem in delay tolerant networks, the concept introduced there such as conveyance fee paid by sender and receiver of a bundle is different from the concepts of benefits of requests and opportunity cost proposed in our work. Furthermore, our work achieves distributed optimal solution when multiple nodes are present.

On the other hand, resource allocation has been the subject of extensive study in related overlay networks. In [59], resource allocation in overlay networks to protect the network from being overloaded is investigated. There, cost function and benefit function are defined. And subsequently, the decision whether to accept packets into the network is made by comparing the benefit of accepting a packet with cost on relevant paths. However, the approach is a centralized one since an oracle is needed to compute the total cost of all links in a path. While the concepts of benefit and opportunity cost in our work are similar to the ones in [59], our approach is based on local information of each node which makes it applicable to the dynamic environment of delay tolerant networks.

There also exist a set of papers studying that network capacity can be significantly improved by exploring the node mobility in wireless networks [62, 79–81]. For example, in [79], the authors introduced autonomous agents as additional participants in delay tolerant networks. The agents can adapt their movements in response to variations in network capacity and demand to improve network performance. In [81], authors investigated the delay-throughput tradeoffs in mobile ad hoc networks under hybrid random walk and one dimensional mobility models. Our congestion handling strategy developed later is a general approach independent of the mobility model. Instead, it only depends on the remaining storage space in receiving nodes and the request reward.

3.3 Preliminaries

While an introduction to delay tolerant networks has been provided in Chapter 1, we here recapitulate the properties of delay tolerant networks for better understanding the remaining contents of this chapter.

Delay tolerant networks have attracted tremendous interests from academia, military and industry recently. Interested applications include interplanetary networks, mobile tactical military networks, communication networks for remote rural area, which often consist of wireless communications and user mobility [47, 50]. In these environment, often building a standard network with end-to-end connectivity is impractical, or transmission latencies are inherently high. Often such challenges arise due to user mobility and wireless communications.

In order to address networking issues in these challenged environments where end-to-end connections are often absent, delay tolerant networks implement a store-and-forward message switching architecture where each intermediate forwarding node is embedded with storage and intelligence for forwarding decision making. This is

achieved by overlaying a new protocol layer, termed the *Bundle Layer*, at the application layer or at least above the transport layer [47]. The bundle layer stores and forwards entire bundles (or bundle fragments) between nodes. Bundles are also called messages, which can consist of multiple pieces of applications data. Usually, a single bundle layer is employed across potentially heterogenous network domains in order to form a delay tolerant network.

In order to provide reliable delivery, the bundle protocol employs a mechanism termed custody transfer where delivery responsibility (custody) is transferred among successive nodes in a hop-by-hop fashion. Specifically, a node holding a bundle with custody is called custodians. When the current custodian sends a bundle to the next node, it requests a custody transfer. If the next node accepts the custody, it returns an acknowledgement to the sender. Accepting a message with custody transfer amounts to promising not to delete it until either another node accepts custody (or it is delivered to the message's destination), or the expiration of the message's time to live.

The bundle layer can use reliable transport layer protocols together with custody transfers to move points of retransmission progressively forward toward the destination. This property minimizes the number of potential retransmission hops, and consequently reduces additional network load caused by retransmission, and the total time to convey a bundle reliably to its destination.

3.4 Problem Formulation

While nodes in delay tolerant networks may offer custody transfer, this decision is at their own discretion. Furthermore, a node that has been accepting messages and corresponding custodies can decide, on its own, to cease the accepting option if its local resources become substantially consumed. Similarly, the accepting operation

can be resumed at the node when resources become more plentiful after, for example, the messages are successfully transferred to its downstream nodes. It is this decision as to whether to accept the custody of a message given the current resource constraint we are addressing in this chapter. In doing so, our goal is to maximize the overall system benefit subject to the constraints of the system resources.

Before proceeding further on the formal model, we remark that routing algorithm can affect the performance of delay tolerant networks significantly. Unfortunately, complete knowledge of the delay tolerant network and routes is often not available in advance nor are they static. In order to focus on the buffer management problem, we follow [49] and separate the management mechanism form the route selection problem.

3.4.1 System Model

Without loss of generality, assume that node $i+1$ is any node to whom node i can forward messages by custody transfer. Similarly, node $i-1$ can be considered as any node that wants to forward messages to its downstream custodian i during a contact opportunity. Node $i-1$ sends a request for a new custodian to fulfill the bundle transfer. It then waits for acknowledgement from its neighbor node. If node i receives a request for custodian from its neighbor node $i-1$, it will decide whether to accept or reject the request based on the current available storage space, and predefined optimal control strategies. Here we assume that the storage space is the key resource constraint. Notice that we have not made any assumption regarding the contacts among nodes and hence can accommodate different mobilities.

By accepting a bundle, a node accumulates a certain amount of benefit denoted by $B_\ell > 0$, where $\ell \in \{1, \cdots, m\}$ is the index of the class of benefits. The benefit functions can be of various forms and correspondingly different optimization goals can

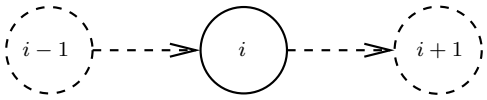


Figure 3.1: Simple DTN Scenario

be achieved. For example, by properly adjusting the benefit function, performance issues such as throughput and policy issues such as fairness can be addressed [59].

Notably, the benefit can be a function of the bundle size with different weights based on their corresponding traffic priorities. For simplicity, we assume that the bundle sizes of different priorities are homogeneous in volume and thus indistinguishable when filling the buffer if they are accepted. We also assume that arrivals of requests for custody occur at discrete points in time, which are called decision epochs. Notice that the arrival requests (events) drive the decision epochs, in other words, the decision epochs are not deterministic but rather given by the arrival requests themselves. We also assume that the departure of a message can occur at anytime between decision epochs.

Over finitely many decision epochs, i.e., a finite time horizon, our objective is to determine the optimal congestion management strategies that maximize the expected total benefits by accepting/forwarding the bundles, subject to the request and buffer constraints. Formally, the objective is to choose congestion management strategies that maximize the total expected reward

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t u_t\right], \tag{3.1}$$

over a time horizon of T decision epochs, where $r_t \in \{B_1, B_2, \ldots, B_m\}$, and $u_t = 1$ if the transfer request is accepted at decision epoch t, $u_t = 0$ otherwise; $\mathbb{E}(\cdot)$ is mathematical expectation.

We here remark that in this model, for each hop of custody transfer, the benefit is accumulated. And these benefits are summarized across the whole network over a finite time period. Additionally, unlike many economic models, our model does not try to reach an equilibrium state based on the rationality of participant nodes or influence noncooperative behavior. Rather, the goal is to optimize the overall revenue by accepting bundle transfer requests under the assumption of minimally cooperative (nonrational) behaviors of nodes [5, 57].

3.4.2 State Variable and Action Variables

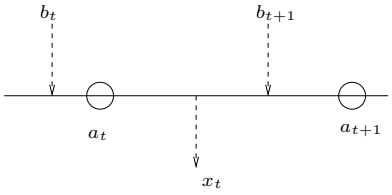


Figure 3.2: State transition relation

We define the state variable a_t to be the remaining capacity at time t. Since we assume that a request arrives at a node at time t and drives the decision epoch, departures can occur after the decision epoch during opportunistic contact between the current custodian and its future custodian. The remaining capacity in the current node at decision epoch $t + 1$ may include the space released by bundles transferred to the next custodian during opportunistic contacts. Let the request size be x_t at decision epoch t, and the released space available for next decision epoch $t+1$ be b_{t+1}. The transfer relation shown in Fig. 3.2 can be described by the following transfer function.

$$a_{t+1} = a_t - u_t x_t + b_{t+1} \tag{3.2}$$

where u_t has been defined in (2.1); $b_{t+1} = 1$ if the bundle is successfully transferred to the next custodian, $b_{t+1} = 0$ otherwise; $x_t = 1$.

As we have said before, storage space may be released in time duration between decision epoch t and decision epoch $t+1$ due to message departure. Since the released space is only beneficial to arrivals at decision epoch $t + 1$ and later, b_{t+1} is indexed with $t + 1$.

3.4.3 Opportunity Cost and Benefit Function

Our goal is to choose the optimal strategies for maximizing the expected sum of benefits. Toward this, we need to consider two conflicting forces. First, it is wasteful to commit resources to requests that are not "desperate" for that resource, i.e., not enjoying the maximal possible benefit from occupying the resource. Second, it is equally dangerous to gamble that each resource can be occupied with maximal benefit gained without knowing the sequence of requests coming in the future. The key aspect of the above situations is that each decision can not be viewed in isolation since one must balance the desire for high benefit request with the undesirability of low future benefit request.

We use *opportunity cost* and *benefit function* to balance the above two conflicting forces. The opportunity cost measures the value of the storage capacity, which is the benefit that may be lost by higher benefit request as a result of consumption of the above resource by the lower benefit request. Theoretically, the opportunity cost can be captured by defining a *value function* $V_t(\cdot)$, which measures the optimal expected benefit as a function of the remaining capacity a_t at time t [51]. The opportunity cost is then the difference between the value function at a_t and the value function at $a_t - 1$, that is, $V_t(a_t) - V_t(a_t - 1)$. Obviously, the theoretical analysis of the optimal control strategies will heavily rely on the value function $V_t(\cdot)$.

On the other hand, the benefit function, as we have discussed before, denotes the gain of moving a bundle to the next hop. It can be defined according to users' specifications in the systems (for example as a function of the bundle size and priority). While there is no strict limitation to choose benefit function, the choice of benefit function should guarantee that the value of benefit function and opportunity cost are comparable.

3.5 Single Node Congestion Control

In this section, we will focus on the decision for custody acceptance/rejection for a single node. The case for multiple nodes will be studied in the next section. Toward this end, we present a dynamic programming approach to the problem formulated in the previous section.

Dynamic programming can handle situations where decisions are made at decision epochs. The outcome of each decision may not be fully predictable but can be anticipated to a certain extent before the next decision is made. The objective usually is to maximize a certain reward. A key aspect of such a situation is that a decision can not be viewed in isolation since one must balance the desire for high present reward with the undesirability of low future rewards. Dynamic programming can capture this tradeoff. At each decision epoch, the dynamic programming technique ranks decisions based on the sum of the present reward and the expected future reward, assuming optimal decision making for subsequent decision epochs [54]. Our technique will follow this storyline as well.

Recall that the value function $V_t(\cdot)$ denotes the value of the remaining capacity at time t, that is, the value of remaining capacity at time t is $V_t(a_t)$. We assume that

the probability of an arrival of class ℓ at time t is denoted by $p_\ell(t)$, and *at most one request arrives in one decision epoch.* Subsequently, we easily yield

$$\sum_{\ell=1}^{m} p_\ell(t) \leq 1 \tag{3.3}$$

Evidently, the arrival probability may vary with time t, and hence the mix of classes that arrive may vary over time. This varying probability, as we will see later, will not affect the optimal control strategies, but it will incur extra computation.

For now, let us assume that a decision epoch will be driven by a request of one message, that is, at most one message request arrives in one decision epoch (the case for one request of multiple message custody transfer will be discussed in 3.5.2).

Let r_t be a random variable, with $r_t = B_\ell$ if a request for class ℓ arrives at decision epoch t, and $r_t = 0$ otherwise. Note that the probability $\mathbb{P}(r_t = B_\ell) = p_\ell(t)$. Our goal is to maximize the sum of the current reward and the reward to go, i.e.,

$$\max_{u\in\{0,1\}} \Big\{ r_t u + V_{t+1}(a_t - u x_t + b_{t+1}) \Big\} \tag{3.4}$$

Since $x_t = 1$, we can rewrite (3.4) as

$$\max_{u\in\{0,1\}} \Big\{ r_t u + V_{t+1}(a_t - u + b_{t+1}) \Big\} \tag{3.5}$$

Intuitively, the above objective function targets at achieving a balance between the current reward ($r_t = B_\ell$) to accept a request and the potential value (V_{t+1}) of the available buffer space (i.e., remaining capacity in the buffer).

Let

$$\hat{V}_{t+1}(a_t) = V_{t+1}(a_t + b_{t+1}) \tag{3.6}$$

The optimal value functions $V_t(a_t)$ for each decision epoch (or time period) t can be written as follows.

$$V_t(a_t) = \mathbb{E}\left[\max_{u\in\{0,1\}}\{r_t(t)u + \hat{V}_{t+1}(a_t - u)\}\right] \tag{3.7}$$

Given the finite time horizon being considered, the boundary conditions here are

$$V_t(0) = 0, \;\; t = 1, \ldots, T.$$

and

$$V_T(a_T) = r_T.$$

Here r_T stands for a salvage reward for the remaining amount of resource at the end of the time horizon. If $a_T = 0$ then $r_T = 0$; if $a_T \neq 0$, we assume that r_T is a concave function of the remaining capacity a_T such as piece-wise linear function. This assumption will guarantee that the value function $V_t(\cdot)$ is also concave.

3.5.1 Optimal Strategy for Accepting Custody Transfer

In this subsection, our goal is to prove the following theorem regarding the optimal policy for the congestion control.

Theorem 3.1 *For a class j request with reward $r_t = B_\ell$ arriving at decision epoch t, it is optimal to accept the request if and only if*

$$r_t \geq \Delta\hat{V}_{t+1}(a_t) \tag{3.8}$$

where $\Delta\hat{V}_{t+1}(a_t) = \hat{V}_{t+1}(a_t) - \hat{V}_{t+1}(a_t - 1)$

This theorem actually denotes that for a request, if the benefit (r_t) is greater than its opportunity cost ($\Delta\hat{V}_{t+1}(a_t)$), the message shall be accepted and the decision actually is optimal.

To prove the above theorem, we first prove Lemma 3.1-3.3 that will be employed to deduce expression suitable for proving Theorem 3.1.

Lemma 3.1 *If x, y are integers, and $x \geq y$, we have*

$$\hat{V}_t(x) = \hat{V}_t(x-y) + \sum_{k=x-y+1}^{x} \Delta\hat{V}_t(k)$$

where $\Delta\hat{V}_t(k) = \hat{V}_t(k) - \hat{V}_t(k-1)$.

$$\begin{aligned} \hat{V}_t(x) &= \hat{V}_t(x) - \hat{V}_t(x-1) + \hat{V}_t(x-1) \\ &= \hat{V}_t(x) - \hat{V}_t(x-1) + \hat{V}_t(x-1) - \hat{V}_t(x-2) \\ &\quad + \cdots - \hat{V}_t(x-y) + \hat{V}_t(x-y) \\ &= \hat{V}_t(x-y) + \sum_{k=x-y+1}^{x} \Delta\hat{V}_t(k) \end{aligned} \tag{3.9}$$

Lemma 3.2

$$\begin{aligned} \max_{u\in\{0,1\}} \{r_t u + \hat{V}_{t+1}(x-u)\} &= \hat{V}_{t+1}(x) \\ &\quad + \max_{u\in\{0,1\}} \{(r_t - \Delta\hat{V}_{t+1}(x))u\} \end{aligned}$$

where $\Delta\hat{V}_{t+1}(x) = \hat{V}_{t+1}(x) - \hat{V}_{t+1}(x-1)$ is the expected marginal value of remaining capacity in decision epoch $t+1$.

From Lemma 3.1, it is apparent that

$$\hat{V}_{t+1}(x) = \hat{V}_{t+1}(x-y) + \sum_{k=x-y+1}^{x} \Delta\hat{V}_{t+1}(k) \tag{3.10}$$

In equation (3.10), note that if $y = 0$, the last summation disappears. Let $y = u$, then we have

$$\hat{V}_{t+1}(x-u) = \hat{V}_{t+1}(x) - \sum_{k=x-u+1}^{x} \Delta\hat{V}_{t+1}(k) \tag{3.11}$$

As u can be 0 or 1, (3.11) can be changed to

$$\hat{V}_{t+1}(x-u) = \hat{V}_{t+1}(x) - \Delta\hat{V}_{t+1}(x)u \tag{3.12}$$

By (3.12), the expression

$$\max_{u\in\{0,1\}} \{r_t u + \hat{V}_{t+1}(x-u)\} \tag{3.13}$$

can be rewritten as

$$\begin{aligned}
&\max_{u\in\{0,1\}} \{r_t u + \hat{V}_{t+1}(x-u)\} \\
&= \max_{u\in\{0,1\}} \{r_t u + \hat{V}_{t+1}(x) - \Delta\hat{V}_{t+1}(x)u\} \\
&= \hat{V}_{t+1}(x) + \max_{u\in\{0,1\}} \{(r_t - \Delta\hat{V}_{t+1}(x))u\}
\end{aligned}$$

By Lemma 3.2, the Bellman equation (2.12) can be rewritten as

$$\begin{aligned}
V_t(a_t) &= \mathbb{E}\left[\max_{u\in\{0,1\}} \{r_t(t)u + \hat{V}_{t+1}(a_t - u)\}\right] \\
&= \hat{V}_{t+1}(a_t) + \mathbb{E}\left[\max_{u\in\{0,1\}} \{(r_t - \Delta\hat{V}_{t+1}(a_t))u\}\right]
\end{aligned} \tag{3.14}$$

And as a result, we can now prove Theorem 3.1. We will use mathematical induction to prove Theorem 3.1. In order to prove the above theorem, it is necessary to investigate the concavity of the value function $V_t(\cdot)$ and $\Delta V_t(\cdot)$. To do so, we need the Definition 3.1 and Lemma 3.3.

Definition 3.1 *A function defined on the set of nonnegative integers $g : \mathcal{Z}_+ \to \mathcal{R}$ is concave if it has nonincreasing differences, that is, $g(x+1) - g(x)$ is nonincreasing in $x \geq 0$.*

The above definition can be considered as a discrete version of concave definition of a continuous function defined on $\mathcal{R}$. Additionally, we also present the following lemma from [51].

Lemma 3.3 *Suppose $g : \mathcal{Z}_+ \to \mathcal{R}$ is concave. Let $f : \mathcal{Z}_+ \to \mathcal{R}$ be defined by*

$$f(x) = \max_{a=0,1,\ldots,m} \{ap + g(x-a)\}$$

for any given $p \geq 0$ and nonnegative integer $m \leq x$. Then $f(x)$ is concave in $x \geq 0$ as well.

Interested readers are referred to [51] for details on the proof of Lemma 3.3. Using Lemma 3.3, we will prove Theorem 3.1 below.

Proof of Theorem 3.1:

We first prove that the function $V_t(x)$ is concave in x. The proof is by induction at decision epochs. Note that in the terminal decision epoch T, there are two possibilities: the first one is that $a_T = 0$, then $V_T = 0$; the second one is that $a_T \neq 0$ and then $r_T \neq 0$. For the second case, we assume that the remaining capacity receives a salvage reward that is concave in a_T. For the above two cases, $V_T(a_T)$ is concave in a_T.

Assume $V_{t+1}(x)$ is concave in x at decision epoch $t+1$. Since $\hat{V}_{t+1}(x) = V_{t+1}(x+b_{t+1})$, it is also concave at decision epoch $t+1$. By Lemma 3.3, we can easily know that

$$\mathbb{E}\left[\max_{u\in\{0,1\}}\{r_t(t)u + \hat{V}_{t+1}(a_t - u)\}\right]$$

is concave because it is simply the expectation of $\max_{u\in\{0,1\}}\{r_t(t)u + \hat{V}_{t+1}(a_t - u)\}$ based on the probability $\mathbb{P}(r_t = B_\ell) = p_\ell(t)$. Therefore, we conclude that $V_t(x)$ is concave.

We can also prove that $\Delta\hat{V}_t(x) \geq \Delta\hat{V}_{t+1}(x)$ by way of induction. The intuition of this inequality is that the value of additional capacity at any point in time has a decreasing marginal benefit and the marginal value at any given remaining capacity x decreases with time.

From the above argument, we can know that the optimal value can be achieved if and only if $r_t \geq \Delta\hat{V}_{t+1}(a_t)$.

Theorem 3.1 guarantees that if the node follows the accepting condition $r_t \geq \Delta\hat{V}_{t+1}(a_t)$, the node will have best rewards in finite time periods.

3.5.2 Discussion

In this subsection, we first analyze the properties of setting a fixed opportunity cost, then discuss the case of one request for multiple message custody transfer.

Note that $\Delta\hat{V}_{t+1}(a_t)$ is a function of a_t, that is, the opportunity cost dynamically varies with remaining capacity of a_t. The opportunity cost increases as the remaining capacity decreases. While setting a fixed opportunity cost to accept or reject a request is simple, to be effective, the opportunity cost must be updated after accepting a request. Without this ability to make opportunity cost a function

of capacity, however, a simple static opportunity cost is indeed a dangerous form of control [51].

In the above discussion, for the sake of simplicity, we have also assumed that only one request arrives at each decision epoch. Indeed, the approach can be easily generalized to the scenario where multiple requests can arrive at the same decision epoch. If a message contains custodian transfer request for multiple bundles, the receiving node may decide to partially accept any quantity q in the range $0 \leq q \leq Q$ given a request of $Q \geq 1$ units. In this case the analysis in 3.5.1 still holds by serializing the unit requests and applying the above steps. In other word, the group arrival drives a decision epoch, accepting node then serializes all requests contained in the group arrival with a certain rule. The optimal control strategies can decide whether each request can be accepted or not one by one based on the dynamic opportunity costs.

However, a message may contain custodian transfer requests for multiple bundles and the request must be satisfied in an *all-or-none basis*. In other words, given a request containing $Q > 1$ bundles, all Q bundles or none must be accepted. In this case, the value function may not be concave. The marginal value of capacity may actually increase [51], and the congestion management strategies developed in Section 3.5.1 may not be optimal. To address this issue, we first must specify the distribution of group sizes to model how much demand we have from groups of various sizes. This in fact does not increase the theoretical difficulty. The difficulty lies in that the value function may not be concave which can make the optimality issue intractable [51, 66].

We also remark that the benefit function itself does not address the issue about how the message is forwarded. On the contrary, the route itself shall be decided by the routing algorithms. In other words, we rely on the routing algorithm to prevent the loops or tossing back of messages for artificial benefit inflation.

3.6 Network Congestion Control

In the above section, we have studied the optimal congestion control policy for a single node case. In this section, we first extend dynamic programming based approach in Revenue Management (RM) to network capacity control with multiple resources, then develop the congestion management strategies for delay tolerant networks with multiple nodes.

3.6.1 Optimal Policy with Global Information

Suppose that the network has n nodes and there are m requests. Each request may need a combination of resources on the n nodes. Define an *incident matrix* $\mathbf{A} = [a_{lh}]_{n\times m}$, where $a_{lh} = 1$ if resource on node l is used by request h and $a_{lh} = 0$ otherwise. As a result, the h^{th} column of $\mathbf{A}$, denoted by $\mathbf{A}_h$, is the incidence vector for request h; the l^{th} row, denoted by $\mathbf{A}^l$, has an entry of one in column h corresponding to a request h that uses resource on node l. If the network topology is fixed and routing path is predefined, we can easily construct the incidence matrix $\mathbf{A}$ with appropriate dimensions based on schemes that the requests use the resources.

Demand in period t can be modeled as the realization of a single random vector $\mathbf{B}(t) = (B_1(t), \ldots, B_m(t))$. If $B_h(t) = B_h > 0$, a request h arrives and the benefit to accept it is B_h; if $B_h(t) = 0$, then there is no request with type of h. A realization $\mathbf{B}(t) = 0$ means that there is no request at time period t.

The state of the network can be described by $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))$, where $x_l(t)$ is the remaining buffer capacity of resource on node l at time period t. Let $u_h(t)$ be the decision variable for a request at time period t. $u_h(t) = 1$ if request h is accepted in time period t, $u_h(t) = 0$ otherwise. The decision to accept, $u_h(t)$, is a function of the remaining capacity vector $\mathbf{x}(t)$ and benefit B_h of request h. In other words, $u_h(t) = u_h(t, \mathbf{x}, B_h)$. Since we can accept at most one request in any

period, if the current remaining capacity is $\mathbf{x}(t)$, then the following condition should be satisfied:

$$\mathbf{x}(t) \geq \mathbf{A}_h.$$

Similar to Section 3.5, let $V_t(\mathbf{x})$ denote the optimal value function for networks. Then $V_t(\mathbf{x})$ satisfies the following Bellman equation.

$$V_t(\mathbf{x}) = \mathbb{E}\left[\max_{u_h \in \{0,1\}} \left\{ B_h(t)u_h + \hat{V}_{t+1}(\mathbf{x} - \mathbf{A}_h u_h) \right\}\right] \tag{3.15}$$

Appropriate boundary conditions for the above Bellman equation can be set at decision epoch T provided that the salvage benefit at decision epoch T is concave. Subsequently, as in 3.5.1, we can deduce the optimal control strategy form the above equation as follows.

$$u_h(t, \mathbf{x}, B_h) = \begin{cases} 1 & \text{if } B_h \geq \hat{V}_{t+1}(\mathbf{x}) - \hat{V}_{t+1}(\mathbf{x} - \mathbf{A}_h) \text{ and} \\ & \quad \mathbf{x}(t) \geq \mathbf{A}_h, \\ 0 & \text{otherwise.} \end{cases} \tag{3.16}$$

The idea behind the above optimal control policy is that accepting a request h with benefit B_h if and only if there are sufficient remaining capacities in relevant resources and benefit is greater or equal to the opportunity cost to occupy the storage spaces.

The structure of the value function for network case (3.15) is similar to the value function for the single node (2.12). However, this is partially resulted from the assumption that global information is available and centralized calculation can be performed. This is most likely impractical for delay tolerant networks where the network is potentially highly dynamic in almost every aspect.

3.6.2 Distributed Optimal Policy

The above analysis follows the classic revenue management with an unrealistic assumption on the availability of global information. Furthermore, even if the global information were available, the potential large dimension of the state space would often render the solution hopeless in practice. Fortunately, the unique setup of delay tolerant networks naturally provides an approach toward a distributed solution.

The key for a distributed solution is the independence of the resource requirement at each node in the network. In a conventional resource management strategy, the resources are requested simultaneously at multiple nodes in order to pave the end-to-end path. This is actually reflected in the above analysis when global information is available. In such a scenario, if one of the nodes could not fulfill the request, the request actually will be rejected. Fortunately, for delay tolerant networks, end-to-end path is often not present and hop-by-hop forwarding and control is employed. Once the bundle is transferred to another hop, the resource originally occupied will become immediately available. Whether this transaction will be executed will solely dependent on the receiving node.

Based on above discussion, a request in delay tolerant networks can be expressed as $A_j = (0, \ldots, 0, 1, 0, \ldots, 0)$, where A_j is a request vector that resource is requested on node j. Without losing generality, we assume that the value function $V_{t+1}(\mathbf{x})$ has a gradient $\nabla V_{t+1}(\mathbf{x})$. In other words, the value function $V_{t+1}(\mathbf{x})$ is differentiable with respect to vector $\mathbf{x}$. Formally,

$$\begin{aligned} & V_{t+1}(\mathbf{x}) - V_{t+1}(\mathbf{x} - A_j) \\ & \approx \nabla V_{t+1}^T(\mathbf{x}) A_j \\ & = \sum_{i \in A_j} \pi_i(t, \mathbf{x}) = \pi_j(t, \mathbf{x}) \end{aligned} \tag{3.17}$$

where $\pi_j(t, \mathbf{x}) = \frac{\partial}{\partial x_j} V_{t+1}(\mathbf{x})$ is the *opportunity cost* of resource j at decision epoch $t+1$.

From the above equation, we can see that the opportunity cost evidently depends on the format of the utility function. Assume that $V_{t+1}(\mathbf{x})$ has the following format

$$V_{t+1}(\mathbf{x}) = \sum_{i=1}^{n} V_{t+1}^{i}(x_i) \tag{3.18}$$

where $V_{t+1}^{i}(x_i)$ is the value function of resource i at decision epoch $t+1$. Then, from (3.17) and (3.18), we have

$$\begin{aligned} &\pi_j(t, \mathbf{x}) \\ &= \frac{\partial}{\partial x_j} V_{t+1}(\mathbf{x}) = \frac{\partial}{\partial x_j} \sum_{i=1}^{n} V_{t+1}^{i}(x_i) \\ &\approx V_{t+1}^{j}(x_j) - V_{t+1}^{j}(x_j - 1) \end{aligned} \tag{3.19}$$

From (3.19), the opportunity cost of resource j at decision epoch $t+1$ only depends on its remaining capacity.

In the above discussion, we assume that the value function $V_{t+1}(\mathbf{x})$ is differentiable with respect to vector $\mathbf{x}$. If the value function $V_{t+1}(\mathbf{x})$ is not differentiable with respect to vector $\mathbf{x}$, gradient $\nabla V_{t+1}(\mathbf{x})$ can be replaced by subgradient, it will not affect the outcome except by introducing extra computation [56].

From Theorem 3.1 and (3.19), the congestion control policy in a node j is as follows.

$$r_t \geq \Delta \hat{V}_{t+1}^{j}(a_t^j) \tag{3.20}$$

where $\Delta \hat{V}_{t+1}^{j}(a_t^j) = \hat{V}_{t+1}^{j}(a_t^j) - \hat{V}_{t+1}^{j}(a_t^j - 1)$. This policy can achieve network level optimality given the value function presented (3.18).

One reason to choose the value function such as (3.18) is that we consider delay tolerant networks where resources maximize a common additive value. Each resource has information only about its value component, and maximizes that component while exchanging information between any two resources only during their contact opportunity.

If we choose a general value function other than the format of (3.18), the key point is still on how to compute the opportunity cost. From (3.17), we can know that the opportunity cost for node j at decision epoch $t+1$ depends on the remaining storage capacities in other nodes. Since it is not practical for each node to have information about other nodes in delay tolerant networks, this is another reason that we employ the value function as in (3.18).

For general value functions, we can use the decomposition approach to decompose the general value function into the format in (3.18). The decomposition is at the expense of losing some network information.

3.7 Simulation

We developed a discrete event-driven simulation based DTN simulator to evaluate our congestion management strategy [82]. The simulation implements the congestion management strategy as proposed in the previous section. To isolate the effect of link bandwidth on the congestion management strategy, we assume that each link has infinite bandwidth.

3.7.1 Simulation Settings

The simulated network consists of static nodes, destination, and mobile nodes distributed in a 3000 by 3000 meters field. Static nodes are randomly distributed in the field and generate messages following poisson processes. Each static node can

Table 3.1: Simulation Parameters

Parameter	Value
Simulation field size	3000×3000 (m^2)
Transmission range	150 m
Average arriving rate of a request	0.2/second
Number of static nodes	6/8
Number of mobile nodes	40/50

generate five classes of request messages, each with average generation probability of 0.2 message/second. Mobile nodes function as relay nodes and their mobility follows the random-way-point model with random initial location as well. The random way point model employed for mobile nodes has a moving speed uniformly distributed in [0.2, 0.5] meters/s and the pause time of a stop is uniformly distributed in [1, 2] seconds. The destination node has unlimited storage capacity and is randomly located in the field and ready to accept messages during opportunistic contacts. The storage capacity in each mobile node has a size of 50 messages. Since messages have to traverse lower layers of the network, they are ultimately subject to the restrictions there in terms of maximum packet size. For example, on most IP networks it is safest to assume that a single packet should be less than 1500 bytes long. Therefore, we assume that each message has a size of 1500 bytes [69]. We consider two scenarios with different mobile/static node mix: scenario 1 with 40/6 mix and scenario 2 with 50/8 mobile/static mix, respectively. The parameters are summarized in Table 3.1.

We assume that there is an oracle for message routing. The oracle knows everything and can distribute routing information around the network [69]. Notice that the oracle is only responsible for message routing. Congestion control in each node is addressed by congestion management strategy.

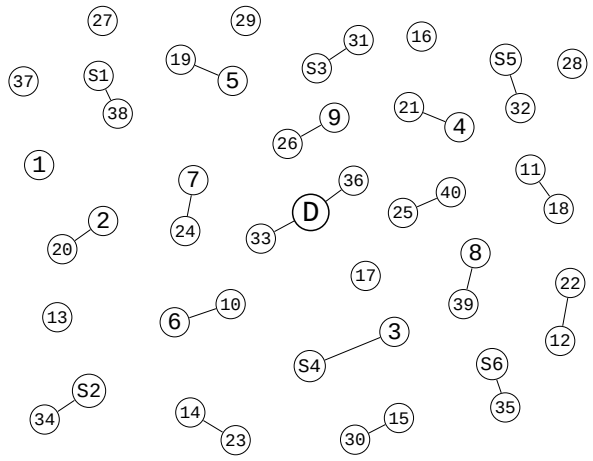


Figure 3.3: Nodes' position snapshot at 600s for 40/6 node mix for case listed in Fig. 3.5a.

Fig. 3.3 shows a snapshot of the network. Here, the destination is marked as D, static nodes are marked $S1 - S6$, mobile nodes are indexed 1-40, and the lines stand for existing links between nodes.

The main performance metrics of congestion management strategies in the simulation are throughput of the simulated network and the buffer utilization in each node. In order to evaluate the performance of our proposed scheme, we compare the congestion management strategy with dynamic opportunity cost with the one with static opportunity cost under the same routing scheme. We employ the function $w \log x$, where w is an adjustable weight, x is the remaining capacity of a node, to compute the salvage reward of remaining amount of capacity, the reason to choose such a function is to guarantee that the salvage reward is concave in the remaining amount of capacity at the end of the time horizon.

3.7.2 Simulation Results and Discussions

As we have discussed, we separate the congestion management mechanism from the route selection problem. The routing algorithm is based on the oracle in the sys-

tem. Fig. 3.4 shows the distribution of hop-count of messages of two different mobile node/static node mixes under different congestion policies (scheme with dynamic opportunity cost and scheme with static opportunity cost). From Fig. 3.4, we can see that there is no significant difference in hop count among several simulation scenarios.

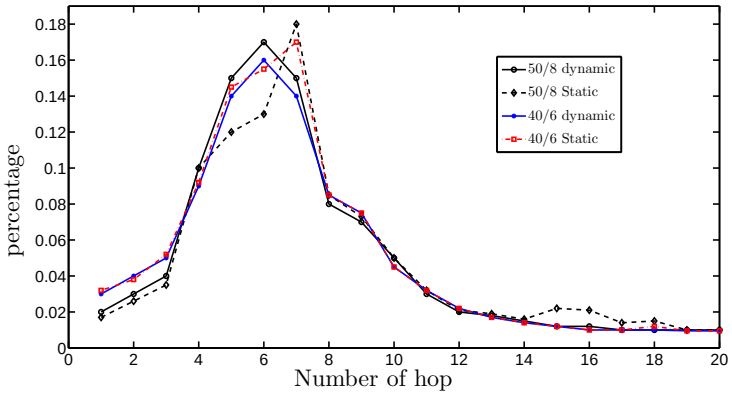
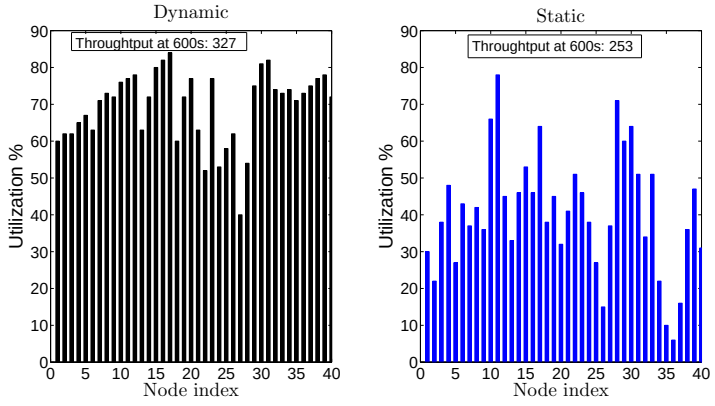


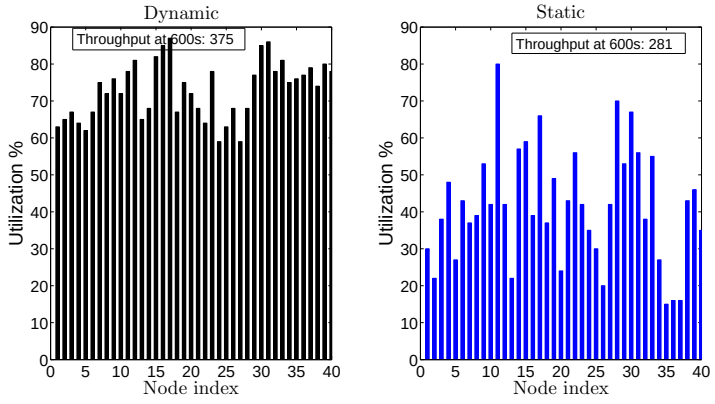
Figure 3.4: Hop count distribution

Fig. 3.5 shows snapshots of buffer utilization under the two control policies with 40/6 node mix at 600s. Fig 3.5a is the snapshot of buffer utilization with arrival density $\lambda_1 = 1/2$ of 6 poisson processes. Fig. 3.5b is the snapshot with message arrival density $\lambda_1 = 5/9$ of 6 poisson processes. Fig. 3.6 shows the snapshots of buffer utilization with 50/8 node mix at 600s. Fig. 3.6a is the snapshot of buffer utilization with arrival density $\lambda_1 = 1/2$ of 8 poisson processes. Fig. 3.6b is the snapshot with message arrival density $\lambda_1 = 5/9$ of 8 poisson processes.

From Fig. 3.5 and 3.6, we can see that control policy with dynamic cost can achieve much more evenly balanced loads and higher utilization in all the nodes of the network and better throughput. Since a node can not predict the coming request in advance, if the opportunity cost is fixed it is possible to reject the request even if



(a)

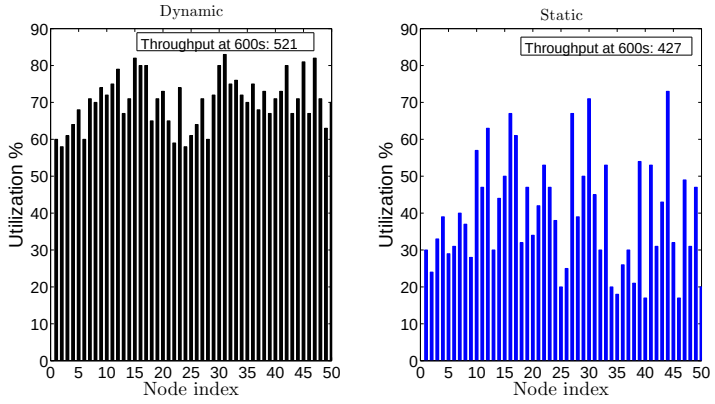


(b)

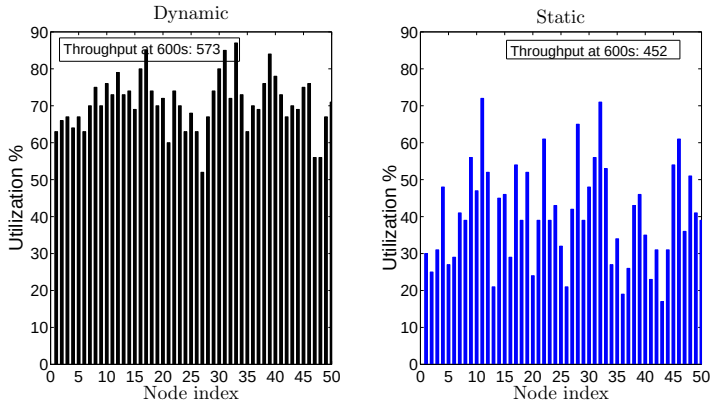
Figure 3.5: Load distribution in nodes - 40/6 node mix. (a) message generation rate $\lambda = 1/2$ message/second. (b) $\lambda = 5/9$ message/second.

the utilization is still low. Therefore it is a dangerous control to set a fixed cost to obtain optimization solution in revenue and utilization [51]. On the other hand, a dynamic policy can adapt the opportunity cost in a node based on varying storage space and the space usage can be optimized.

Fig. 3.7 shows node utilization and throughput for 40/6 node mix at 600s, where simulation conditions are the same as those of Fig. 3.5a except that traffics are generated at a constant rate (0.5 message/second) from the static nodes. From Fig. 3.7, we can see that the congestion control policy with dynamic opportunity cost can achieve better buffer utilization and throughput than the one with static opportunity



(a)



(b)

Figure 3.6: Load distribution in nodes - 50/8 node mix. (a) message generation rate $\lambda = 1/2$ message/second. (b) $\lambda = 5/9$ message/second.

cost even in this extreme case. For other simulation scenarios, we have very similar results.

3.8 Conclusion

In this chapter, we have developed an optimal congestion management strategy for delay tolerant networks based on the concept of revenue management and dynamic programming. Relying only on the information of local storage space, our scheme can be readily applied to the dynamic and often unpredictable environments of delay tolerant networks. Our simulation results show that the proposed conges-

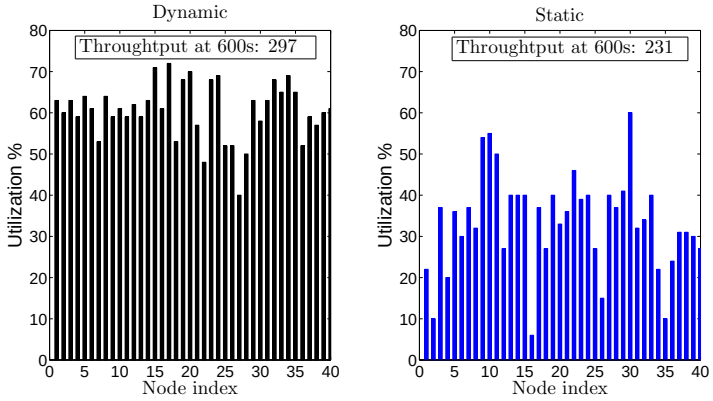


Figure 3.7: Load distribution in nodes - 40/6 node mix (traffic generated at constant speed 0.5 message/s).

tion management strategy can effectively outperform a simple static, threshold based scheme.

CHAPTER 4
REPEATED GAME MODELING OF CONGESTION MANAGEMENT IN INTERMITTENTLY COMMUNICATING NETWORKS

4.1 Introduction

In the previous chapter, we developed congestion control strategies for delay tolerant networks based on the concepts of revenue management and dynamic programming. We assumed that the time horizon is finite in making the decision of resource allocation. However, in practice, in certain situations, it might be difficult or impossible to predict when the dynamic behavior will stop. As an alternative solution, we employ repeated games to model the decision making for custody transfer in this chapter.

Since delay tolerant networks are dynamic in nature, they may have no fixed topology, where nodes arrive, leave and move away changing neighborhood, every node can not be sure that it is going to communicate with different opponents next round. Even if the topology is fixed, since the communication between any two nodes is intermittent, there is no knowledge when the dynamic behavior is going to stop. This kind of communication patterns can be modeled by repeated games. Repeated games can describe situations in which the number of communication rounds is infinite and finite (even only once) and capture the dynamic interacting behaviors of selfish nodes over time [94].

The power of the repeated game model is that it allows to design simple mechanisms that enforce cooperation in network lifetime even if there is no knowledge on how long the network life time is. Our goal is to maximize each node's payoff while respecting the resource constraint of each node.

The main contribution of this chapter is to employ repeated games to model the decision making for custody transfer and propose a new congestion control strategy.

This chapter is organized as follows. In Section 4.2, we investigate the related work. Section 4.3 briefly presents some basic knowledge of repeated game theory. The congestion management problem is reformulated in Section 4.4. Section 4.5 studies the custody transfer game. We propose a congestion control strategy and analyze it in Section 4.6. In Section 4.7, we use several simulation scenarios to show the performance of the proposed control strategy. Finally, we conclude this chapter in Section 4.8.

4.2 Related Work

Game theory has been used to model selfish behaviors on achieving socially desirable equilibria in networks [88–93, 96, 97].

In [92], Nurmi used a bayesian game to model energy constrained routing in selfish ad hoc networks, discussed the structure of strategies and proposed a method that allows the nodes to learn equilibrium strategies over time. Urpi *et al* developed a model, based on bayesian game, capable of formally explaining characteristics of ad hoc networks (the nodes's selfishness or the network mobility), this model allows to formally study and analyze strategies for cooperation [89].

Many peer-to-peer systems rely on cooperation among self-interested users. For example, users of file-sharing systems who don't share their own resources cause long delay or download failures. When non-cooperative users benefit from free-riding on others' resources, the "tragedy of the commons" is inevitable. Avoiding this problem requires an incentive for cooperation. In [90], the authors used an infinitely repeated game to model peers' interactions, in their model, all peers are server and client in a game, a set of demand relationships among the peers in the network is given

exogenously, and remains constant throughout. Lai *et al* employed evolutionary game to capture the dynamic behavior in peer-to-peer networks, in each game, one player is the client and one player is the server [91].

In [96], the authors studied optimal routing using repeated game theory, and investigated the existence of a Nash equilibrium point that achieves the system-wide optimum cost. In [86], Afergan used a repeated game to design incentive-based routing systems, and viewed the exchange of pricing information at an interconnect as a repeated game between the relevant players. Afergan *et al* also used repeated game to model user behavior and studied the practical tradeoff between a user's short-term desire for quality and long-term desire for the network's continued existence [87].

In static networks, game theory has long been used to model the routing decisions of networks. However, once we move to dynamic and resource constrained settings, such as sensor networks, delay tolerant networks, we need to seek new models that capture the dynamic nature of the decisions and the resource constraints of the networks are needed.

4.3 Basics of Repeated Game

In this section, we first introduce the repeated game theory, then briefly describe dynamic game, a variation of repeated game.

When players interact by playing a similar static game many times, the game is called a repeated game. The building block of a repeated game, the game which is repeated, is called the stage game. To define the repeated game, we must specify the players' strategy space and payoff functions. Unlike a game played once, a repeated game allows a strategy to be contingent on past moves, thus allowing reputation effects and retribution, which give possibility for cooperation [94, 95].

Table 4.1: Payoffs for the prisoners' dilemma

		Player 2	
		C	D
Player 1	C	$(1,1)$	$(-1,2)$
	D	$(2,-1)$	$(0,0)$

Consider, for example, the canonical example of the Prisoners' Dilemma. In this game we have two players. The action space of this game is simple - each player can either cooperate (C) or defect (D), and the players move simultaneously. Based on their actions, each player receives a payoff as given by the matrix in Table 4.1. In the matrix, each cell represents the payoff of a particular pair of actions. For example, if both players play C then both get a payoff of 1, or $u_1(C,C) = u_2(C,C) = 1$. If however, player 1 plays D but player 2 plays C, then the payoff of player 1 is $u_1(D,C) = 2$.

In the one-shot game, both players will play D. Regardless of what the other does, it is always in the best interest for a particular player to defect. The only Nash Equilibrium (NE) of this game is therefore (D,D). This also holds when the number of rounds is finite and known, as reverse induction shows that each stage game is equivalent to the one shot game.

When the number of rounds is infinite or unknown, other equilibrium outcomes are possible. It is standard to use a discount factor to capture the fact that future payments are less valuable. Typically, this factor is represented by $\delta\,(0 < \delta < 1)$, and can - for example - represent the time-value of money. Here $\delta = 1$ represents perfectly patient players whereas $\delta = 0$ represents perfectly impatient players. It is interesting to note that both cases can be analyzed in the same fashion, using δ as a parameter to understand the space between the extremes.

For example, assume that Player 2 plays the following strategy:

1. Play C,
2. If Player 1 ever plays D, then Play D forever.

Now look at player 1 in either case, fixing the strategy of Player 2 as above. If she cooperates she receives $u_1(C, C) = 1$ forever. However, if she deviates, she obtains $u_1(D, C) = 2$ once and then $u_1(D, D) = 0$ for the rest of the game.

The sample strategy will be an equilibrium strategy if and only if the payoffs of playing the strategy are greater than or equal to the payoffs of deviating from the strategy and suffering the consequences. Again, we model player's preferences over streams of payoffs by discounting the payoffs with a decaying parameter δ. Using the above values, we can determine whether or not the sample strategy is an equilibrium strategy by comparing the payoffs to various actions.

While in the finitely repeated game cases, a strategy can explicitly state what to do in each of the T periods, specifying strategies for infinitely repeated games is trickier because it must specify actions after all possible histories, and there is an infinite number of these. Here are the specifications of several common strategies.

- **Always Defect (ALL-D)**. This strategy prescribes defecting after every history no matter what the other player has done.
- **Always Cooperate (ALL-C)**. This strategy prescribes cooperating after every history no matter what the other player has done.
- **Tit-for-Tat (TFT)**. This strategy prescribes cooperation in the first period and then playing whatever the other player did in the previous period: defect if the other player defected, and cooperate if the other player cooperated.

The detailed discussion about repeated game theory is referred to reference [83, 94].

A dynamic game is a specific kind of repeated game. It allows the possibility that the stage game changes from period to period for a fixed set of players, possibly

randomly as a function of the history of play [94]. The analysis of a dynamic game typically revolves around a set of game states that describe how the stage game varies from period to period. Each state determines a stage game, captured by writing payoffs as a function of states and actions. The specification of the game is completed by a rule for how the state changes over the course of play. In many applications, the context in which the game arises suggests what appears to be a natural candidate for the set of states. The detailed discuss about dynamic game is referred to [94].

While the repeated games are the well-studied dynamic games, the control strategies developed for them are very complicated [83, 94, 95]. We will employ repeated single-decision games to model decision-making for custody transfer in delay tolerant networks.

4.4 Problem Formulation

In this section, we reformulate the problem based on the one formulated in Section 3.4 so that the problem formulation is suitable for description by repeated single-decision games.

We assume that there are n nodes in a delay tolerant network. Each node can not be sure that it is going to communicate the next round with different nodes since the communication is intermittent and the topology may be dynamic. For each node, there may be no knowledge when the communication with other nodes is going to stop. The above situations can also happen in a mobile ad hoc network, where nodes arrive, leave and move away changing neighborhood [89].

If the communication is present between any two nodes, the nodes may communicate to transfer messages if allowed. Figure 4.1 shows a simple communication scenario of two nodes in a delay tolerant network. Node j may send a message transfer request to node i when there exists communication opportunity between them.

Node i makes a decision whether to accept or reject the request from node j based on its occupied storage space and request reward. We here assume that the storage space in each node is limited, and it is the key resource constraint in our problem.

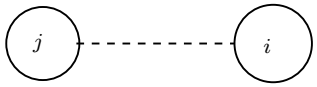


Figure 4.1: Custody transfer of two players

By accepting a message, a node accumulates a certain amount of payoff. The payoff is defined as the difference between request reward and custodian cost. We suppose that each message transfer request has a certain reward, which is denoted by B_ℓ, where $B_\ell > 0$, $\ell \in \{1, \ldots, m\}$ is the index of the class of rewards. The reward can be of various forms and correspondingly different optimization goals can be achieved. For example, by properly adjusting reward, performance issues such as throughput and fairness can be addressed [59].

Notably, reward can be a function of the message size with different weights based on their corresponding traffic priorities or traffic types. For simplicity, we assume that message sizes of different priorities (or types) are homogeneous in volume and thus indistinguishable when filling the buffer if they are accepted. We also assume that arrivals of requests for custody occur at discrete points in time, which are called decision epochs. Notice that the arrival requests (events) drive the decision epochs. In other words, the decision epochs are not predetermined but rather given by the arrival requests themselves.

It is worth noting that the custodian cost of a message in a node depends on the occupied storage space. The custodian cost increases as the utilization of storage

space increases. The state transition of the available storage space in a node has been discussed in Section 3.4.2, we here follow these rules.

In this chapter, the dynamic interactions between any two nodes during opportunistic communication will be modeled as repeated single-decision games. The nodes try to maximize their payoffs with the resource constraints.

4.5 The Custody Transfer Game

In this section, we will use repeated single-decision games to model the making decision of custody transfer in delay tolerant networks. We use the words "stage game" and "single-decision game" interchangeably later.

We suppose that any two nodes set up a game during their opportunistic communication. For any two nodes that meet for games, one node may make message transfer request, the other node can make a decision to accept the message transfer requst or reject the message transfer request. For clarity, we take an instance of game shown in Figure 4.1 as an example, in this situation, node j can make a message transfer request, node i will make a decision to accept the message transfer request or reject the message transfer request. However, this situation is not fixed. Node i can set up another game with an other node in the future, request message transfer. Node j can also set up a game with an other node and make a decison to accept the message transfer request or reject the transfer request.

In order to more clearly explain the custody transfer game, we modify Figure 4.1 into Figure 4.2 and use it to describe the custody transfer game. In the game shown in Figure 4.2, we assume that node N_i^t can make a transfer request, node i can make a decision to accept transfer request or reject transfer request. Index t in N_i^t stands for the decision epoch (or game stage index), that is, at decision epoch (or

game stage) t, node i sets up a game with node $N_i^t \in \{1, \cdots, n\} \setminus i$. We next make no difference between decision epoch and game stage index.

Table 4.2 shows payoff matrix of a specific single-decision game seceario between two nodes. In this game scenario, since node i's payoff is greater than 0 if it makes a decision to accept the transfer request, the decision to accept the transfer request will dominate the the decision to reject the transfer request.

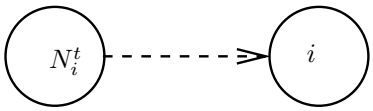


Figure 4.2: Game scenario of two players

Table 4.2: Payoff matrix of a game scenario between node i and node N_i^t.

		Node i	
Node N_i^t		Accept request	Reject request
	Request transfer	$(1, 7)$	$(0, 0)$
	Don't request	$(0, 0)$	$(0, 0)$

We next take the scenario shown in Figure 4.2 as an example to describe custody transfer game between any two nodes in a delay tolerant network.

- The players are nodes in the networks. Let $i \in \{1, \cdots, n\}$ denote a node. $N_i^t \in \{1, \cdots, n\} \setminus i$ denotes a node to communicate with node i at game stage t. For simplicity, we assume that N_i^t is a singleton. For node i, the N_i^t can be different in different stage game (stage game index t needs to be updated).
- Node i can make a decision to accept the transfer request or reject the transfer request, its action space is $S_i = \{\textit{accept request}, \textit{reject request}\}$. Node N_i^t can make a transfer request, its action space is $S_{N_i^t} = \{\textit{request transfer}, \textit{don't request}\}$.

- Table 4.3 specifies a general form of payoff matrix for a stage game. The payoffs should meet the following requirements and associated inequalities. The reason to set these requirements is to make incentives for the custoday transfer betwen any communication pair.
  1. The situation that node 1 makes a transfer request and the request is accepted by node 2 should lead to higher payoff than the situation that node 1 makes no request and node 2 also ignores the request ($R_1 + R_2 > P_1 + P_2$).
  2. The situation that node 1 makes a transfer request and the request is accepted by node 2 should lead to higher payoff than the situation that node 1 makes no request or node 2 rejects the transfer request ($R_1 + R_2 > S_1 + T_2$ and $R_1 + R_2 > T_1 + S_2$).
  3. *Reject request* (*Don't request*) dominates *accept request* (*request transfer*) at the individual level for at least one of nodes ($T_2 + P_2 > R_2 + S_2$ or $T_1 + P_1 > R_1 + S_1$).

Table 4.3: General form of a payoff matrix in a stage game for custody transfer

		Node 2	
Node 1		Accept request	Reject request
	Request transfer	(R_1, R_2)	(S_1, T_2)
	Don't request	(T_1, S_2)	(P_1, P_2)

In Table 4.3, $R_i, S_i, T_i,$ and P_i $(i \in \{1, 2\})$ stand for payoffs of node i in their corresponding strategies, respectively.

- In order to fit the requirements specified in the above item, we define a specific payoff matrix for each stage game, which is specified in Table 4.4. We will use this specific payoff matrix to specify single-decision game (stage game).

Table 4.4: A specific payoff matrix of a stage game between node i and node N_i^t.

Node N_i^t \ Node i	Accept request	Reject request
Request transfer	$(1, U_i^t(o_i^t, \beta_i^t))$	$(0, 0)$
Don' request	$(0, 0)$	$(0, 0)$

- In Table 4.4, the payoff $U_i^t(o_i^t, \beta_i^t)$ is defined as

$$U_i^t(o_i^t, \beta_i^t) = -\alpha_i(o_i^t) + \beta_i^t \tag{4.1}$$

  where $\alpha_i(o_i^t)$ is custodian cost when the occupied storage space of node i at game stage t is o_i^t, $\alpha_i(\cdot)$ is some nondecreasing function, we further assume that $\alpha_i(\cdot)$ is concave, differentiable with respect to o_i^t; $\beta_i^t \in \{B_1, \cdots, B_m\}$ is reward to accept a message transfer request at game stage t, it is dependent on the type of the request.

- Node i makes a decision to accept the message transfer request or reject the message transfer request based on control strategy defined in Section 4.6. Players (nodes) observe each other's actions, but not their strategies.

It is worth noting that in our game framework, node N_i^t's payoff is 1 if it can successfully transfer a message to node i. If node N_i^t releases its occupied space, its future custodian cost will decrease. Hence there will be possibility that it can accept messages with low rewards. It is also beneficial to drive the system to high utility.

It is also worth noting that in our framework, we have no restriction on the number of communication rounds between node i and node N_i^t (but the game stage index t will be updated). This game model supposes that each round of communication sets up a custody transfer game.

In our model, each node tries to maximize its payoff with its the storage constraint. We assume that the storage capacity for each node is C units.

4.6 Control Strategy

In this section, we first propose the control strategy, then discuss the properties of the proposed strategy. Our goal is to design a control strategy to maximize each node's payoff while respecting its resource constraint.

4.6.1 Control Strategy

In this subsection, we propose the rule for node i to choose action in each stage game. Even though the rule is the same in every stage. Note that this does not necesarily mean that the action chosen in each stage will be the same.

For a node i, if node N_i^t requests a message transfer during their communication opportunity, it can make a decision using control function σ_i defined below. In practice, node i makes its decision based on the potential request reward and its occupied custodian space, that is, the control strategy σ_i is a function of payoff $U_i^t(o_i^t, \beta_i^t)$:

$$\sigma_i[t] = \sigma_i\left[U_i^t(o_i^t, \beta_i^t)\right] = \begin{cases} 1 & \text{if } U_i^t(o_i^t, \beta_i^t) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

where $\sigma_i[t] = 1$ stands for accepting a message transfer request at decision epoch t, $\sigma_i[t] = 0$ means that a message transfer request is rejected at decision epoch t.

Note that $\sigma_i[t]$ takes as input only local information of node i that is available at decision epoch t. The rationale is that only information at decision epoch t can affect $U_i^t(\cdot)$, hence the decision of node i. Since the output of the function $\sigma_i[t]$ depends on the input $U_i^t(o_i^t, \beta_i^t)$, the strategy is reactive.

Our model only requires that node i be able to observe its occupied storage space and transfer request information at each decision epoch. Therefore the proposed strategy is completely distributed approach. This control strategy is significantly suitable for delay tolerant networks, where only local information of each node is available.

4.6.2 Strategy Analysis

In this subsection, we analyze the proposed control strategy in repeated single-decision games.

From Section 3.4.2, we can know that the state transition of available storage space in node i can be described as follows.

$$a_i^t = a_i^{t-1} - \sigma_i[t-1]x_i^{t-1} + b_i^t \tag{4.3}$$

where $\sigma_i[t-1]$ is control strategy of node i at decision epoch t; $b_i^t = 1$ if a message is successfully transferred to next custodian node, $b_i^t = 0$ otherwise; $x_i^{t-1} = 1$. The detailed explanation of equation (4.3) is referred to Section 3.4.2.

The relation between the occupied storage space and the available storage space in node i is as follow.

$$o_i^t = C - a_i^t \tag{4.4}$$

From the definition of payoff $U_i^t(o_i^t, \beta_i^t)$, the current state can have effect on payoff, the action taken by node i in a particular state determins its payoff. The control strategy for node (player) i is a mapping σ_t^i, associating the strategy with the occupied storage space o_t^i or the available storage space a_i^t.

Based on the single-decision game theory and the above analysis, we yield the following conclusions:

1. If neighbor N_i^t of node i makes a message transfer request at decision epoch t, and node i's payoff $U_i^t(o_i^t, \beta_i^t) \leq 0$, then the *reject request* will dominate the *accept request*. The strategy (*request transfer*, *reject request*) is Nash equilibrium.
2. If neighbor N_i^t of node i makes a message transfer request at decision epoch t, and node i's payoff $U_i^t(o_i^t, \beta_i^t) > 0$, then the *accept request* will dominate the *reject request*. The strategy (*request transfer*, *accept request*) is Nash equilibrium.

The goal of each node is to maximize its payoff that accumulates over time. For node i, the number of single-decision game rounds (stage game) is unpredictable (infinite or finite). We assume that the future payoffs are discounted by a factor δ for each stage game. The cumulative payoff $\bar{U}_i$ of node i is computed as the weighted sum of the payoff U_i^t that node i obtains in each stage game:

$$\bar{U}_i = \sum_{t=0}^{\infty} \delta^t U_i^t, \tag{4.5}$$

where $0 < \delta < 1$. The discounting factor δ represents the degree to which the payoff of a stage game is discounted relative to the previous stage game.

For node i, its request can be one of bids with benefits $B_1, B_2, \ldots, B_m$ at each decision epoch. The question is that node i can not know the future request given that decisions to accept "low" bids can not be reversed after knowing about future higher bids. On the other hand, it is dangerous to gamble that the future bids will be higher since they may never arrive.

Our control strategy is to make a decision to accept transfer request or reject transfer request at each decison epoch based on the occupied storage space in node i and request reward. Therefore node i makes its "best" decision based on the local information at decision epoch t without gambling its future benefits.

Based on the above analysis, we can know that node i balances its benefit at the current decision epoch and its further potential benefits.

4.7 Simulation

We developed a discrete event-driven simulator based on the one in [72] to evaluate our congestion management strategy. The simulator implements a congestion management strategy as proposed in the previous section. To isolate the effect of link bandwidth on the congestion management strategy, we still assume that each link has infinite bandwidth as we did in Section 3.7.

4.7.1 Simulation Settings

In our simulation, the simulator generates delay tolerant networks consisting of both mobile nodes and static nodes in a $3,000 \times 3,000m^2$ field. Static nodes are randomly distributed in the field and generate messages following poisson processes. Each static node can generate five classes of request messages with probability of 0.2 for each kind of messages. Mobile nodes function as relay nodes. All nodes have a uniform transmission range of $100m$. The destination node has unlimited storage capacity and is randomly located in the field and always ready to accept messages during opportunistic contact with mobile nodes. The storage capacity in each mobile node has a size of 50 messages.

The simulation parameters are shown in Table 4.5. We consider four scenarios with different mobile/static node mix: scenario 1 with 40/20 mobile/static node mix, scenario 2 with 50/20 mobile/static node mix, scenario 3 with 50/30 mobile/static node mix and scenario 4 with 60/30 mobile/static node mix, respectively.

The mobility of mobile nodes is generated as follows: (1) a square region of a given size is placed at a random position in the network, there are 20 square regions in

Table 4.5: Simulation parameters

Parameter	Value
Simulation field size	$3,000 \times 3,000$ (m^2)
Transmission range	100 m
Number of static nodes	20/20/30/30
Number of mobile nodes	40/50/50/60
Region size	400/400/400/400(m)

the network field; (2) 2 or 3 nodes are randomly placed in each region; (3) the nodes placed in each region will move in random-way-point with random initial locations in their regions. The random way point model employed for mobile nodes has a moving speed uniformly distributed in [0.2, 0.5] m/s and the pause time of a stop is uniformly distributed in [2, 3] seconds.

Since messages have to traverse lower layers of the network, they are ultimately subject to the restrictions there in term of maximum packet size. For example, on most IP networks it is safest to assume that single packet should be less than 1500 bytes long. Therefore, we assume that each message has a size of 1500 bytes [69].

It is worth noting that the routing algorithm can significantly affect the performance of congestion control in delay tolerant networks. In order to focus on congestion management, we here follow [49] and separate the congestion strategy from the routing algorithm.

We assume that there is an oracle for message routing. The oracle knows everything and can distribute routing information around the network. Notice that the oracle is only responsible for message routing. Congestion control in each node is addressed by the congestion management strategy.

4.7.2 Custodian Cost Function

The custodian cost is zero if the utilization of a node is zero. This means that the custodian cost is zero as long as the storage buffer is empty. As the messages accumulate in the storage buffer, the custodian cost increases. The cost can theoretically go as high as infinite when the storage space is fully occupied. In practice, the custodian cost of a node will increase until a given value $\mathcal{C}_{\max}$ when no message transfer request can be enforced.

The utilization of storage space in node i is given by o_i^t/C, where o_i^t is the actual number of messages occupying the storage space of node i at decision epoch t and C is the storage capacity of each mobile node. In the simulation, we scale the custodian cost of storage space in each mobile node from 0 to $\mathcal{C}_{\max}$ and choose the following custodian cost function.

$$\alpha_i(o_i^t) = \mathcal{C}_{\max} \times \frac{e^{o_i^t/C} - 1}{e - 1} \tag{4.6}$$

In the function (4.6), the custodian cost stays near 0 until the buffer utilization is almost 1, then the custodian cost goes up very quickly. Obviously, it is a concave and nondecreasing function with respect to variable o_i^t. Given the exponential nature of the custodian cost function, it is possible to approximate the custodian cost of a node, the high utilization is to yield a dramatically high custodian cost value. In practice, this custodian cost function sets variable threshold values depending on the utilization of the storage space of a node, then filters the requests with low benefits when the utilization is high.

It is worth noting that the custodian cost function (therefore payoff function) is not unique. Its choice may be determined by the application domain. We here choose an exponential function to compute the custodian cost of storage space in a node.

4.7.3 Simulation Results and Discussion

In this subsection, we will show the simulation results and give a brief analysis for the simulation results.

In the simulation, we choose $C_{\max} = 80$. The rewards for the five kinds of requests are 15, 25, 35, 45, 75 respectively. By equation (4.6), if the utilization of the storage space is near 1, no request can be enforced. In practice, there are no strict rules on the choice of request rewards and $C_{\max}$, the only requirement is that the custodian cost should be comparable with request rewards.

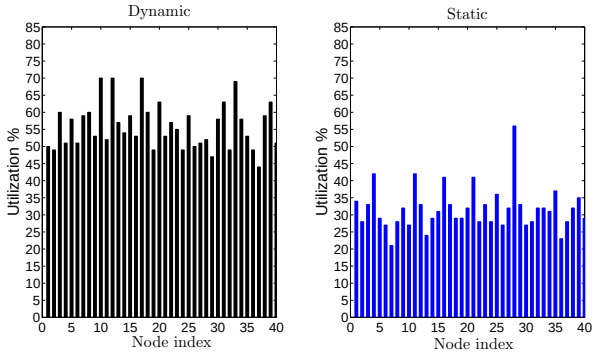
Figure 4.3 shows the snapshots of the buffer utilization for 40/20 mobile/static node mix at time $t = 400s$ and $t = 800s$ when the message arrival density generated by each static node is 1/2 message/second. Figure 4.4 shows the snapshots of the buffer utilization for 40/20 mobile/static node mix at time $t = 400s$ and $t = 800s$ when the message arrival density generated by each static node is 2/3 message/second.

Figure 4.5 shows the snapshots of the buffer utilization for 50/20 mobile/static node mix at time $t = 400s$ and $t = 800s$ when the message arrival density generated by each static node is 1/2 message/second. Figure 4.6 shows the snapshots of the buffer utilization for 50/20 mobile/static node mix at time $t = 400s$ and $t = 800s$ when the message arrival density generated by each static node is 2/3 message/second.

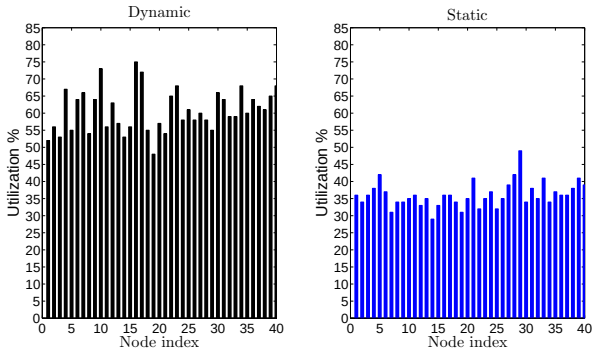
Figure 4.7 shows the snapshots of the buffer utilization for 50/30 mobile/static node mix at time $t = 400s$ and $t = 800s$ when the messages arrival density generated by each static node is 2/3 messages/second. Figure 4.8 shows the snapshots of the buffer utilization for 60/30 mobile/static node mix at time $t = 400s$ and $t = 800s$ when the message arrival density generated by each static node is 2/3 message/second.

Figure 4.9 shows the throughput at $t = 400$ and $t = 800$ for several simulation scenarios.

By analyzing the simulation results, we can obtain the following conclusions:



(a)



(b)

Figure 4.3: Load distribution in nodes - 40/20 node mix, message arrival density = 1/2 message/second. (a) Load distribution at $t = 400s$. (b) Load distribution at $t = 800s$.

- The congestion control strategy with dynamic custodian cost can achieve much evenly balanced loads and higher utilization in all nodes of the network and better throughput.
- Under the same simulation conditions except the message arrival density, the increase in message arrival density can not significantly improve the throughput.
- Under the same simulation conditions except the mobile nodes, the more mobile nodes move in the network field, the more throughput can be obtained. The reason is that if the number of mobile nodes increases in the network field, there will be more communication opportunity among network nodes, i.e., the communication opportunity between static nodes and mobile nodes, the com-

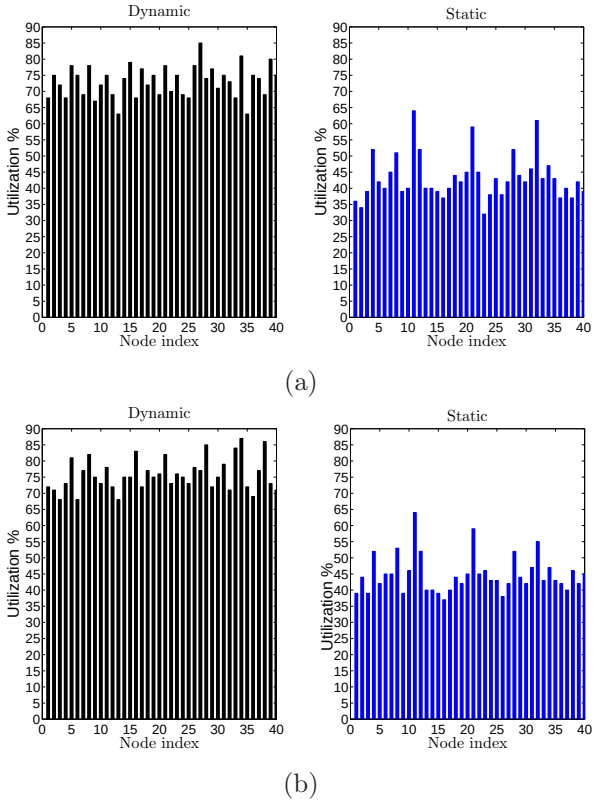
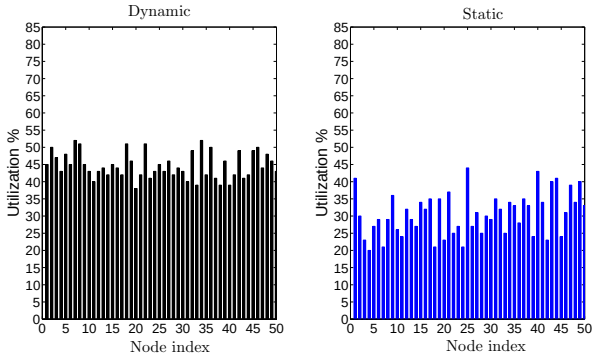


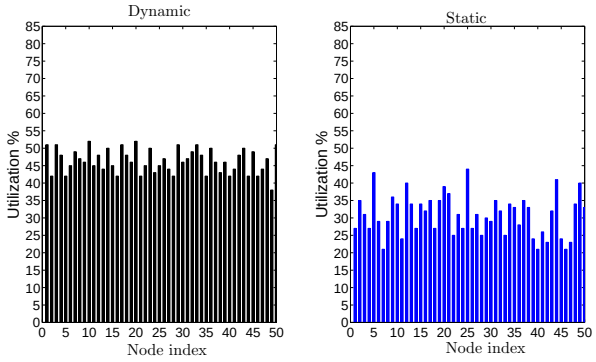
Figure 4.4: Load distribution in nodes - 40/20 mobile/static node mix, message arrival density = 2/3 message/second. (a) Load distribution at $t = 400s$. (b) Load distribution at $t = 800s$.

munication opportunity among mobile nodes, the communication opportunity between mobile nodes and destination nodes.

It is worth noting that it is difficult to set a "optimal" static custodian cost as a threshold. If this threshold is too low, the threshold can not filter any requests with low rewards, then there will be no difference between the requests with low rewards and the requests with high rewards. If the threshold is too high, most of the requests will be rejected even if the utilization of the storage buffer is very low. Therefore, it is dangerous to set a static custodian cost to obtain optimal solution. In our simulation, we choose the average value of the five request rewards. However, the average value does not mean that it is the optimal threshold value. Since the dynamic policy can



(a)



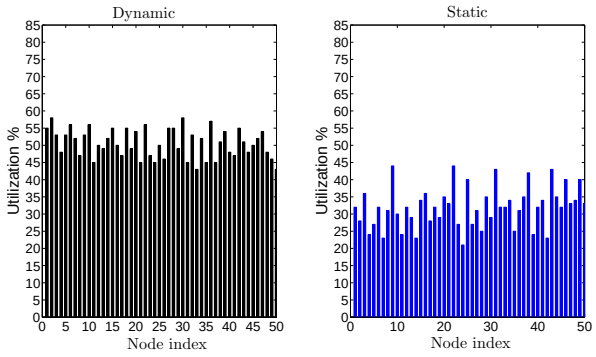
(b)

Figure 4.5: Load distribution in nodes - 50/20 mobile/static node mix, message arrival density = 1/2 message/second. (a) Load distribution at $t = 400s$. (b) Load distribution at $t = 800s$.

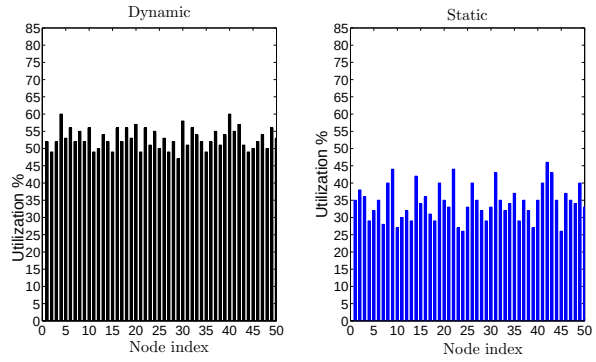
adapt the custodian cost based on the varying occupied storage space, the storage space usage can be optimized and the throughput can be improved.

4.8 Conclusion

In this chapter, we employ repeated one-decision games to model dynamic behaviors of delay tolerant networks. The repeated one-decision game based approach is significantly suitable for modeling the intermittently communicating networks such as delay tolerant networks, where every node can not be sure that it is going to communicate with different opponents next round and there is no knowledge when the dynamic behavior is going to stop. Simulation results show that the proposed control



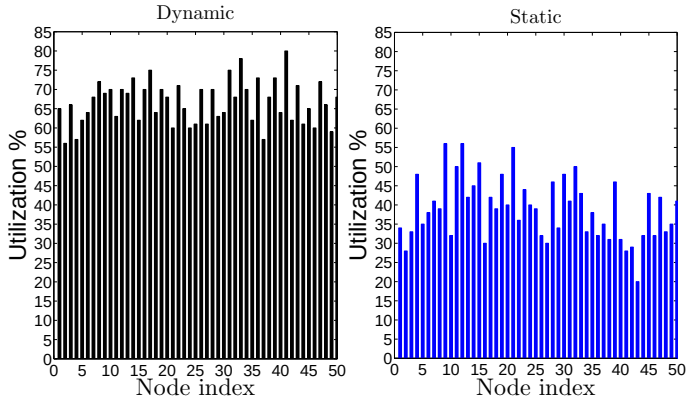
(a)



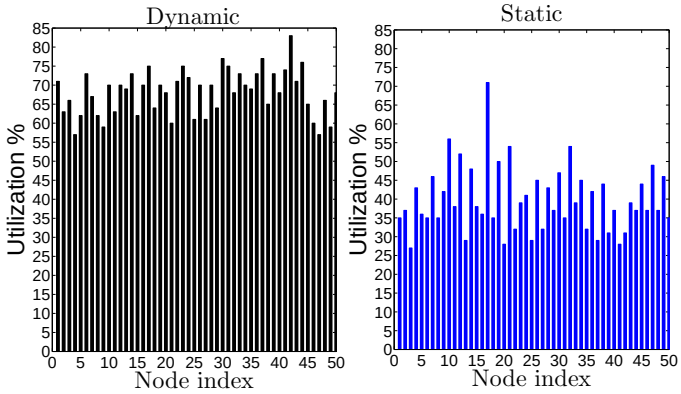
(b)

Figure 4.6: Load distribution in nodes - 50/20 mobile/static node mix, message arrival density = 2/3 message/second. (a) Load distribution at $t = 400s$. (b) Load distribution at $t = 800s$.

strategy is effective in avoiding congestion and balancing network load among the nodes.



(a)



(b)

Figure 4.7: Load distribution in nodes - 50/30 mobile/static node mix, message arrival density = 2/3 message/second. (a) Load distribution at $t = 400s$. (b) Load distribution at $t = 800s$.

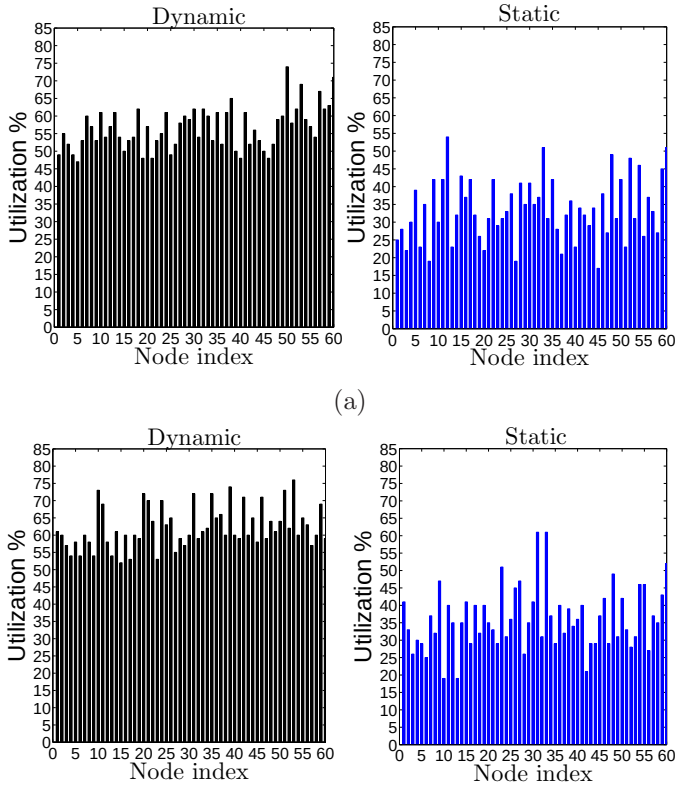


Figure 4.8: Load distribution in nodes - 60/30 mobile/static node mix, message arrival density = 2/3 message/second. (a) Load distribution at $t = 400s$. (b) Load distribution at $t = 800s$.

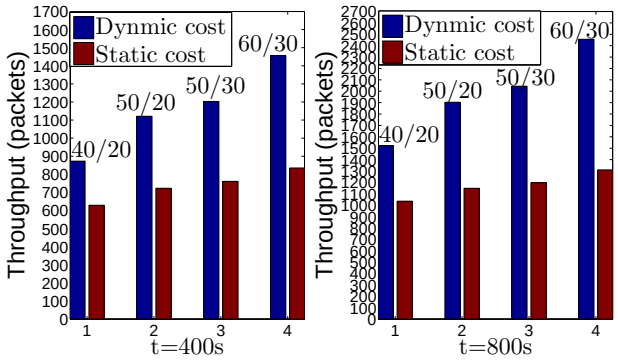


Figure 4.9: The throughput for several simulation scenarios at $t = 400s$ and $t = 800s$ (message arrival density = 2/3 message/second).

CHAPTER 5

CONCLUSION

In this dissertation, we explore congestions control for networks in challenged environments.

In Chapter 1, we introduce congestion control in the Internet, and describe two kinds of challenged networks, i.e., networks with time varying link capacities and networks that intermittently communicate. In this chapter, we also motivate the research in the dissertation.

In Chapter 2, we explicitly model link capacities to be time varying and study the congestion control strategies. In particular, we propose a convex optimization based congestion control algorithm which is proved to be trajectory stable in the absence of feedback delay. Different from system stability around a single equilibrium point, trajectory stability guarantees the system is stable around a time varying reference trajectory. Moreover, we obtain sufficient conditions for the congestion control algorithm to be locally stable in the presence of delay. We model time variation of capacity as perturbation to a constant to evaluate the impact of link capacity variation on the congestion control algorithm and through simulations study the tradeoff between stability and robustness of the congestion control algorithm against link capacity variation.

Chapter 3 explores the congestion control strategies in intermittently communicating networks, where continuous end-to-end connectivity may not exist, the round trip delay can be excessively high and TCP breaks. For this kind of challenged networks, the optimization framework developed for the Internet based congestion

control algorithm is not applicable. Therefore, we apply the concepts of revenue management such as benefit function and opportunity cost, and employ dynamic programming to study congestion control strategies in intermittently communicating networks. The developed congestion control strategies are distributed in nature where only the local storage information of a node and the reward of the request are required. The control scheme is especially suitable for intermittently communicating networks, where global information is not available, and networks are inherently dynamic. Simulation results show that the proposed congestion control strategies are effective to avoid potential congestion and balance network load among the nodes.

In Chapter 4, we employ repeated one-decision games to model the dynamic behaviors of delay tolerant networks. The repeated one-decision game based approach is significantly suitable for modeling the intermittently communicating networks such as delay tolerant networks, where every node can not be sure that it is going to communicate with different opponents next round and there is no knowledge when the dynamic behavior is going to stop. Our simulation results show that the control strategy based on repeated one-decision games is effective in avoiding congestion and balancing network load among the nodes.

REFERENCES

[1] M. Welzl, *Network Congestion Control: Managing Internet Traffic*. John Wiley & Sons, Ltd, 2005.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks," *Computer Networks*, 47(2005) pages 445-487.

[3] Ö. B. Akan, I. F. Akyildiz, "ATL: An Apaptive Transport Layer Suite for Next-Generation Wireless Internet," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 5, June 2004.

[4] S. Cen, P. C. Cosman, G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, October 2003.

[5] F. P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proporttional Fairness and Stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237-252.

[6] F. P. Kelly, "Mathematical modelling of the Internet," In *Mathematics Unlimited - 2001 and Beyond* (Editors B. Engquist and W. Schmid). Springer-Verlag, Berlin, 2001

[7] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8 (1997) pages 33-37.

[8] S. H. Low, F. Paganini, and J. C. Doyle, "Internet Congestion Control," *IEEE Control Systems Magazine*, vol. 22, no. 1, February 2002.

[9] Y. Xue, B. Li, and K. Nahrstedt, "Optimal Resource Allocation in Wireless Ad Hoc Networks: A Price-Based Approach," *IEEE Trans. Mobile Computing*, vol. 5, no. 4, April 2006.

[10] S. H. Low and R. Srikant, "A Mathematical Framework for Designing a Low-Loss, Low-Delay Internet," *Networks and Spatial Economics*, 4:(2004) 75-101.

[11] S. H. Low, L. L. Peterson, and L. Wang, "Understanding TCP Vegas: A Duality Model," *Journal of the ACM*, vol. 49, no. 2, March 2002.

[12] S. H. Low and D. E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, December 1999.

[13] B. Johansson, P. Soldati, and M. Johansson," Mathematical Decomposition Techniques for Distributed Cross-Layer Optimization of Data Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, August 2006.

[14] H. Elaarag, "Improving TCP Performance over Mobile Networks," *ACM Computing Surveys*, vol. 34, no. 3, September 2002.

[15] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[16] A. Goldsmith, *Wireless Communication*. Cambridge University Press, 2005.

[17] J. Kurose, K. Rose, *Computer Networking: A top down approach featuring the Internet*. Addison-Wealey, 2005.

[18] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhäuser Boston, 2004.

[19] S. S. Kunniyur and R. Srikant, "End-to-end Congestion Control: Utility Functions, Random Losses and ECN Marks," *IEEE Trans. Automat. Contr.*, vol. 48, no. 10, October 2003.

[20] S. S. Kunniyur and R. Srikant, "Stable, Scalable, Fair Congestion Control and AQM Schemes that Aehieves High Utilization in the Internet," *IEEE Trans. Automat. Contr.*, vol. 48, no. 11, November 2003.

[21] S. S. Kunniyur and R. Srikant, "An Adaptive Virtual Queue(AVQ) Algorithm for Active Queue Management," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, April 2004.

[22] A. Lakshmikantha, C. L. Beck, and R. Srikant, "Robustness of Real and Virtual Queue-Based Active Queue Management Schemes," *IEEE/ACM Trans. Networking*, vol. 13, no. 1, February 2005.

[23] C. Jin, X. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," In Proceedings of *IEEE INFOCOM 2004*, March 2004, Hongkong, China.

[24] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP flows," *IEEE Trans. Automat. Contr.*, vol. 47, no. 6, June 2002.

[25] M. Chiang, "Balancing Transport and Physical Layers in Wireless Multihop Networks: Jointly Optimal Congestion Control and Power Control," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, January 2005.

[26] M. Chiang, S. H. Low, A. Calderbank, and J. Doyle, "Layering as Optimization Decomposition: AMathematical Theory of Network Architectures," *Proceedings of the IEEE*, vol. 95, no. 1, January 2007.

[27] B. Radunović and J. L. Boudec, "Rate Performance Objective of Multihop Wireless Networks," *IEEE Trans. Mobile Computing*, vol. 3, no. 4, October-December 2004.

[28] Y. Yi and S. Shakkottai, "Hop-by-Hop Congestion Control over a Wireless Multihop Network," *IEEE/ACM Trans. Networking*, vol. 15, no. 1, February 2007.

[29] X. Lin and N. B. Shroff, "The Impact of Imperfect Scheduling on Cross-Layer Rate Control in Wireless Networks," In Proceedings of *IEEE INFOCOM 2005*, March 2005, Miami, Florida.

[30] X. Lin, N. Shroff, and R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, August 2006.

[31] A. Gurtov and S. Floyd, "Modeling Wireless Links for Transport Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, April 2004.

[32] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, August 1993.

[33] M. J. Neely, "Energy Optimal Control for Time Varying Wireless Networks," In Proceedings of *IEEE INFOCOM 2005*, March 2005, Miami, Florida.

[34] L. Georgiadis, M. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, vol. 1, no. 1, 2006.

[35] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for wireless ad hoc networks," In Proceedings of *IEEE INFOCOM 2005*, March 2005, Miami, Florida.

[36] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", In Proceedings of *ACM SIGCOMM*, August 2002, Pittsburgh, PA.

[37] S. Liu, T. Başar, and R. Srikant, "Pittfalls in the fluid modeling of RTT variations in windows-based congestion control," In Proceedings of *IEEE INFOCOM 2005*, March 2005, Miami, Florida.

[38] S. Liu, T. Başar, and R. Srikant,“ Exponential-RED: A Stabilizing AQM Scheme for Low- and High-Speed TCP Protocols,” *IEEE/ACM Trans. Networking*, vol. 13, no. 5, October 2005.

[39] S. Sastry, M. Bosdon, *Adaptive Control: Stability, Convergence, and Robustness*. Prentice Hall, 1989.

[40] R. R. Mohler, *Nonlinear Systems: Vol I: Dynamics and Control*. Prentics-Hall, Inc., 1991.

[41] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[42] H. K. Khalil, *Nonlinear Systems* 3rd ed. Prentice Hall, 2002.

[43] C. A. Desoer and Y. Wang, “On the Generalized Nyquist Stability Criterion,” *IEEE Trans. Automat. Contr.*, vol. 25, no. 2, 1980.

[44] D. Serre, *Matrices: Theory and Applications*. Springer-Verlag, New York, Inc 2002.

[45] A. Deif, *Sensitivity Analysis in Linear Systems*. Springer-Verlag, 1986.

[46] Y. Zhang and M. Ahmed, “A Control Theoretic Analysis of XCP,” In Proceedings of *8th IEEE Global Internet Sympoisum*, March 2005, Miami, Florida,

[47] K. Fall, “A Delay-Tolerant Network Architecture for Challenged Internets,” In Proceedings of *ACM SIGCOMM’03*, August 2003, Karlsruhe, Germany.

[48] K. Fall, W. Hong, and S. Madden, “Custody Transfer for Reliable Delivery in Delay Tolerant Networks,” Available [online]: `http://www.dtnrg.org/papers/custody-xfer-tr.pdf`.

[49] M. Seligman, K. Fall, and P. Mundur, “Alternative Custodians for Congestion Control in Delay Tolerant Networks,” In Proceesings of *SIGCOMM’06 Workshop*, September 2006, Pisa, Italy.

[50] S. Burleigh, E. Jennings, and J. Schoolcraft, "Autonomous Congestion Control in Delay-Tolerant Networks," Available [online]: `http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/39835/1/06-0856.pdf`.

[51] K. Talluri, G. Van Ryzin, *The Theory and Practice of Revenue Management*. Kluwer Academic Publishers 2004.

[52] Dimitri P. Bertsekas, *Nonlinear Programming* (2nd Edition). Athena Scientific, 1999.

[53] S. Boyd, L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004

[54] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[55] K. Harras, K. Almeroth, "Transport Layer Issues in Delay Tolerant Mobile Networks," In Proceedings of *IFIP Networking Conference*, May 2006, Coimbra, Portugal.

[56] D. Bertsekas, A. Nedić, and A. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[57] A. Nedić, A. Ozdaglar, "On the rate of convergence of distributed subgradient methods for multi-agent optimization ," In Proceedings of *the 46th IEEE CDC*, December 2007, New Orleans, LA.

[58] N. Chang, M. Liu, "Revisiting the TTLbased Controlled Flooding Search: Optimality and Randomization," In Proceedings of *ACM MobiCom'04*, September 2004, Philadelphia, Pennsylvania.

[59] Yair Amir *et al*, "A Cost-Benefit Flow Control for Reliable Multicast and Unicast in Overlay Networks," *IEEE/ACM Trans. Networking*, vol. 13, no. 5, October 2005

[60] M. Park, V. Rodoplu, "Traffic Allocation in Delay-Tolerant Wireless Ad Hoc Networks," In Proceedings of *IEEE WCNC 2006*, April 2006, Las Vegas, Nevda.

[61] Y. Wang, H. Wu, "DFT-MSN: The Delay/Fault-Tolerant Mobile Sensor Network for Pervasive Information Gathering," In Proceedings of *IEEE INFOCOM 2006*, April 2006, Barcelona, Spain.

[62] M. Grossglauser, D. Tse, "Mobility Increases the Capacity of Ad Hoc Wireless Networks," *IEEE/ACM Trans. Networking*, vol. 10, no. 4, August 2002.

[63] J. Zhao, G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Neworks," *IEEE Trans. Vehicular Technology*, vol. 57, no. 3, May 2008.

[64] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," In Proceedings of *ACM Mobihoc'04*, May 2004, Tokyo, Japan.

[65] W. Zhao, M. Ammar, "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," In Proceedings of *IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003, San Juan, Puerto Rico.

[66] M. Mukarram, B. Tariq, M. Ammar, and E. Zegura, "Message Ferry Route Design for Space Ad Hoc Networks with Mobile Nodes," In Proceedings of *ACM Mobihoc'06*, May 2006, Florence, Itlay.

[67] A. Vahdat, D. Becker, "Epidemic routing for partially-connected ad hoc networks," Available [online]: `http://citeseer.ist.psu.edu/306010.html`

[68] S. Burleigh, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-Tolerant Networking: An approach to Interplanetary Internet," *IEEE Communication Magazine*, vol. 41, no. 6, June 2003.

[69] S. Farrell, V. Cahill, *Delay- and Disruption- Tolerant Networking.* Artech House, 2006.

[70] S. Farrell, V. Cahill, D. Geraghty, and I. Humphreys, "When TCP breaks: Delay- and Disruption-Tolerant Networking," *IEEE Internet Computing*, vol. 10, no. 4, July-August 2006.

[71] J. Wu, S. Yang, and F. Dai, "Logarithmic Store-Carry-Forward Routing in Mobile Ad Hoc Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 6, June 2007.

[72] C. Liu, J. Wu, "Scale Routing in Delay Tolerant Networks," In Proceedings of *ACM Mobihoc'07*, September 2007, Montréal, Canada.

[73] S. Merugu, M. Ammar, and E. Zegura, "Routing in Space and Time in Networks with Predictable Mobility," Technical Report, GIT-CC-04-07, Georgia Institute of Technology.

[74] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Single-copy Case," *IEEE/ACM Trans. Networking*, vol. 16, no. 1, February 2008.

[75] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multi-copy Case," *IEEE/ACM Trans. Networking*, vol. 16, no. 1, February 2008.

[76] K. Harras, M. Wittie, K. Almeroth, and E. Belding, "ParaNets: A Parallel Network Architecture for Challenged Networks," In Proceedings of *IEEE Workshop on Mobile Computing Systems and Applications* (*HotMobile*). Tucson, AZ, February 2007.

[77] S. Jain, "Routing in Delay Tolerant Networks," Ph. D. dissertation, University of Washington 2005.

[78] T. Voice, "Stability of Multi-Path Dual Congestion Control Algorithms," *IEEE/ACM Trans. Networking*, vol. 15, no. 6, December 2007.

[79] B. Burns, O. Brock, and B. Levine, "MORA Routing and Capacity Building in Disruption-Tolerant Networks," *Elsevier Ad hoc Networks Journal*, vol. 6, no. 5, July 2008.

[80] B. Burns, O. Brock, and B. Levine, "Autonomous Enhancement of Disruption Tolerant Networks," In Proceedings of *2006 IEEE International Conference on Robotics and Automation*, May 2006, Orlando, Florida.

[81] L. Ying, R. Srikant, "Optimal Delay-Throughput Trade-offs in Mobile Ad-hoc Networks: Hybrid Random Walk and One-dimensional Mobility Models," In Proceedings of *ITA workshop 2007*. Available [online] `http://arxiv.org/abs/0705.0326`.

[82] DTN simulator. Available [online] `http://tuxmaster.usc.edu/`.

[83] D. Fudenberg, J. Tirole, *Game Theory*. MIT Press 1991.

[84] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press 2007.

[85] M. Afergan, Applying the Repeated Game Framework to Multiparty Networked Applications. Ph.D. Dissertation, MASSACHUSETTS INSTITUTE OF TECHNOLOGY 2005.

[86] M. Afergan, "Using Repeated Games to Design Incentive-Based Routing Systems," In Proceedings of *IEEE INFOCOM 2006*, April 2006, Barcelona, Spain.

[87] M. Afergan, R. Sami, "Repeated-Game Modeling of Multicast Overlays," In Proceedings of *IEEE INFOCOM 2006*, April 2006, Barcelona, Spain.

[88] B. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz, "Characterizing Selfishly Constructed Overlay Routing Networks," In Proceedings of *IEEE INFOCOM 2004*, March 2004, Hong Kong, China.

[89] A. Urpi, M. Bonuccelli, S. Giodano, "Modeling cooperation in mobile ad hoc netowrk: a formal description of selfishness," In Proceedings of *IEEE/ACM WIOPT 2003*, March 2003, France.

[90] L. Jian, J. MacKie-Mason, "Why Share in Peer-to-Peer Networks?" Available [online] `http://hdl.handle.net/2027.42/49504`.

[91] K. Lai, M. Feldman, I. Stoica, J. Chuang, "Incentives for Cooperation in Peer-to-Peer Networks," Available [online] `http://citeseer.ist.psu.edu/579402.html`.

[92] P. Nurmi, "Modeling Energy Constrained Routing in Selfish Ad Hoc Networks," In Proceedings of *GameNets'06*, October 2006, Pisa, Italy.

[93] P. Nurmi, "Modelling Routing in Wireless Ad Hoc Networks with Dynamic Bayesian Games," In Proceedings of *IEEE SECOM 2004*, October 2006, Santa Clara, CA.

[94] G. Mailath, L. Samuelson, *Repeated Games and Reputations: Long-run Relationships.* Oxford University 2006.

[95] J. Webb. *Game Theory - Decisions, interaction and Evolution.* Springer-Verlag London Limited 2007.

[96] R. La, V. Anantharam, "Optimal Routing Control: Repeated Game Approach," *IEEE Trans. Automat. Contr.*, vol. 47, no. 3, March 2002.

[97] Z. Han, C. Pandana, and K. Liu, "A Self-Learning Repeated Game Framework for Optimizing Packet Forwarding Networks," In Proceedings of *IEEE WCNC 2005*, March 2005, New Orleans, LA, USA.

BIOGRAPHICAL STATEMENT

Guohua Zhang was born in Suzhou, Jiangsu Province, P.R. China. He received his B.S. degree from Dalian Fisheries University, P.R. China, in 1987, his M.S. in Robotics from Harbin Institute of Technology. He also received M.S. degree in Computer Science from The University of Texas at Dallas in 2002. From 2002 to 2008, he was Ph.D. student at Department of Computer Science and Engineering Engineering, The University of Texas at Arlington. His research interest focuses on optimization and congestion control of communication systems, convex optimization, discrete event systems and large scale optimization.