CAPACITY EVALUATION FOR AD HOC NETWORKS WITH END-TO-END DELAY CONSTRAINTS

ZHANG JUNXIA

(B.Eng., Tianjin University)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgement

Although this thesis presents my individual work, there are many people who contributed to it by their discussion and support. Firstly I thank Dr. Winston Khoon Guan Seah, my supervisor, whose guidance, motivation and discussion have been invaluable throughout my studentship in I2R. I also thank Er Inn Inn, Li Xia, and Tan Hwee Xian for their help and support on my research work, and useful tips for programming and experiments. Also thanks to everybody who gave me help to made me finally complete my thesis.

The I2R and ECE provide me an excellent environment in which to study and research. Thanks also to all staff who have been very helpful throughout my study.

Finally, thanks also to my family for their love and support: Mom for her enthusiasm, Dad for his advice and motivation, and Qi, my little brother, for his concern. Many thanks to Su Mu; I couldn’t finish the research without his encouragement and support. Also thanks to all my good friends, they bring me the most happiness and get me through the difficult research stage.

Table of Contents

List of Figures

List of Tables

Summary

Once rooted in research for military networks and applications, ad hoc networks have become increasingly important in commercial applications. Nodes in ad hoc networks move randomly and self-organize and self-manage without any infrastructure support or central administration. These properties make ad hoc networks suitable for use in hostile terrains where wired networks cannot be built. In some of these special situations, like battlefields, high performance wireless communication is needed. These factors, amongst others, have motivated the continuous research and development efforts to improve the performance of ad hoc networks.

Ad hoc network performance has been investigated under different transmission scenarios and network models. However, most of them have achieved satisfactory network capacity at the expense of increased transmission delay. In these scenarios, applications are delay-tolerant. Nevertheless some real-time applications, such as audio and video transmission, may require end-to-end delay to be below a certain threshold. These kinds of applications are delay-sensitive. Thus, besides delay-tolerant applications, there is a need to support delay-sensitive real-time applications in ad hoc networks too. So far, little work has been done to evaluate the capacity in this domain where there are still many aspects that need to explore.

Hence, our research objective is to design algorithms to obtain the capacity of ad hoc networks serving delay sensitive applications. Due to the requirement of real-time services, these algorithms should be feasible, scalable, run in polynomial time and use easily obtained information.

In this thesis, the network capacity is defined as the number of sessions that can be supported in the network simultaneously subject to the end-to-end delay constraints. The ad hoc networks are modeled as an undirected graph G(V,E,$\mathbf{A}$), where V denotes the node set in the network and $\mathbf{A}$ is the adjacency matrix that describes the topology of the network. Algorithms are designed based on one-hop and multi-hop adjacency matrixes to obtain the network capacity through a set of selecting and deleting operations. These algorithms can achieve results close to optimal results achieved by exhaustive brute-force search algorithm, with much less time complexity. In addition, our algorithms only require each node to have local knowledge of its adjacent neighbors, which makes our algorithm scalable.

The upper-bound of the capacity can serve as a reference or criteria for accepting new communication requests, where any of the source-destination pairs containing these sessions should meet end-to-end delay constraints. On the other hand, the lower-bound of capacity can be adopted to scale the network resources utilization.

We also estimate the maximum end-to-end delay for the flows running in the network adopting IEEE 802.11 as the MAC protocol. Although some previous works in performance evaluation for IEEE 802.11 have addressed this topic, the results are not directly applicable here. Our research solves this problem through mathematical analysis.

Besides the major contributions mentioned above, there are another two supplements in this study. Firstly, we designed an algorithm to obtain the average hop count of the paths in the networks. Secondly, we calculated the queuing delay caused by IEEE 802.11 MAC protocol, which enables us to estimate the end-to-end delay of a flow.

Chapter 1 Introduction

1.1 Background and Motivation

1.1.1 Background

1.1.1.1 Mobile Ad Hoc Networks

Emerging in the 1970s, wireless networks have become increasingly popular in the network industry. A category of wireless network architectures, viz., Mobile Ad Hoc Networks (MANETs) are expected to play important roles in civilian applications. A MANET consists of a group of autonomous wireless nodes which are all mobile, and create a wireless network dynamically among themselves without using any infrastructure or administrative support [1][2]. One ad hoc network example is shown in Figure 1.1. MANETs can be created and used "anytime, anywhere" and they are self-configuring, self-organizing and self-administering [3]. The nodes in an ad hoc network are mobile and can dynamically join and leave the network. Thus the network topology changes, since they are not limited by fixed topologies. MANETs offer unique benefits and versatility which cannot be satisfied by wired networks for certain environments and applications. These perceived advantages have elicited the widespread use of MANETs in military and rescue operations, especially under disorganized or hostile environments.

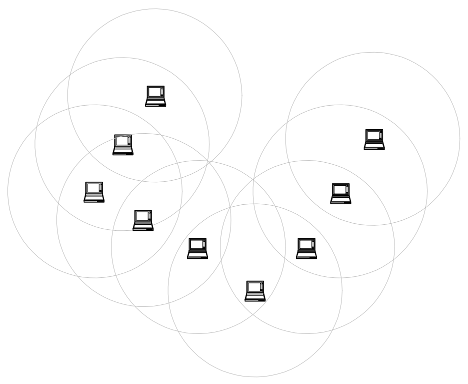


Figure 1.1： An ad hoc network example

On the other hand, mobile ad hoc networking technology faces a unique set of challenges which includes, but is not limited to, effective multihop routing, medium access control (MAC), mobility and data management, congestion control and quality of service (QoS) support. A set of six properties listed below form the basis of these challenges [4]:

- Lacking of centralized authority for network control, routing or administration (e.g. Base Station).
- Network devices can move in time domain and space domain rapidly and randomly (Mobility). Hence, the topology of a MANET may change rapidly and randomly from time to time. Route instability, caused by the mobility of nodes, is expected to result in short-lived links between nodes as the nodes move in and out of range of one another. Strict QoS, as in wired networks, cannot be guaranteed in an ad hoc network when mobility is present.
- All communications are carried over the bandwidth-constrained wireless media. Furthermore, after accounting for the effects of multiple access, fading, noise and interference conditions, and other factors, the realized throughput of wireless communications is often much less than a radio's maximum transmission rate. These

effects will also result in time-varying channel capacity, making it difficult to determine the aggregate bandwidth between two endpoints.

- Resources, including energy, bandwidth, processing capability and memory are strictly limited and must be conserved. The limited power of the mobile nodes and the lack of a fixed infrastructure in ad hoc networks restrict the transmission range, requiring multihop routing.
- Mobile nodes that are end points for user communications and applications must operate in a distributed and cooperative manner to handle network functions, most notably routing and MAC, without specialized routers.
- Each node may have different capabilities. In order to be able to connect to infrastructure-based networks (to form a hybrid network), some nodes should be able to communicate with more than one type of network.

1.1.1.2 Network Performance

In some crucial situations, like communication on the battlefield that sees an unknown terrain and requires minimum network planning or administration, the ad hoc network must support a wide category of services, such as group calls, situation awareness, fire control, and so on. In addition, users would like to transmit a variety of information, such as data, audio, and video [5]. The different services will have varied Quality of Service (QoS) demands, i.e. different demands on delay, packet loss ratio, throughput, etc.

Given the dynamics of the network topology, the underlying network protocols must be able to cope with the topology dynamics efficiently while yielding good communication performance.

To supply satisfactory ad hoc network performance, we need to consider various critical factors when evaluating MANETs, such as end-to-end delay, capacity utilization, power efficiency and throughput. Different performance metrics are defined or need to be defined under various MANET conditions and they would help to measure the network functionalities to fulfill the QoS requirements of users.

1.1.2 Motivations

Performance evaluations of MANETs have been carried out by various researchers. Most of them have chosen throughput, delay, packet loss, etc. as performance metrics. The work can be categorized base on mobility, routing protocols, MAC protocols, topology management or some other aspects. Besides these, there are other scenario/situation-related parameters relevant to performance evaluations, for example, the mean call connection time in telephone system.

However, most of the previous work have focused primarily on the performance issues of delay-tolerant applications under different network models and transmission scenarios and achieved satisfactory network capacity at the expense of increased transmission delay. If the flows in the ad hoc networks are carrying video or audio traffic, these methods are no longer suitable because these kinds of flows often have certain delay constraints. According to the ITU (International Telecommunication Union), human conversation tolerates a maximum end-to-end delay of between 150 and 300 milliseconds. Therefore, besides those delay-tolerant applications, we should put effort in delay-sensitive real-time applications supported by ad hoc networks as well.

Some work has been done to evaluate the capacity of ad hoc networks carrying delay-sensitive flows. They evaluate capacity metrics under end-to-end delay constraints.

Although these metrics can give us a picture of performance of an ad hoc network, they cannot be adopted to obtain better quality of service (QoS) easily. This motivates us to search for a capacity metric which not only evaluates the performance of the networks, but can also be used to achieve certain service quality.

Another motivation for us is that much work on evaluating capacity of ad hoc networks has been done through simulations. Simulation is a good straightforward method to evaluate the capacity of ad hoc networks with the ability to reasonably model real-life scenarios. However, it lacks expansibility because the simulation result from a specific scenario is unlikely to be easily applied to other scenarios.

Mathematical analysis can complement this inadequacy. Mathematical analysis refers to the use of mathematical tools, such as graph theory, queuing theory, etc. to derive the mathematical expressions for performance metrics, such as delay and throughput. Changes in network scenarios can be easily analyzed, simply by modifying certain parameters in the mathematical expressions, thus reducing considerable time and effort spent on simulations and their subsequent analysis of the results.

1.2 Thesis Aims

The objective of our research is the mathematical analysis of network capacity subject to certain end-to-end delay constraints. The capacity metric should be able to: (i) represent the performance of network; (ii) be evaluated using mathematical method; and (iii) be a criterion used to achieve certain quality of service.

In this research, the capacity metric is defined as the number of sessions an ad hoc network can support simultaneously with certain end-to-end delay constraints. The metric can determine whether a new communication request can be accepted or not to guarantee that all running flows meet the predefined end-to-end delay constraints. [6].

In addition, the time to obtain the capacity should not be too long because the mobility of nodes makes the network topology change rapidly. Hence, the algorithms should have low time complexity. Furthermore, the algorithms should be suitable for networks with different sizes because nodes may enter and leave an ad hoc network randomly.

1.3 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 presents the related work. In the first part, capacity evaluations under different network models and transmission scenarios are introduced, which includes the related work on the capacity evaluations with end-to-end delay constraints which is closely related to our work. In the second part, some works of the performance evaluations on IEEE802.11 MAC protocol are introduced, based on which, a few parts of our research are developed. Chapter 3 describes the network capacity definition and the mathematical model used in our research. We combine the graph theory and matrix theory to model ad hoc networks. Based on the mathematical model, in Chapter 4 we propose two algorithms to obtain the upper-bound of network capacity for two scenarios: non-channel-sharing scenario and channel-sharing scenario without considering queuing delay at each node. In Chapter 5, we estimate the queuing delay. The main components of the service time are the transmission time and the channel contention time which is determined by MAC protocol, IEEE 802.11 in our case. Queuing delay can be obtained through solving the mean and variance of the service time and inter-arrival time. Chapter 6 extends the methods and results of Chapter 5 to estimate the end-to-end delay of randomly chosen flows. Based on the end-to-end delay estimation, we propose the algorithm to obtain the lower-bound of the network capacity in Chapter 7. Finally, a summary of the work presented in the thesis is given in Chapter 8. It points out the key contributions of our work and some directions for the future work.

Chapter 2 Reviews of Related Work

2.1 Introduction

An ad hoc network is a self-organizing and rapidly deployable network, in which neither a wired backbone nor a base station is necessary and allows nodes to move about arbitrarily. This feature enables ad hoc networks to be used in some special situations where it is infeasible to build a wired network.

However, this property also restricts available resources in ad hoc networks due to the resource limitations on each node, such as bandwidth and power. Each node can only communicate directly with other nodes within its transmission range. If the destination node is out of the transmission range of the source node, the packets have to be relayed by intermediate nodes along the path selected by particular routing protocol. This is called multihop transmission.

Multiple factors affect the performance of the ad hoc networks. Routing is an important factor and the Medium Access Control (MAC) protocol is another important aspect. A MAC protocol is used to schedule the data flows on a shared channel in an ad hoc network. The effectiveness of these protocols will affect the performance of the ad hoc networks. Besides these, power control, and scalability are also the factors affecting the performance of ad hoc networks.

Taxonomy for performance evaluation of ad hoc networks is presented in Figure 2.1. In the taxonomy, we classify general ad hoc networks into two categories, one of which is the pure ad hoc network and the other is the hybrid ad hoc network with a wired backbone. In both categories, two aspects of evaluation have to been taken into consideration: (i) network capacity and (ii) protocol performance.

The network capacity needs to be evaluated for either delay-tolerant services or delay-sensitive services respectively according to the real scenarios. On the other hand, typical performance of protocols includes the evaluation of routing protocols, MAC protocols and the power control algorithms.

Many capacity metrics, such as delay, throughput, packet loss ratio etc., have been defined to measure the efficiency of a network or a protocol. Moreover, some special capacity metrics are also defined for particular systems, such as the mean connection time and the mean number of connections for telephone system.

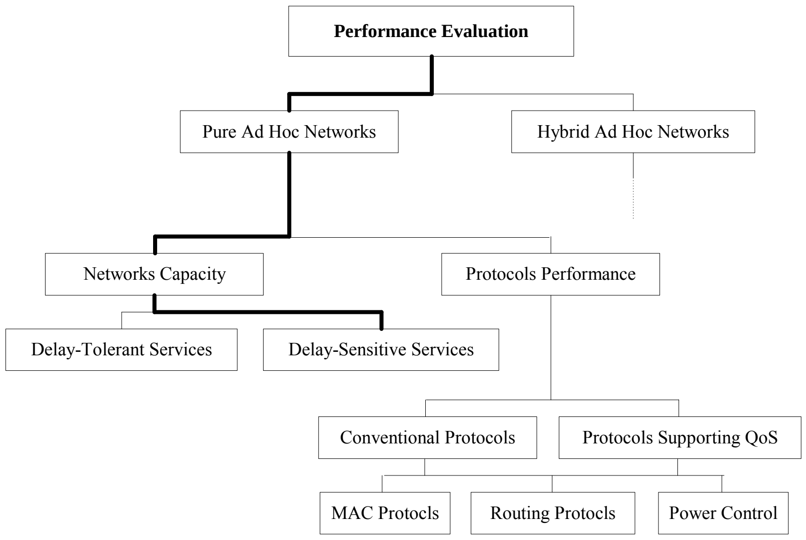


Figure 2.1： The taxonomy for performance evaluation in ad hoc networks

In this thesis, our research focus is the network capacity with end-to-end constraints, which is highlighted in bold in Figure 2.1. The first part of this chapter will introduce methods and results for capacity evaluation in ad hoc networks with particular emphasis on end-to-end delay constraints. In the latter part, we will elaborate on the performance study of MAC protocol IEEE802.11 because the delay introduced by it is an important component in end-to-end delay of a packet.

2.2 Overview for the network capacity evaluation

2.2.1 Background

In recent years, many studies have been done for capacity evaluation in ad hoc networks. Though these studies address various transmission scenarios and performance metrics of ad hoc networks, most of them focus only on the capacity of ad hoc networks carrying delay-tolerant services while ignoring the delay factor. These studies propose bounds for capacity metrics (described in section 2.2.1.1), provide methods to improve the capacity (described in section 2.2.1.2), analyze the network capacity under different traffic patterns (described in section 2.2.1.3) or study other capacity aspects (described in section 2.2.1.4). They provide us useful conclusions, good analysis methods and effective analysis models which are the bases of our research.

2.2.1.1 Throughput capacity study of ad hoc networks

The throughput capacity of a random wireless network is studied in [7], where fixed nodes are randomly placed in the network and each node sends data to a randomly chosen destination. The throughput capacity per node is given by $\Theta\left(\frac{W}{\sqrt{n\log n}}\right)$, as n approaches infinity, where n is the number of nodes in the network (the same below) and W is the common transmission rate of each node over the wireless channel. $f(n) = \Theta(g(n))$

denotes that $f(n) = \mathrm{O}(g(n))$ as well as $g(n) = \mathrm{O}(f(n))$. Thus the aggregate throughput capacity of all the nodes in the network is given by $\Theta\left(\sqrt{\frac{n}{\log n}}W\right)$.

The analysis for the upper bound is extended to a three-dimensional topology, which is expressed as follows [8].

$$R_{network} = \frac{(\sum_i R_i)}{T.(r_{ave-})} \qquad (2.1)$$

where R is the link rate, which maps the received SINR and T is the number of time slots. Equation 2.1 implies that the network capacity increases with the number of nodes although the throughput per-node decreases.

In addition, the aggregate throughput of a random three-dimensional wireless ad hoc network has been studied and proven as $\Theta\left(\left(\frac{n}{\log n}\right)^{\frac{2}{3}}W\right)$ [9].

These three papers [7], [8] and [9] all use throughput as their capacity metrics and derive the upper bound of the network throughput under different network structures. An important conclusion derived from their results is that if a specific minimum per user rate is required, the network cannot be arbitrarily large. This poses scalability issues in the analysis of network performances.

2.2.1.2 Methods to improve the network capacity

An analysis of the power consumption of the nodes to enhance the communication between the nearest neighbors is proposed in [10] . Assuming n nodes are placed in a unit area disk uniformly and independently and any pair of nodes can communicate between each other if and only if their distance is less than r(n), the resulting network will be asymptotically connected with probability 1 "*if and only if* $c(n) \rightarrow \infty$ " when each node covers an area of

$$\pi r^2(n) = (\log n + c(n))/n \qquad (2.2)$$

Grossglauser and Tse proposed a scheme that takes advantage of the mobility of the nodes [11]. By allowing only one-hop relaying, the scheme achieves an aggregate throughput capacity of $O(n)$ at the cost of unbounded delay and buffer requirement.

A method to increase network capacity without degrading the node throughput is provided by Carlos E. Caicedo B [12]. It adds n_B additional nodes that are inter-connected through a wired high-capacity network to act as relaying nodes only (i.e. base station). If each relaying node can transmit and receive W bits/sec, there will be a $\Theta(Wn_b)$ bit-meters/sec increment in the network capacity. This is the upper bound limit, assuming that each source/destination pair chooses optimum relaying nodes.

In the case of arbitrary network configurations, [12] gives a specific form of the best total capacity achievable in the network:

$$\textit{Total} \quad \textit{network} \quad \textit{capacity} \le \sqrt{\frac{8}{\pi}} * \frac{W}{\Delta}\sqrt{An} + \frac{n_B}{2}\overline{D} \qquad (2.3)$$

where A denotes the area of the nodes located in the region and $\overline{D}$ denotes the mean traversed distance between relaying nodes. This function implies that in order to improve the network capacity by a factor of m, a number of base station nodes proportional to $m\sqrt{n}$ should be added.

In summary, these three papers [10], [11] and [12] propose algorithms to improve the network capacity. Their conclusions give us intuition to design ad hoc networks.

2.2.1.3 Capacity analysis under different traffic pattern

[13] and [14] are two papers that evaluate network capacity under different traffic patterns.

Gastpar and Vetterli presented a capacity study under a special traffic pattern in [13]. There is only one active source and destination pair, while all remaining nodes serve as relays, assisting the transmission between the source and destination nodes. The capacity is shown to scale as $O(\log n)$.

Li et al. examined the effect of IEEE 802.11 on network capacity and presented specific criteria of the traffic pattern that makes the capacity scale with the network size [14]. In this paper, IEEE 802.11 distributed coordination function [15] is used as the access method in a static ad hoc network, i.e. the nodes in the network do not move significantly during packet transmission times.

Due to MAC interactions, the simulation results show that the capacity of node is less than the theoretically computed ideal results. As in the case of a chain of nodes, the ideal capacity is 1/4 as compared to the simulation result, 1/7. This result is also a consequence of the fact that nodes appearing earlier in the chain starve those appearing later.

A performance parameter, one-hop capacity, is defined in [14], which takes all radio transmissions for data packets that successfully arrive at their final destinations, including packets forwarded by intermediate nodes, into consideration. It is determined by the amount of spatial reuse, which is proportional to the physical area of the network. Letting C denote the total one-hop capacity of the network (proportional to the area), the capacity can be expressed as follow:

$$C = kA = k\frac{n}{\delta} \qquad (2.4)$$

where n is the number of nodes, δ is the node density and A is the physical area of network.

Because the total one-hop capacity in the network required to send and forward packets subject to condition $C > n \cdot \lambda \cdot \frac{\tau}{r}$, combining this with formula (2.4), the rate of each node originals packets λ (the capacity available to each node) can be obtained by:

$$\lambda < \frac{kr}{\delta} \cdot \frac{1}{L} = \frac{C/n}{L/r} \qquad (2.5)$$

where L is the length of physical path from source to destination, r is the fixed radio transmission range and $\frac{L}{r}$ is the minimum number of hops required to deliver a packet.

The inequality implies that as the expected path length increases, the available bandwidth for each node to originate packets decreases. Therefore, the traffic pattern has a great impact on scalability.

2.2.1.4 Other capacity analysis

Uysal-Biyikoglu and Keshavarzian explored the network capacity achievable with no relaying in a mobile interference network, i.e. via only direct communication [16]. In this scenario, sender/receiver pairs in the network are placed randomly in a region of unit area. The capacity is defined as the highest rate that can be achieved by each sender/receiver pair over a long period of time. The Gaussian interference channel and the TDMA scheme are used in their analysis. In addition to the results in [7], which provide the upper bound of the capacity, they derive the lower bound of network capacity, given by $O(\frac{\log(n)}{n})$.

Li et al. evaluated the capacity of ad hoc networks under various topologies [17] . The performance of ad hoc networks based on the 2Mbps IEEE 802.11 MAC is extensively examined for a single channel. In this paper, the scaling laws of throughput for large scale of ad hoc networks are presented and the theoretical guaranteed throughput bounds of per node are proposed for multi-channel, multi-hop ad hoc networks. Their protocol stacks are based on four layers: physical layer, multiple access control (MAC) layer, network layer, and application layer. However, their evaluations are concentrated on the effect of the MAC layer. In the network layer, a proactive shortest-path routing algorithm is used.

In summary, this section discusses several typical studies on network capacities that have been taken on the wireless ad hoc network under various scenarios. Their results obtained are useful to estimate the real capacity of the ad hoc network. The factors affecting the improvement of network transport capacity suggest the direction of the intending design and research.

2.2.2 Capacity evaluation with end-to-end delay requirements

In MANETs, transmission delay is a tradeoff with network capacity enhancements because of multihop routing. Comaniciu and Poor study the capacity of ad hoc networks supporting delay sensitive traffic [18]. Two capacity parameters are defined: (i) signal-to-noise ratio (SNR), which is the ratio between the transmitted power and the noise power, and (ii) parameter α, which reflects the physical layer capacity and is defined by the fixed ratio of nodes number N and normalized spreading sequences of length L. i.e. $\alpha = N / L$.

The discussed ad hoc network consists of N mobile nodes with uniform stationary distribution over a square area of dimension $D^* \times D^*$. It is denoted by a random graph G (N, p), where p is the probability of a link between any two nodes.

The authors derived the denotation of delay based on the assumption that since each packet travels only one hop during each time slots, in that the end-to-end delay can be measured as the number of hops required for a route to be completed. Both the throughput and the delay are influenced by the maximum number of hops allowed for a connection, and consequently by the network diameter D. Thus the delay constraints are mapped into a maximum network diameter constraint D.

Hence, the maximum average source-destination throughput is given by following equation, where W is system bandwidth.

$$T_{S-D} = \frac{W}{LD} \qquad (2.6)$$

The formula implies that the lower network diameter constraints will ensure lower transmission delays and higher source-destination throughput for the network.

From the results, general trends for capacity have been observed: the performance improves at both the physical and the network layer as the number of nodes in the network increases. This conclusion helps people to know how the capacity is decided on the whole.

In contrast to [18], Perevalov and Blum explored the influence of the end-to-end delay on the maximum capacity of a wireless ad hoc network confined to a certain area

[19]. The diversity coding approach in combination with the secondary diversity routing of [11] is used to asymptotically achieve the upper bound for a relaying strategy. Based on the node capacity C_∞ achieved by the one relay node approach in [11], the capacity under the constraint that the end-to-end delay does not exceed *d* is

$$C_d = \left(1 - e^{-\lambda d} - \left(\left(\frac{3\sqrt{3}\mu(Q)}{4} \right)^2 e^{-\lambda d} \lambda \tau \right)^{\frac{1}{3}} \right) C_\infty \qquad (2.7)$$

Both above two papers analyze the network capacity with end-to-end delay constraints. From their results, we can infer that the capacity degrades when the end-to-end delay constraints are guaranteed. As mentioned before, the transmission delay is a tradeoff with network capacity enhancement. Most studies improve the network capacity at the expense of increased transmission delay. It is more important to guarantee certain QoS in the systems that serve delay-sensitive applications.

2.3 Performance evaluation on IEEE802.11 MAC protocol

The medium access control (MAC) protocol performs the challenging tasks of resolving contention amongst nodes while sharing the common wireless channel for transmitting packets. The MAC protocol is an important factor that affects the performance of an ad hoc network. Since the emergence of ad hoc networks, a lot of MAC protocols have been adopted to direct the behavior on MAC layer and physical layer, such as Aloha, Carrier Sense Multiple Access (CSMA), TDMA, FDMA, CDMA, IEEE802.11 etc. However, IEEE802.11 standard has emerged as the leading WLAN protocol today. Its primary mechanism, referred to as Distributed Coordination Function (DCF), is a variant of CSMA.

Recently, two major performance domains of IEEE802.11 are studied: 1) IEEE 802.11 DCF and 2) hidden terminal problem in CSMA/CA.

There are three papers [20], [21] and [22] study the efficiency of the IEEE 802.11 protocol by investigating the maximum throughput that can be achieved under various network configurations. They analyze the backoff mechanism and propose alternatives to the extant standard mechanisms in order to improve system performance. Bianchi [20] presented a simple analytical model to compute saturation throughput performance assuming a finite number of stations and ideal channel conditions. Wu et al. [21] extended the same model and takes into account of the frame retry limits, which predict the throughput of 802.11 DCF more accurately. Furthermore, Rahman [22] built an analytical model based on Bianchi's original model of 802.11 DCF with station retry limits that accurately predicts the finite load throughput incorporating *ACK-timeout* and *CTS-timeout* parameters. In addition, he also designed an analytical model that incorporates presence of hidden terminals in static and dynamic environments for saturation and finite load throughput calculations.

Tobagi and Kleinrock [23] proposed a framework for modeling hidden terminals in CSMA networks. Let $i = 1, 2, \ldots, M$ index the M terminals. An $M*M$ square matrix $\mathbf{H} = [m_{ij}]$ is used to model hidden terminals, where the m_{ij} entry is given by:

$$m_{ij} = \begin{cases} 1 & \textit{if stations } i \textit{ and } j \textit{ can hear each other} \\ 0 & \textit{otherwise} \end{cases} \qquad (2.8)$$

Since stations that hear the same subset of the population behave similarly, stations with identical rows or columns are said to form groups. This framework is extended in [11] to accurately predict interference resulting from presence of hidden terminals.

Khurana, et. al. incorporated both hidden terminals and mobility of wireless stations into throughput calculations [24]. Their study implies that delay increases significantly in the presence of hidden terminals; using RTS/CTS to mitigate the effect of hidden terminals. However, this study lacks an analytical study to accurately predict throughput. Moreover, it only concentrates on the effects of hidden terminals and mobility on throughput and stations blocking probability through simulations.

Bianchi provided a straightforward but extremely accurate, analytical model to compute the 802.11 DCF throughput, assuming of finite number of terminals and ideal channel conditions [20]. Both the basic access and the RTS/CTS access mechanisms are analyzed. Backoff window size is modeled by the discrete-time Markov Chain whose states are denoted by $\{s(t), b(t)\}$, where $b(t)$ is the stochastic process representing the backoff time counter for a given station and $s(t)$ is the stochastic process representing the backoff stage $(0,\ldots,m)$ of the station at time t. The throughput S is defined as the fraction of time the channel used to successfully transmit payload bits and is expressed as:

$$S = \frac{P_s P_{tr} E[p]}{(1-P_{tr})\sigma + P_s P_{tr} T_s + P_{tr}(1-P_s)T_c} \qquad (2.9)$$

The results imply that the performance of the basic access method strongly depends on the system parameters, mainly minimum contention window and number of stations in the wireless network. On the other hand, performance is only marginally dependent on the system parameters when the RTS/CTS mechanism is considered.

Different from [20], which concentrates on the throughput, Carvalho et al. chose delay as the performance metric of IEEE 802.11 DCF [25]. They proposed an analytical model to calculate the average service time and jitter experienced by a packet when transmitted in a saturated IEEE 802.11 ad hoc network. They used a bottom-up approach and built the first two moments of a node's service time based on the IEEE 802.11 binary exponential backoff algorithm and the three possible events underneath its operation. In their results, the average backoff time is expressed as following:

$$\overline{T}_B = \frac{\alpha(W_{\min}\beta - 1)}{2q} + \frac{(1-q)}{q} t_c \qquad (2.10)$$

They also linearized Bianchi's model [20], and derived the simple formulas for these quantities in the expression. Their model is applied to the saturated single-hop networks with ideal channel conditions. A performance evaluation of a node's average service time and jitter is carried out for the DSSSS and FHSS physical layers. One conclusion is obtained that as far as delay and jitter are concerned, DSSS performs better than FHSS. They also conclude that the higher the initial contention-window size, the smaller the average service time and jitter are, especially for large networks, and the smaller the packet, the smaller the average service time and jitter are.

2.4 Conclusion

In this chapter, we first review the previous works on capacity evaluation based on diverse capacity models and capacity metrics. The capacity evaluations with end-to-end delay constraints in the ad hoc networks are emphasized. Then, the performance analysis for IEEE802.11 is discussed. Some major differences between these study efforts and ours are listed below.

Capacity metrics in the previous works depict the network capacity with respect to the packets, such as throughput, delay, and packet loss rate. In our work, we adopt a new metric to depict the network capacity in term of sessions. Through this metric we can find out the number of sessions that can be supported by ad hoc networks simultaneously under certain conditions. We also use different network models and mathematical model to analyze this capacity metric. The formulas for queuing delay caused by IEEE 802.11 and end-to-end delay are derived based on the results from some previous studies. These outline our contributions and highlight the major differences in our research as compared to other studies.

Chapter 3　Capacity Definition and Mathematical Model

3.1 Introduction

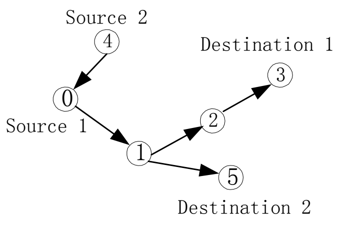
In this chapter, we define our capacity metric and build a mathematical model. Our capacity metric depicts the network capacity from the point of view of flows which not only shows the capacity of a network but can also be adopted to provide certain quality of service assurances. The mathematical model is built based on adjacency matrixes, which depict the topologies of networks.

3.2 Capacity Definition

We measure the network capacity by the number of sessions that can simultaneously exist in the network with a constraint on the end-to-end delay.

Definition: Session

A session is defined as one hop or several sequential hops within a path without differentiating source, destination, or intermediate nodes of the path (Figure 3.1). A session is called an *n*-hop session if it contains *n* hops.



Path 1: 0->1->2->3	
hop count	sessions
one-hop	0->1 1->2 2->3
two-hop	0->1->2 1->2->3
three-hop	0->1->2->3

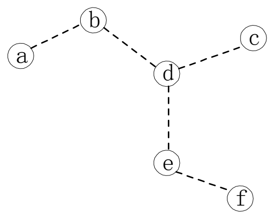
Figure 3.1:　Paths and the Sessions

One-hop path can be shared by multiple one-hop sessions as long as they transmit within their end-to-end delay constraints. We estimate the number of sessions simultaneously existing in the network without considering the number of paths that these sessions belong to. This capacity metric can serve as a reference for the acceptance of new communication requests and the value of the metric depends on both the available bandwidth of the channel and the end-to-end delay constraint of the delay-sensitive traffic. Furthermore, any source-destination pair containing these sessions satisfies both the maximum link sharing and end-to-end delay constraints.

3.3 Mathematical model

In our study, we assume that every source-destination pair in the ad hoc network communicates through a common broadcast channel using omni-directional antennas with the same transmission range. The topology of an ad hoc network can thus be modeled by an undirected graph G(V,E,**A**). V denotes the set of nodes in the network and $E \in V \times V$ denotes the set of links between nodes. For a link $(i, j) \in E$, its converse link $(j,i) \in E$ also exists. **A** is an adjacency matrix that depicts the topology of the network.

An *adjacency matrix* of a graph is a {0,1} matrix where the *ij*th entry is 1 if there is a link between *node i* and *node j* and zero otherwise [26]. In our scenario, the value is 1 if two corresponding nodes are within the transmission range of each other. Otherwise, the value is 0.



(A) Newwork topology

$$
\mathbf{A} = \begin{array}{c|cccccc}
 & a & b & c & d & e & f \\
\hline
a & 0 & 1 & 0 & 0 & 0 & 0 \\
b & 1 & 0 & 0 & 1 & 0 & 0 \\
c & 0 & 0 & 0 & 1 & 0 & 0 \\
d & 0 & 1 & 1 & 0 & 1 & 0 \\
e & 0 & 0 & 0 & 1 & 0 & 1 \\
f & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

(B) Adjacency matrix

Figure 3.2： The network topology and its Adjacency Matrix

Figure 3.2 (A) illustrates a simple topology of an ad hoc network. Each node is assigned a unique identifier. The dashed lines between any two nodes denote that they are within the transmission range of each other and shortest path between them is one hop. In this case, these two nodes are called one-hop neighbors of each other, for example, node *a* and node *b*. Expanding this concept, if shortest path between node *a* and *b* is *k* hops, we call them the *k*-hop neighbors of each other.

Figure 3.2 (B) is the corresponding adjacency matrix of the network shown in (A). In the matrix **A**, "1" denotes that two corresponding nodes are one-hop neighbors and "0" denotes they are out of the transmission range of each other. Since **A** only contains one-hop paths in this case, it is called a "*one-hop adjacency matrix*". In the ad hoc network, many shortest paths between sources and destinations are more than one hop. Thus, we extend the one hop adjacency matrix to the "*multi-hop adjacency matrix*" according to the following proposition in matrix theory [26].

<u>Proposition</u>:

Let G = (V,E) be a graph with vertex set $V = \{v_1, v_2, ... v_n\}$ and let $\mathbf{A}^k$ denote the *k*th power of the adjacency matrix. Let $a_{ij}^{(k)}$ denote the element of the matrix $\mathbf{A}^k$ at position

(i,j). Then $a_{ij}^{(k)}$ is the number of walks of length exactly k from the vertex v_i in the graph G.

We obtain multi-hop adjacency matrixes by the process as follows. Let

$$a_{ij}^{k+1} = \begin{cases} a_{ij}^{k} & if \quad a_{ij}^{k} > 0 \\ k+1 & if \quad a_{ij}^{k} = 0 \quad and \quad a_{ij}^{k+1} > 0 \end{cases}$$

This guarantees that all paths are shortest paths. We call this process as *Exact Multiplication* (*EM*) and express it as $[x \times x \times \cdots \times x]^*$.

$$\mathbf{A}^2 = \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline a & 1 & 1 & 0 & 2 & 0 & 0 \\ b & 1 & 1 & 2 & 1 & 2 & 0 \\ c & 0 & 2 & 1 & 1 & 2 & 0 \\ d & 2 & 1 & 1 & 1 & 1 & 2 \\ e & 0 & 2 & 2 & 1 & 1 & 1 \\ f & 0 & 0 & 0 & 2 & 1 & 1 \end{array} \qquad \mathbf{A}^3 = \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline a & 1 & 1 & 3 & 2 & 3 & 0 \\ b & 1 & 1 & 2 & 1 & 2 & 3 \\ c & 3 & 2 & 1 & 1 & 2 & 3 \\ d & 2 & 1 & 1 & 1 & 1 & 2 \\ e & 3 & 2 & 2 & 1 & 1 & 1 \\ f & 0 & 3 & 3 & 2 & 1 & 1 \end{array}$$

(A) 2-hop (B) 3-hop

Figure 3.3: 2-hop and 3-hop Adjacency Matrix

Figure 3.3(A) shows the 2-hop adjacency matrix $\mathbf{A}^2$ obtained by using Exact Multiplication on $\mathbf{A} \times \mathbf{A}$, where $\mathbf{A}$ is the matrix in Figure 3.2. It lists all the node pairs that can reach each other within two hops. Similarly, we can get $\mathbf{A}^3$ (Figure 3.3(B)), $\mathbf{A}^4$ till $\mathbf{A}^n$ where n is the largest hop count among all the shortest paths in the network.

Therefore, one-hop adjacency matrix shows the neighborhood of an ad hoc network. All adjacency matrixes can depict the topology of the network.

Chapter 4 The Upper Bound of Network Capacity

4.1 Introduction

As in chapter 3, the network capacity is measured by the number of sessions that can simultaneously exist in the network subjecting to the end-to-end delay constraint. Thus, in this chapter, the upper bound of the network capacity is the maximum number of sessions that can simultaneously exist in the network. In particular, the definitions of session and capacity are given as the follows.

Definition: session

$$Sess := \{n_1, n_2, ...n_m\}$$

where n_1 is the source, n_m is the destination, and n_j s are the intermediate nodes on the session following the packet-forwarding sequence.

Definition: session set

$$SessSet := \{Sess_1, Sess_2, ...Sess_n\}$$

Capacity Measurement: $|SessSet|$, which is the cardinality of the set *SessSet*.

Definition: capacity upper-bound

$$upper\_bound := \max(\ |\ SessSet\ |\ )$$

In ad hoc networks, obtaining the upper-bound of sessions that can exist simultaneously is an optimization problem. A brute-force search algorithm is a straightforward method to solve this kind of problems. The brute-force search algorithm [27] systematically enumerates every possible valid set of sessions until all possible sets have been exhausted. Finally, the algorithm can determine the maximum number of the sessions from these session sets. However, it is a hog of computation time, especially when the number of nodes in the network is large. It needs exponential time complexity to resolve the problems and only very small networks are amenable to this approach.

It is also unsuitable for our capacity estimations, because:

(1) nodes may join or leave an ad hoc network;

(2) the number of nodes in an ad hoc network could be very large; and

(3) we assume that network is stationary in the period of capacity estimation, so a lengthy computation process will invalidate the capacity results because network topology may change frequently.

Therefore, we need to design heuristic algorithms to obtain the capacity value that can be closely approximated to the results of brute-force search algorithm with low time complexities.

In this chapter, we present two capacity computation algorithms based on one-hop and multi-hop adjacency matrices to compute the upper bound of network capacity for two different scenarios [6]. One is the non-channel-sharing scenario, where each channel is used by one session, and the other is the channel-sharing scenario, where a channel is shared by multiple sessions running through it. The latter scenario is closer to real ad hoc network scenarios. The algorithm for the non-channel-sharing scenario has been designed to verify the validity of our basic arithmetic through simple scenarios.

Our algorithms are based on an assumption [18], which is at each time slot packets travel one hop, such that the end-to-end delay can be measured as the number of hops required for a route to be completed. In addition, the topology of the network is assumed to be known by signaling on each node.

To address mobility, our algorithms coordinate computation in an on-demand fashion. Capacity computation is only performed when it is required or when some new flows request admission.

4.2 Capacity Computation for Non-channel-sharing scenario

This section focuses on the non-channel-sharing scenario, where each channel is used by only one session. Each session belongs to only one path so that if each session is seen as a path with the same hop count, the number of sessions equals to the number of paths.

4.2.1 Algorithm description

In this scenario, any two paths that exist in the system within a particular time period are separated.

The one-hop and multi-hop adjacency matrices depict the topology of the ad hoc network and the shortest paths between two arbitrary nodes. Based on them, we design the Matrix Select-Delete Algorithm (MSDA), as shown in Figure 4.1.

Matrix Select-Delete algorithm is a 1-level greedy algorithm that comprises a series of selection iterations. Rules (1) and (2) guarantee that the maximum available nodes remain after one selection in order to obtain maximum number of paths. Rule (3) is designed according to the transmission property of the wireless ad hoc networks as shown in Figure 4.2. [14]. If node 1 is transmitting to node 2, node 3 cannot transmit since node 2 is also in the transmission range of node 3. Any transmission of node 3 will result in node 2 not being able to receive the packet from node 1 correctly due to interference and collision. However, node 4 can transmit simultaneously because node 2 is out of its transmission range, and will not be affected by its transmission.

```
Begin [Matrix Select-Delete Algorithm (MSDA)]
    Input number of nodes ($n$) and one-hop adjacency matrix ($\mathbf{A}(n,n)$)
    Input hop count of the available paths ($k$)
    Compute $\mathbf{B}(n,n) = \left[ \underbrace{\mathbf{A}(n,n) \times \mathbf{A}(n,n) \times \cdots\cdots \times \mathbf{A}(n,n)}_{k} \right]^*$
    Store all the paths in PathSet;
    SelectedPaths := ∅;
    While (PathSet <>∅)
    {   (1)source := select the node with the fewest one-hop neighbors;
        (2)dest := select the node, which is one of k-hop neighbors of the source and
            has the fewest one-hop neighbors;
        AddPath(SelectedPaths, source, dest): Add path originating from source to
            dest, to SelectedPaths;
        (3) B(n,n) := delete all the columns and rows of the source and destination, relay
            nodes and their one-hop neighbors;
        PathSet := delete all corresponding paths according to the deletion of B(n,n)
            from PathSet;
    }
    Output (SelectedPaths);
End
```

Figure 4.1： Matrix Select-Delete Algorithm

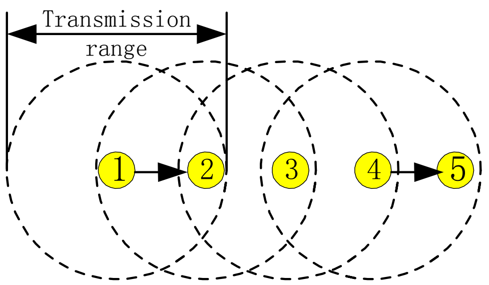


Figure 4.2： Transmission property (When node 1 is transmitting, then node 4 can transmit simultaneously while node 2, 3 cannot due to collision.)

4.2.2 Algorithm validation for MSDA

To verify the correctness of Matrix Select-Delete Algorithm (MSDA), we compare its results with those of a brute-force search algorithm in choosing the shortest paths in the same scenarios.

Ten scenarios are used in our simulations, where the i_{th} scenario is the one with all the valid shortest paths having i hops (i = 1, 2…10). Figure 4.3 shows an ad hoc network with 26 nodes and the average number of neighbors is 3 with the corresponding simulation results shown in Figure 4.4. In Figure 4.5, the network has 40 nodes and the average number of neighbors per node is 3. The simulation results are illustrated in Figure 4.6.

Figure 4.4 and Figure 4.6 show that MSDA can obtain the results close to that of the brute-force search algorithm with a deviation of one, with difference in time complexities. Given N nodes and k hops between all sources and destinations, the time complexity of the brute-force search algorithm is $\mathrm{O}\left([N/(k+1)]^N\right)$ while that of MSDA is $\mathrm{O}\left(N^2/k\right)$. Therefore, the time required by MSDA is much shorter than that of the brute-force search algorithm.

Due to the mobility property, resulting in the frequent changes in network topology, our algorithm can therefore effectively estimate the current network capacity, making it feasible for real-time capacity estimation.

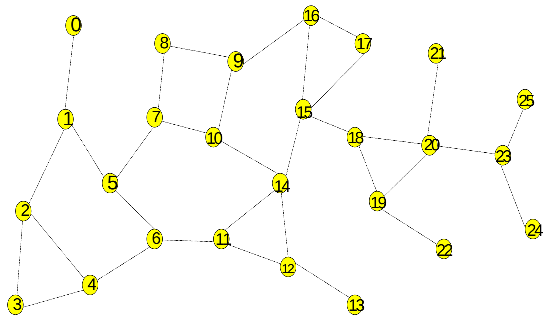


Figure 4.3: Simulation Topology I (26 nodes, the average number of neighbors: 3)

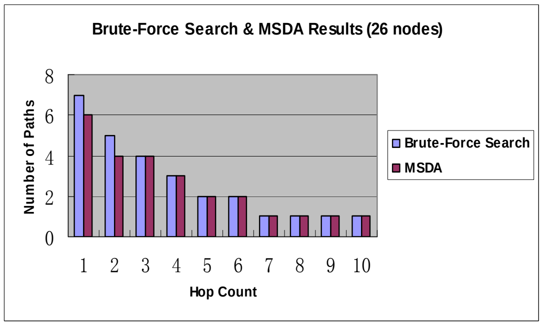


Figure 4.4: Brute-Force Search & MSDA Results I (26 nodes)

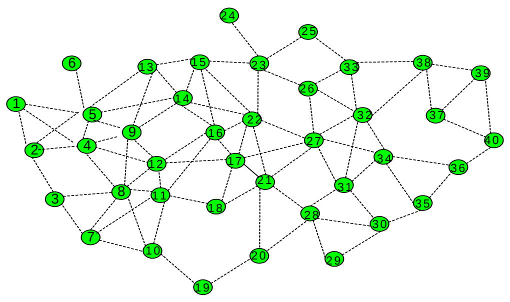


Figure 4.5: Simulation Topology II (40 nodes, the average number of neighbors: 3)

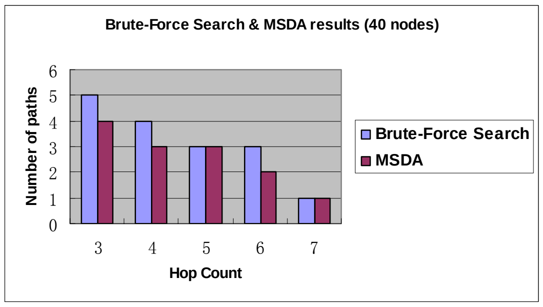


Figure 4.6: Brute-Force Search & MSDA Results II (40 nodes)

4.3 Capacity Computation for Channel-sharing scenario

In this section, we focus on the scenario where two or more paths share a common channel. The capacity here is the maximum number of one-hop sessions with channel bandwidth constraints and end-to-end delay constraints, with packets assumed to be transmitted one hop in one time slot.

4.3.1 Average hop count algorithm

In order to implement the end-to-end delay constraint in an ad hoc network, we map the end-to-end delay constraint to the hop-by-hop delay constraint by applying the formula below:

$$Hop-by-Hop \quad delay \quad constra\ int = \frac{End-to-End \quad delay \quad constra\ int}{Average \quad Hop \quad count}$$

Average hop count (AHC) is an indicator of the statistical situation of the paths in the network.

$$Average \quad hop \quad count = \frac{\sum\limits_{\substack{All \ possible \\ communication}} hop \quad count \quad of \quad shortest \quad path}{number \quad of \quad source-destination \quad pairs}$$

In order to calculate an accurate average hop count, we propose an algorithm based on adjacency matrices of the network (Figure 4.7). Assuming the number of nodes in the ad hoc network is *n* and the network diameter is *l*.

```
Begin
    input $dc$ (end-to-end delay constraint) and adjacency matrix $\mathbf{A}(n,n)$;
    $k = min[dc,l]$;
    compute $\mathbf{B}(n,n) = \left[\underbrace{\mathbf{A}(n,n)\times\mathbf{A}(n,n)\times\cdots\cdots\times\mathbf{A}(n,n)}_{k}\right]^{*}$;
    for all value $i$ in the matrix $\mathbf{B}(n,n)$, i.e. $i$ = 0, 1, 2……n-1
        compute $sum = \sum_{i\in B(n,n)} i$ ;
    compute $AHC = \left\lceil \frac{sum}{n^2-n-n_0} \right\rceil = \left\lceil \frac{\sum_{i\in B(n,n)} i - n}{n^2-n-n_0} \right\rceil$;
End
```

Figure 4.7: Average hop count algorithm

In Figure 4.7, n_0 denotes the number of the items with value 0 in the matrix **B**(*n*,*n*), and $\lceil x \rceil$ gives the smallest integer greater than or equal to *x*.

4.3.2 Capacity Estimation

Based on the notion of average hop count and link sharing, we extend our algorithm to be applied in more pervasive scenarios. First, we define the following:

Number of nodes is *N*.

Transmission radius of node is ***r*.**

Bandwidth of the channel is BW_{node}.

Bandwidth needed by a transmission of a packet is BW_{packet}.

End-to-End Delay Constraint is D_E.

The Hop-by-Hop Delay Constraint D_H, is given by:

$$D_H = \frac{D_E}{AHC} = \frac{D_E}{\dfrac{\sum\limits_{i \in B(n,n)} i - n}{n^2 - n - n_0}} = \frac{D_E(n^2 - n - n_0)}{\sum\limits_{i \in B(n,n)} i - n}$$

The number of one-hop sessions that the channel can support is $\lfloor BW_{node} / BW_{packet} \rfloor$. In addition, under end-to-end delay constraints, the number of one-hop sessions that share the same channel is:

$$N_s = \min\left[\left\lfloor \frac{BW_{node}}{BW_{packet}} \right\rfloor , \frac{D_E(n^2 - n - n_0)}{\sum\limits_{i \in B(n,n)} i - n} \right]$$

Based on these assumptions and formulas, we propose the Channel-share Select-Delete Algorithm (CSDA), shown in Figure 4.8.

```
Begin [Channel-sharing Select-Delete Algorithm (CSDA)]
    Input number of nodes (n) and adjacency matrix (A(n,n))
    Select paths from A(n,n), and store all the paths in PathSet;
    SelectedPaths := ∅;
    While (PathSet <> ∅)
    {
        source := select the node with the fewest one-hop neighbors;
        dest := select the node, which is the one-hop neighbor of the source, and has the
            fewest one-hop neighbors;
        AddPath(SelectedPaths, source, dest): Add path originating from source to dest,
            to SelectedPaths;
        A(n,n) := delete all the columns and rows of the source destination and their
            one-hop neighbors.
        PathSet := delete all corresponding paths according to the deletion of A(n,n)
            from PathSet;
    }
    count := the number of elements in set SelectedPaths;
    Compute $N_{session} = \left\lfloor count \times \min\left[ \left\lfloor \frac{BW_{node}}{BW_{packet}} \right\rfloor , \frac{D_E(n^2 - n - n_0)}{\sum_{i \in B(n,n)} i - n} \right] \right\rfloor$    (1);
    Output ($N_{session}$);
End
```

Figure 4.8： Channel-sharing Select-Delete Algorithm (CSDA)

$\lfloor x \rfloor$ denotes the largest integer number that does not exceed

4.3.3 Algorithm validation for CSDA

We present two sets of simulations based on two 26 nodes network topologies. All flows have the same transmission rate of 750 Kbps, and the network channel bandwidth is 2 Mbps. The average hop count of the ad hoc network in Figure 4.9 is 4, and the number of one-hop sessions without channel sharing is 6 according to the MSDA (Figure 4.4).

We choose ns-2 (network simulator) as the simulation tool [28]. In the simulation, we randomly add single flows with shortest path hop count smaller than the end-to-end delay constraint into the network, until the network is saturated. Each new flow is added on the condition that it does not cause any on-going flow to violate the end-to-end delay constraint. The number of one-hop sessions is calculated according to Figure 4.2, where two one-hop sessions is sharing the flow from node 1 to node 5. Generally, the number of one-hop sessions on an ***n***-hop path is $\lceil n/3 \rceil$. Simulation results are shown in Figure 4.10 while Figure 4.12 shows the simulation results of another ad hoc network with topology shown in Figure 4.11.

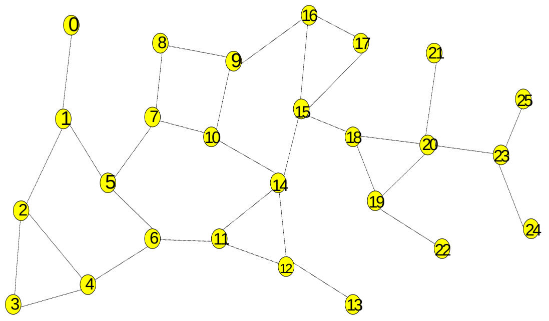


Figure 4.9: The Ad Hoc Network Topology (I)

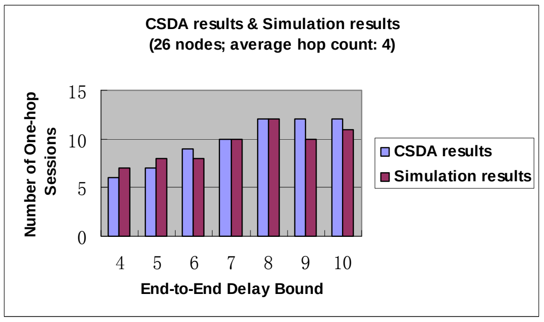


Figure 4.10: CSDA results & Simulation results for topology (I)

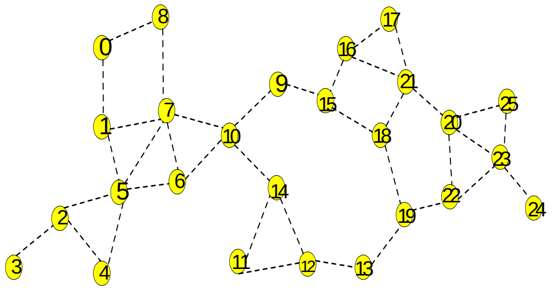


Figure 4.11: The Ad Hoc Network Topology (II)

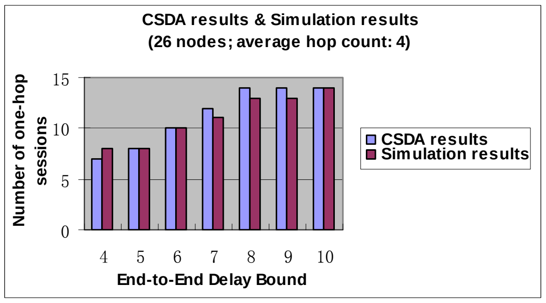


Figure 4.12: CSDA results & Simulation results for topology (II)

In Figure 4.10 and Figure 4.12, the network capacity obtained from CSDA approximates that of simulations with a deviation of one except for the case of the topology in Figure 4.9 when the end-to-end delay constraint is 9 hops.

Simulation results show that when the end-to-end delay constraint is equal to or larger than 8 hops, the number of one-hop sessions the network can support is similar, because the numbers of one-hop sessions sharing one channel are also bounded by the limited bandwidth of the channel. Therefore, even though the end-to-end delay constraint is allowed to increase, the number of simultaneously existing one-hop sessions in the network does not change.

4.4 Conclusions

In this chapter, we have proposed two algorithms: the Matrix Select-Delete Algorithm (MSDA) and the Channel-sharing Select-Delete Algorithm (CSDA) to obtain the network capacity for the non-channel-sharing and channel-sharing scenario respectively. We have also proposed an average hop count algorithm by calculating the probabilities of each possible shortest path hop count. The results obtained from MSDA and CSDA are very similar to those obtained from brute-force search algorithm. However, in contrast to brute-force search algorithm, our algorithms are much more efficient. From formula (1), we can see that the capacity of an ad hoc network is restricted by the bandwidth of channels as well as the end-to-end delay constraint. When the end-to-end delay constraint is small, it limits the number of sessions sharing the same channel. By increasing the end-to-end delay constraint, the network capacity is limited mainly by the bandwidth of the channel. Therefore, the capacity cannot increase unlimitedly. From formula (1), we can also infer that the smaller the average hop count of the flows existing in the network, the more simultaneous one-hop sessions can be supported.

In both MSDA and CSDA, we do not consider the interference among the nodes. Thus, the capacity calculated by them is the upper bound of capacity achievable in real ad hoc networks. In addition, in order to describe the basis of our algorithms more clearly, we have not addressed the delay introduced by the channel contention among nodes that will be discussed in the next chapter.

Chapter 5 Delay Analysis for IEEE 802.11 MAC

5.1 Introduction

The end-to-end delay estimation is the key in the capacity evaluation with the end-to-end delay constraints. The end-to-end delay (D_{ete}) of a flow consists of hop-by-hop delays (D_{hbh}) of each hop this flow run through. D_{hbh} comprises of the queuing delay (D_q), service time (T_{serv}) (shown in following equations).

$$D_{ete} = \sum_{\substack{all\ the\ hops \\ one\ the\ path}} D_{hbh}$$

$$D_{hbh} = D_q + T_{serv}$$

Queuing delay is the time period from the moment a packet enters a queue to the moment it leaves the queue. The latter moment is determined by the service time of other packets that are already in the queue when this packet enters. Service time of a packet is the time period when a node begins to compete channel for this packet till this packet reaches next hop node successfully. Therefore, before we can obtain hop-by-hop delay, we must obtain the service time which is mainly determined by the MAC protocol, IEEE802.11 in our case.

In the previous chapter, the delay caused by channel contention is ignored which will be concentrated on in this chapter. In the follow sections, we will first introduce the principle of IEEE802.11. Following that, the analytical process and formulas of service time and maximum queuing delay are presented based on an example.

5.2 Overview of the IEEE 802.11 MAC

The IEEE 802.11 MAC layer is responsible for a structured channel access scheme and is implemented using a Distributed Coordination Function (DCF) based on the Carrier Sense Medium Access with Collision Avoidance (CSMA/CA) protocol. An alternative to DCF, the Point Coordination Function (PCF), which supports collision free and time bounded services, is also provided in the form of a point for determining the user having the right to transmit. However, because PCF cannot be used in multihop or single-hop ad hoc networks, DCF is widely used, but incurs varying delays for all traffic. In this section, we will only describe the relevant details of the DCF access method. A more complete and detailed description is given by the 802.11 standard [15].

The DCF describes two techniques for packet transmission: the default, a two-way handshake scheme called basic access mechanism, and a four-way handshake mechanism [29].

5.2.1 Basic access mechanism

In IEEE 802.11, priority access to the wireless medium is controlled by the use of inter-frame space (IFS) time between the transmissions of frames. A total of three IFS intervals have been specified by the 802.11 standard: short IFS (SIFS), point coordination function IFS (PIFS), and DCF-IFS (DIFS). The SIFS is the shortest and the DIFS is the longest.

In the basic access mechanism, a node monitors the channel to determine if another node is transmitting before initiating the transmission of a new packet. If the channel is idle for an interval of time that exceeds the Distributed Inter Frame Space (DIFS), the packet is transmitted. If the medium is busy, the station defers until it senses that the channel is idle for a DIFS interval, and then generates a random backoff interval for an additional deferral time before transmitting. The backoff timer counter is decreased as long as the channel is sensed idle, frozen when the channel is sensed busy, and resumed when the channel is sensed idle again for more than one DIFS. A station can initiate a transmission when the back-off timer reaches zero. The back-off time is uniformly chosen in the range (0,*w*-1). Also (*w*-1) is known as the Contention Window (*CW*), which is an integer within the range determined by the PHY characteristics minimum Contention Window CW_{min} and maximum Contention Window CW_{max}. After each unsuccessful transmission, *w* is doubled, up to a maximum value $2^m * CW_{min}$, where *m* is maximum backoff stage.

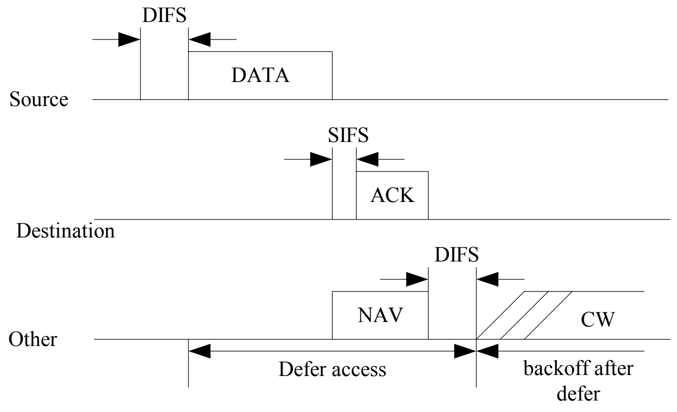


Figure 5.1： Basic access mechanism in DCF

Upon receiving a packet correctly, the destination station waits for a SIFS interval immediately following the reception of the data frame and transmits a positive ACK back to the source station, indicating that the data packet has been received correctly (Figure 5.1). In the case where the source station does not receive an ACK, the data frame is assumed to be lost and the source station schedules the retransmission with the contention window for doubled back-off time. When the data frame is transmitted, all the other stations on hearing the data frame adjust their Network Allocation Vector (NAV). The NAV is used for virtual carrier sensing at the MAC layer and is based on the duration field value in the data frame that was received correctly, which includes the SIFS and the ACK frame transmission time following the data frame.

5.2.2 Four-way handshake mechanism

In 802.11, DCF also provides an alternative way of transmitting data frames that involve transmission of special short RTS and CTS frames prior to the transmission of actual data frame. As shown in Figure 5.2, an RTS frame is transmitted by a station which needs to transmit a packet. When the destination receives the RTS frame, it will transmit a CTS frame after SIFS interval immediately following the reception of the RTS frame. The source station is allowed to transmit its packet only if it receives the CTS correctly. Here, it should be noted that all the other stations are capable of updating the NAVs based on the RTS from the source station and the CTS from the destination station, which helps to combat the *hidden terminal* problems. In fact, a station that is able to receive the CTS frames correctly can avoid collisions even when it is unable to sense the data transmissions from the source station. When the destination received the packet, it will send ACK back to the source.

In our research, we mainly study the RTS/CTS access method.

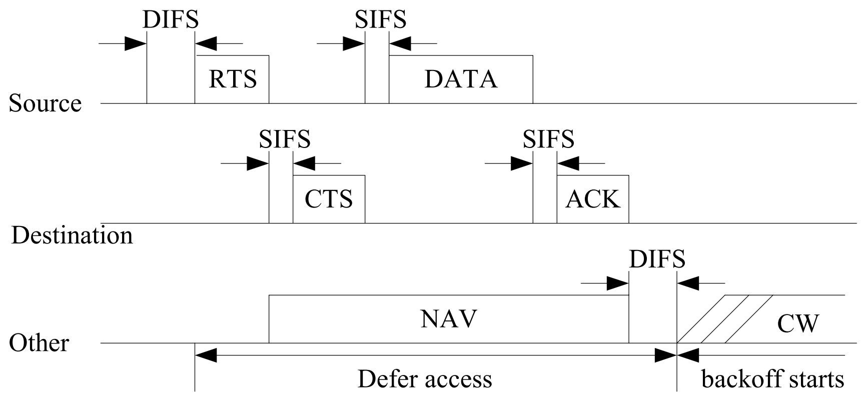


Figure 5.2： RTS/CTS access mechanism in DCF

5.3 Delay Analysis

As described above, $D_{hbh} = D_q + T_{serv}$, we concentrate on the expressions of service time and maximum queuing delay on each hop in the network in this section. They can be obtained by studying the events that may occur within a generic slot time and the queue's situation of each node. The analysis is divided into two parts. First we used results in [20] and [25] to study the behavior of a single station. Then we analyze the maximum queuing delay based on these results.

5.3.1 Service Time Characterization

Service time consists of two components: backoff time and transmission time. Average backoff time (T_B) has been solved in [20] and [25] by using three variables: average backoff step size (α), the probability of a collision seen by a packet being transmitted in the channel (*p*) and the probability that a station transmits in a randomly chosen slot time (τ). The average backoff time can be expressed as following equation:

$$T_B = \frac{\alpha(CW_{\min}\beta - 1)}{2q} + \frac{(1-q)}{q} t_c$$

The details for the computation of these variables are described in the following sections.

5.3.1.1 Average backoff step size (α)

In IEEE 802.11, once a node goes to backoff, its backoff time counter decrements according to the perceived state of the channel. If the channel is sensed idle, the backoff time counter is decremented. Otherwise, it is frozen, remaining in this state until the channel is sensed idle again for more than a DIFS, after which its decrementing operation is resumed. While the backoff timer is frozen, only two mutually exclusive events can happen in the channel: either a successful transmission takes place or a packet collision occurs. Therefore, there are three possible events a node can sense during its backoff:

E_s = {successful transmission}, E_i = {idle channel}, and E_c = {collision}

Each of the time intervals between two consecutive backoff counter decrements, which are called "backoff steps", will contain one of these three mutually exclusive events. In other words, during a node's backoff, the *j*-th "backoff step" will result in a collision, or a transmission, or the channel being sensed idle. Events are assumed to be independent in successive backoff steps, which is reasonable if the time a node spends on collision resolution is approximately the same as the time the channel is sensed busy due to collisions by non-colliding nodes [25]. We also use the same assumption in our research.

Assume that the event E_s, E_i, and E_c have probabilities $p_s=P\{E_s\}$, $p_i=P\{E_i\}$, and $p_c=P\{E_c\}$ and have average time *ts*, σ and *ti*, respectively. These events are independent and mutually exclusive at each backoff step, then the average backoff step size (α) is expressed as:

$$\alpha = \sigma p_i + t_c p_c + t_s p_s$$

Now, we turn to the problem of finding the conditional channel probabilities, represented here by p_s, p_i, and p_c. For this purpose, let P_{tr} be the probability that there is at least one transmission in the considered time slot when n nodes share the channel. Because the probability that a station transmits in a randomly chosen slot time is τ, we have

$$P_{tr} = 1 - (1-\tau)^n$$

The probability P_{suc} that a transmission occurring on the channel is successful is given by the probability that exactly one node transmits in the channel, conditioned on the fact that at least one node transmits, i.e.

$$P_{suc} = \frac{C_n^1 * \tau * (1-\tau)^{n-1}}{P_{tr}} = \frac{n * \tau * (1-\tau)^{n-1}}{1-(1-\tau)^n}$$

Therefore, the probability that a successful transmission occurs in a given time slot is $p_s = P_{tr} P_{suc}$. Accordingly, $p_i = 1 - P_{tr}$ and $p_c = P_{tr}(1 - P_{suc})$.

For the time intervals *ts* and *ti,* we follow the definition given by Bianchi [20], where

$$ts = RTS + SIFS + \delta + CTS + SIFS + \delta + H + E\{P\} + SIFS + \delta + ACK + DIFS + \delta$$

where $E\{P\}=P$ for fixed packet sizes and δ is the propagation delay.

$$tc = RTS + DIFS + \delta$$

5.3.1.2 Two probabilities: *p* and τ.

Bianchi [20] uses a Markov Chain model for the backoff window size. By analyzing the Markov Chain, he obtained two important probabilities: p and τ.

$$\tau = \frac{2(1-2p)}{(1-2p)(CW_{\min}+1) + pCW_{\min}(1-(2p)^m)}$$

where *m* is maximum backoff stage, $CW_{\max} = 2^m CW_{\min}$.

p can also be seen as the probability that, in a time slot, at least one of the *n*-1 remaining stations transmit. Fundamental independence is assumed so that each transmission "sees" the system in the same state, i.e., in steady state. According to the description above, at steady state, each remaining station transmits a packet with probability τ, which yields

$$p = 1-(1-\tau)^{n-1}$$

These two equations form a nonlinear system in the two unknown τ and *p*. However, the author shows that this system has a unique solution.

An approximate solution to this nonlinear system is obtained by linearizing both equations [25]. Through two intermediate variables, one is the probability γ that a node does not transmit in a randomly chosen slot time and another is the probability of success q that a packet experiences when it is transmitted at the end of the backoff stage, where $\gamma = 1-\tau$ and $q = 1-p$, they lead to the following approximation for *p*:

$$p = \frac{2CW_{\min}(n-1)}{(CW_{\min}+1)^2 + 2CW_{\min}(n-1)}$$

From the formula above, $\tau = \dfrac{2(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)}$, the value of p and τ can be obtained if there is a certain value of *n*, which is the number of nodes sharing the same channel.

5.3.1.3 Average backoff time T_B

Based on α and *q*, the average backoff time (T_B) is given by [25].

$$T_B = \frac{\alpha(CW_{\min}\beta - 1)}{2q} + \frac{(1-q)}{q} t_c$$

where $q = 1 - p = \frac{(CW_{\min}+1)^2}{(CW_{\min}+1)^2 + 2CW_{\min}(n-1)}$, $\alpha = \sigma p_i + t_c p_c + t_s p_s$ and

$\beta = \frac{q - 2^m (1-q)^{m+1}}{1 - 2(1-q)}$.

5.3.2 Maximum Queuing delay

In this section, the maximum queuing delay under various scenarios is estimated. In these scenarios, the numbers of nodes sharing the same channel with the node of interest are different. In the following sub-sections, we first introduce the queuing model in our study. After that, analytical process is explained using two-hop network scenarios as an example.

Similar to other work reported in the literature, we also use the maximum average queuing delay to substitute the maximum queuing delay, because the former is quantifiable in analysis, while the latter would be infinite in theory. Hence, in this thesis, the term maximum queuing delay refers to the maximum average queuing delay.

5.3.2.1 Queuing model

We propose an analytical model based on a discrete time *G/G/*1 queue which allows the capacity evaluations being carried for general traffic arrival patterns and arbitrary number of users. Our model is different from those in other studies, most of which assume the arrival rate of packets in the network is Poisson distribution so that they choose M/G/1 or M/D/1 as the queuing model. The reason we choose this model is that not all the packet arrival rates follow the Poisson distribution and general distributed arrival rates can cover all the scenarios including Poisson distribution. We have modeled the service rate distribution under the general distribution because when a node needs to compete for a channel with an uncertain number of other nodes, its service rate cannot be concluded as an assured distribution.

There is a useful bound that has been developed for the waiting time in queue W_q in G/G/1 queue. This can then be used to find bounds on the queuing delay D_q in the usual fashion, i.e. Little's Result. First, we assume that:

λ is average arrival rate of packets (general arrival process);

T is the (random) inter-arrival time with: $\begin{cases} E\{T\} = 1/\lambda \\ \sigma_T^2 = E\{T^2\} - [E\{T\}]^2 \end{cases}$

X is the (random) service time (general service time distribution) with

$$\begin{cases} E\{X\} = \overline{X} \\ \sigma_X^2 = E\{X^2\} - [E\{X\}]^2 \end{cases}$$

$\rho = \lambda \overline{X} = \mathit{Traffic \quad Offered}$. When $\rho < 1$, the queue can be stable.

If the mean and variance (or second moment) of the inter-arrival times and the service times are known, then the following bound has been shown to hold for D_q, the queuing delay of any G/G/1 queue.

$$D_q \le \frac{\lambda(\sigma_X^2 + \sigma_T^2)}{2(1-\rho)}$$

In our analysis process, we try to obtain maximum queuing delay at each node. The main work is to find out all the means and variance of the inter-arrival time and the service time. The following describes the delay analysis for two hops network scenario.

5.3.2.2 Maximum queuing delay analysis for two hops multi-source scenarios

In this section, the target is analyzing the maximum queuing delay on the relay node in the network with different number of flows where maximum hop count of paths is 2 (Figure 5.3). The analysis of this network scenario is the fundamental method used in our research.

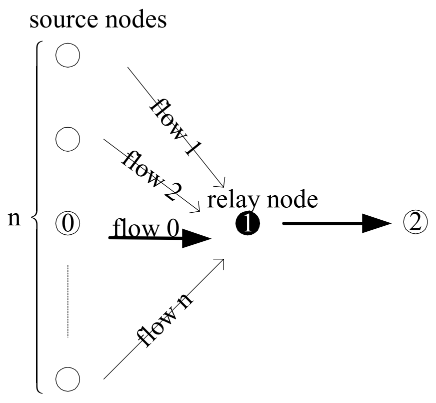


Figure 5.3: The two hops path in the network with multiple sources

In Figure 5.3, *n* flows go to the destination node 2 through the relay node 1. We focus on the end-to-end delay of flows on the path "node 0 → node 1 → node 2". Besides the transmission time from node 0 to node 1 and service time from node 1 to node 2, the queuing delay on node 1 is most difficult to estimate. As previously discussed, we need to determine the mean and variance of inter-arrival times and service times for node 1 in order to evaluate the queuing delay at it.

(A). Average arrival rate (λ) and average service rate (μ)

In most previous work, the parameters in IEEE 802.11 are determined under saturated network conditions, in which all the nodes always have packets to send. In our research, we estimate these parameters under random traffic loads in the networks. Besides the same parameters in [20] and [25], we provide every node with two more parameters: the probability that it has packets to send and the probability that it does not have packets to send. Whether a queue is empty or not is assumed to be independent of other queues.

Table 1 includes the parameter definitions for our scheme. Since all the sources are subject to the same situations, they have the same values as their parameters.

At the source nodes, the average arrival rate equals to the average rate at which packets are generated, namely $\lambda_0 = average\_packet\_generation\_rate$.

Service rate shows the ability of a node in processing packets. In our scenario, the average service rates of the sources can be calculated by analyzing the all possible occurring cases (Table 2).

Therefore, $\mu_0 = \sum_{i=0}^{n-1} C_{n-1}^{i} * pw^{i} * py^{n-1-i} * \left[(1-xx) * \frac{1}{td_{n-i}} + xx * \frac{1}{td_{n+1-i}} \right]$

From the average arrival rate and average service rate of source, we can derive the two probabilities pw and py.

We approximate the probability that the queue of a node is empty as $pw = 1 - \frac{\lambda_0}{\mu_0}$, which is exact for the *M/M/*1 case.

At the relay node 1 which is the focus here, the calculation of average arrival rate is no longer the average rate at which packets are generated. It should be calculated using the same analysis method with the calculation for the average service rate of sources. All possible cases are shown in Table 3. The average arrival rate of relay node can be expressed as follows:

$$avearrirate = \lambda_1 = \sum_{i=0}^{n-1} C_n^i * pw^i * py^{n-i} * \left[(1-xx) * \frac{n-i}{td_{n-i}} + xx * \frac{n-i}{td_{n+1-i}} \right]$$

The probability that the queue of node 1 is empty can be expressed as: $xx = 1 - \frac{\lambda_1}{\mu_1}$.

Similarly, by summarizing all the cases in Table 4, the average service time for relay node 1 can be obtained. $aveservrate = \mu_1 = \sum_{i=0}^{n-1} C_n^i * pw^i * py^{n-i} * \frac{1}{td_{n+1-i}}$.

Table 1： The parameter definitions

λ_0	Average arrival rate of source nodes.
μ_0	Average service rate of source nodes.
pw	The probability that source node does not have packets in queue.
py	The probability that source node has packets in queue. *py*=1-*pw*.
λ_1	Average arrival rate of relay node.
μ_1	Average service rate of relay node.
xx	The probability that relay node has packets in queue.
1-*xx*	The probability that relay node does not have packets in queue.
tb_i	The average backoff time when *i* nodes share one channel
td_i	The average service time when *i* nodes share one channel. $td_i = tb_i + ts$

Table 2： All possible cases for the average service rate of source

Cases		Service rate	Probability
Among remainder (*n*-1) sources, *i* sources do not have packets while (*n*-1-	the relay node does not have packet to send	$\frac{1}{td_{n-i}}$	$\sum_{i=0}^{n-1} C_{n-1}^i * pw^i * py^{n-1-i} * (1-xx)$

i) sources have;	the relay node has packets to send	$\frac{1}{td_{n+1-i}}$	$\sum_{i=0}^{n-1} C_{n-1}^{i} * pw^{i} * py^{n-1-i} * xx$

Table 3： All possible cases for the average arrival rate of relay node 1

Cases		Service rate	Probability
i sources do not have packets and (*n*-*i*) sources have;	the relay node does not have packets to send	$\frac{n-i}{td_{n-i}}$	$\sum_{i=0}^{n-1} C_{n}^{i} * pw^{i} * py^{n-i} * (1-xx)$
	the relay node has packets to send	$\frac{n-i}{td_{n+1-i}}$	$\sum_{i=0}^{n-1} C_{n}^{i} * pw^{i} * py^{n-i} * xx$

Table 4： All possible cases for the average service rate of relay node 1

Cases	Service rate	Probability
i sources does not have packets and (*n*-*i*) sources have	$\frac{1}{td_{n-i+1}}$	$\sum_{i=0}^{n-1} C_{n}^{i} * pw^{i} * py^{n-i}$

From all the formulas above, we can prove that all the parameters have unique solutions. Based on the average arrival rate (*avearrirate*) and average service rate (*aveservrate*), the average inter-arrival time (*avearritime*) and average service time (*aveservtime*) can be expressed as:

$$avearritime = \frac{1}{avearrirate} \ \& \ aveservtime = \frac{1}{aveservrate}$$

(B). variance of inter-arrival time and variance of service time

It is difficult to obtain the variance of inter-arrival time and variance of service time because we cannot derive the probability distribution function (pdf) for inter-arrival time and service time although we can obtain their average values.

In order to estimate the variance of inter-arrival time and variance of service time, we analyze the time interval between two successful transmissions and its probability based on the three events described in section 5.3.1.1:

E_s = {successful transmission}, E_i = {idle channel}, and E_c = {collision}.

We divide the transmission of node 1 into two parts: the arrival part which is the transmission from source to node 1, and the service part which is the transmission from node 1 to the destination (shown in Figure 5.4). We compute the variance of inter-arrival time and variance of service time from these two parts respectively.

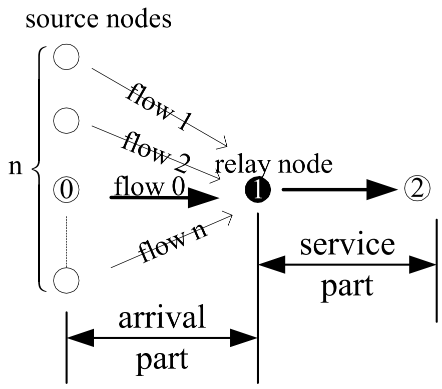


Figure 5.4: Two parts channel model

In an arbitrary time slot, one of three events $E_{s,}$ E_c and E_i must take place. E_s can be divided into two components: successful transmission at arrival part and successful transmission at service part.

The principle of the scheme lies in analyzing all the possible events that can occur between two successful transmissions (refer to the Figure 5.5). Between two successful transmissions, three events may occur: i) channel is idle; ii) collision and iii) successful transmission of the other nodes. Every event can appear infinitely often. We assume that the occurrences of these events are independent. Based on these conditions, two schemes are designed to estimate the variance of inter-arrival time and the variance of service time.

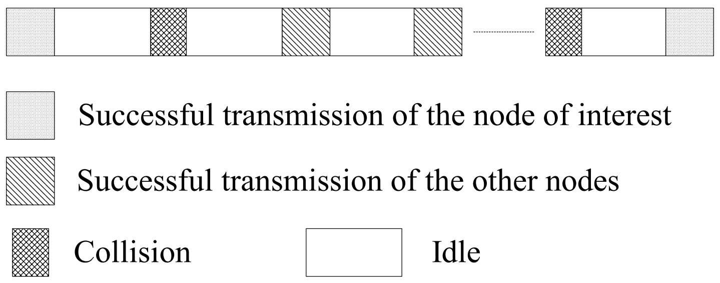

Successful transmission of the node of interest
Successful transmission of the other nodes
Collision
Idle

Figure 5.5: Events in the time slots between two successful transmissions on the node of interest

Before introducing our schemes, we define some parameters, as shown in Table 5.

Table 5： Parameter definitions

p_c	The probability that a collision occurs in a given time slot.
p_i	The probability that channel is idle in a given time slot.
p_{sa}	The probability that a successful transmission occurs in a given time slot in arrival part.
p_{ss}	The probability that a successful transmission occurs in a given time slot on service part.
t_s	Average time of successful transmission.
t_i	Average time of idle slot.
t_c	Average time of collision.
$T1 = i*t_i + j*t_c + k*t_s + t_s$	The inter-arrival time when there are *i* idle time slots, *j* collision time slots and *k* service part's successful transmissions time slots between two successful transmissions of arrival part.
$T2 = i*t_i + j*t_c + t_s$	The inter-arrival time when there are *i* idle time slots and *j* collision time slots between two successful transmissions of arrival part.
$T(1,m)$	$\{T1 \mid$ ***m*** *nodes share channel*$\}$
$T(2,m)$	$\{T2 \mid$ ***m*** *nodes share channel*$\}$
$P1 = p_i^i p_c^j p_{ss}^k p_{sa}$	The probability that the inter-arrival time is $T1$ with the same *i, j, k.*
$P2 = p_i^i p_c^j p_s$	The probability that the inter-arrival time is $T2$ with the same *i, j, k.*
$P(1,m)$	$\{P1 \mid$ ***m*** *nodes share channel*$\}$
$P(2,m)$	$\{P2 \mid$ ***m*** *nodes share channel*$\}$

where has $p_c + p_i + p_{sa} + p_{ss} = 1$.

For the variance of inter-arrival time, we concentrate on the arrival part. In this scenario, the three events that may occur between two successful transmissions are:

i) Channel is idle;

ii) Collision;

iii) Successful transmission on the service part.

All possible cases and the corresponding probabilities of inter-arrival time are shown in Table 6.

Table 6： All possible cases and the corresponding probabilities of inter-arrival time

Cases		Inter-arrival time	Probability
At least one source has packet to send. In *n* sources, *m* (1≤*m*≤*n*) sources have packets in their queues, and (*n-m*) sources do not have packets.	(i) Relay node has packets in its queue	$T(1,m+1)$	$C_n^m * py^m * pw^{n-m} * xx * P(1,m+1)$
	(ii) Relay node does not have packets in its queue	$T(2,m)$	$C_n^m * py^m * pw^{n-m} * (1-xx) * P(2,m)$
All sources have not packets to send. **<u>After that</u>**, *m* (1≤*m*≤*n*) sources have packets in their queues, and (*n-m*) sources do not have packets.	(iii) Relay node has packets in its queue	$T(1,m+1)+\frac{1}{n*\lambda_0}$	$C_n^m * py^m * pw^{n-m} * xx * P(1,m+1)$
	(iv) Relay node does not have packets in its queue	$T(2,m)+\frac{1}{n*\lambda_0}$	$pw^n * C_n^m * py^m * pw^{n-m} * (1-xx) * P(2,m)$

In last two cases, the term “After that” refers to the period after the time interval in which none of the sources have packets to send to the relay node, shown in Figure 5.6.

In case (iii) and (iv) in Table 6, a value $\frac{1}{n*\lambda_0}$ is added to each inter-arrival time. That is because, after $\frac{1}{n*\lambda_0}$ from the time at which all sources have no packet in their queues, at least one source will have packets to send to the relay nodes. $\frac{1}{n*\lambda_0}$ is the average time for generating packets.

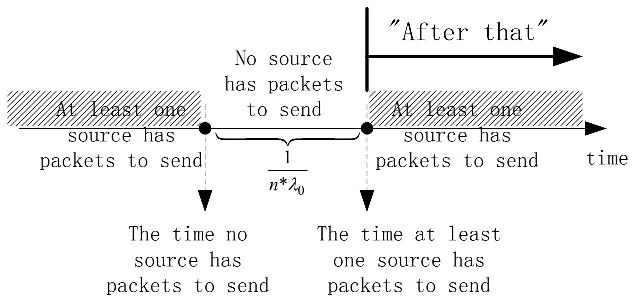

"After that"
No source has packets to send
At least one source has packets to send
At least one source has packets to send
$\frac{1}{n*\lambda_0}$
time
The time no source has packets to send
The time at least one source has packets to send

Figure 5.6：Meaning of term “After that”

Based on the four cases in Table 6, and the definition of variance,

$$\mathrm{var}(x) = E\left(E(x) - x\right)^2$$

we derive the variance of inter-arrival time as shown:

$$\mathrm{var\_int}\,erarrival\_time = \sum_{m=1}^{n}\left[\begin{array}{l}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty} C_n^m * py^m * pw^{n-m} * xx * P(1,m+1) * \left(T(1,m+1) - \frac{1}{\lambda_1}\right)^2 \\ + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} C_n^m * py^m * pw^{n-m} * (1-xx) * P(2,m) * \left(T(2,m) - \frac{1}{\lambda_1}\right)^2\end{array}\right]$$

$$+ \sum_{m=1}^{n} pw^n * \left[\begin{array}{l}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty} C_n^m * py^m * pw^{n-m} * xx * P(1,m+1) * \left(T(1,m+1) + \frac{1}{n*\lambda_0} - \frac{1}{\lambda_1}\right)^2 \\ + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} C_n^m * py^m * pw^{n-m} * (1-xx) * P(2,m) * \left(T(2,m) + \frac{1}{n*\lambda_0} - \frac{1}{\lambda_1}\right)^2\end{array}\right]$$

where $T(1,1)=T(2,1)=ts$, because when there is only one node that has packets to send, it can send immediately because no other nodes are competing for the channel with it. Thus, the inter-arrival time between two packets is *ts*.

For the variance of service time, we concentrate on the service part. In this scenario, three events between two successful transmissions are:

i) Channel is idle;

ii) Collision;

iii) Successful transmission on the arrival part.

Besides the parameters defined for the variance of inter-arrival time, two more parameters are defined for the variance of service time as shown in Table 7.

Table 7： Two parameters' definitions for variance of service time

$PP1 = p_i^i p_c^j p_{sa}^k p_{ss}$	The probability that the service time is *T*1 with the same *i, j, k*.
PP(1,*m*)	{*PP*1 | *m* nodes share channel}

The possible cases of service time are very simple (Table 8).

Table 8： All possible cases and the corresponding probabilities of service time

Cases	Inter-arrival time	Probability
In *n* sources, *m* (*1≤m≤n*) sources have packets in their queues, and (*n-m*) sources do not have packets. Relay node is not taken into consideration	*T*(1,*m*+1)	$C_n^m * py^m * pw^{n-m} * PP(1, m+1)$

$$\text{var}\_service\_time = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty}\sum_{m=0}^{n} C_n^m * py^m * pw^{(n-m)} * PP(1,(m+1)) * (T(1,m+1) - \frac{1}{\mu_1})^2$$

5.3.2.3 Simulations

In simulations, we vary the number of sources, the packet length and the packet generation interval to prove the feasibility of our algorithms.

Figure 5.7 and Figure 5.8 show the comparison between actual queuing delay of two thousand packets randomly chosen from fifty thousand packets in the simulations and the maximum queuing delay obtained by our algorithms for four sources scenario and five sources scenario respectively. The maximum queuing delay obtained by our algorithms is a slightly lower than queuing delay of a small fraction of the packets while larger than most of the packets. That is because that we assume the events are independent which make us underestimate the queuing delay of certain packets.

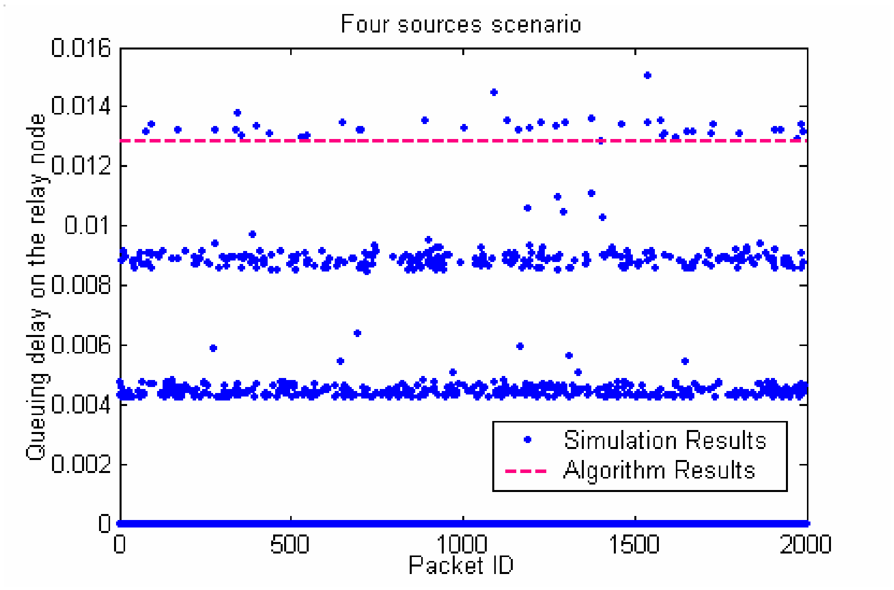


Figure 5.7: Actual queuing delay of 2000 randomly chosen packets and the analyzing maximum queuing delay for four sources scenario

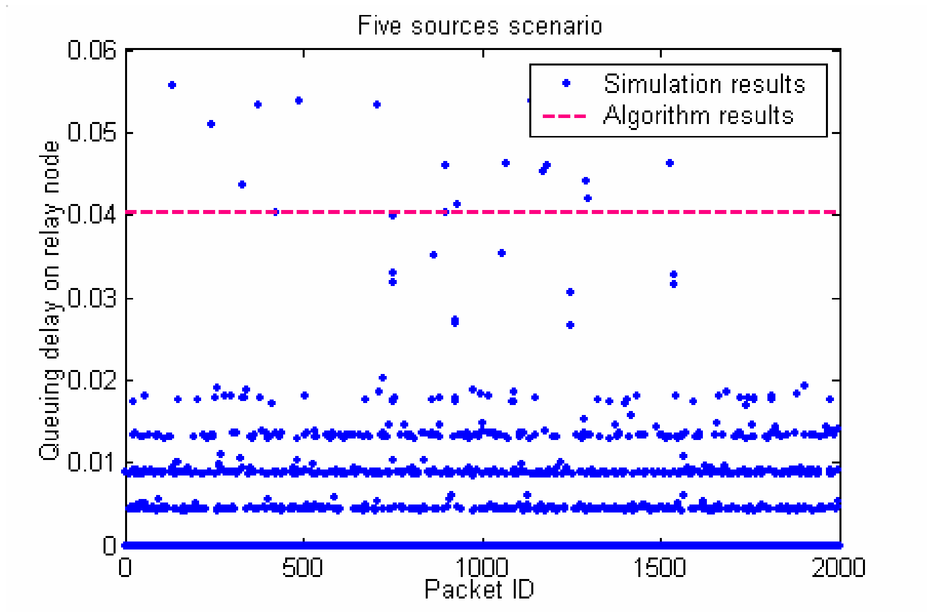


Figure 5.8: Actual queuing delay of 2000 randomly chosen packets and the analyzing maximum queuing delay for five sources scenario

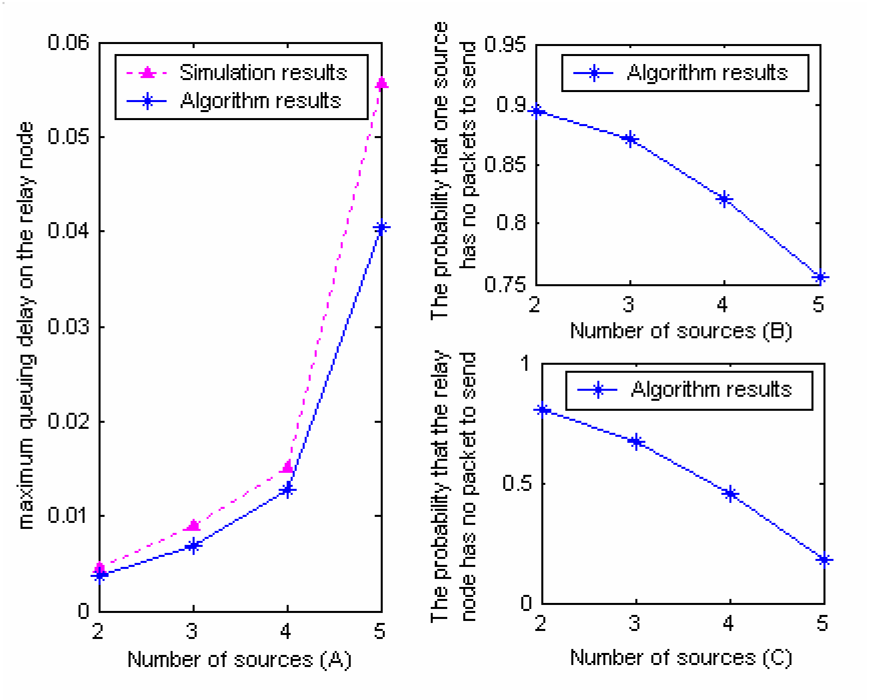


Figure 5.9: (A) Maximum queuing delay on the relay node obtained by simulations and algorithms under different numbers of sources; (B) The probability that a source does not have packets to send; and (C) The probability that the relay node does not have packets to send.

Figure 5.9 shows some results for the scenario with packet length 700 bytes and packet generation interval 0.05s from 2 sources to 5 sources. In Figure 5.9(A), when the number of sources is small, the result obtained by algorithm is close to that of simulation. With the increasing of the number of sources, the gap between the results of simulations and algorithms increases. That is because, when more nodes compete for the channel, the interaction among them has stronger impact on the queuing delay. On the other hand, the independence we assumed make our algorithm become less accurate as number of sources increase as shown by the simulation results. Figure 5.9 (B) and (C) show the probabilities that the source or relay node does not have packets to send. When the length of packets increases, the service time a packet requires increases. Longer time packets staying in the queue results in the decrease of the probability that the queues of nodes are empty.

We change the packet lengths to show the trend of the maximum queuing delay under different packet lengths. The packets generation interval is 0.05s and packet lengths vary from 100 bytes to 900 bytes. The results are shown in Figure 5.10.

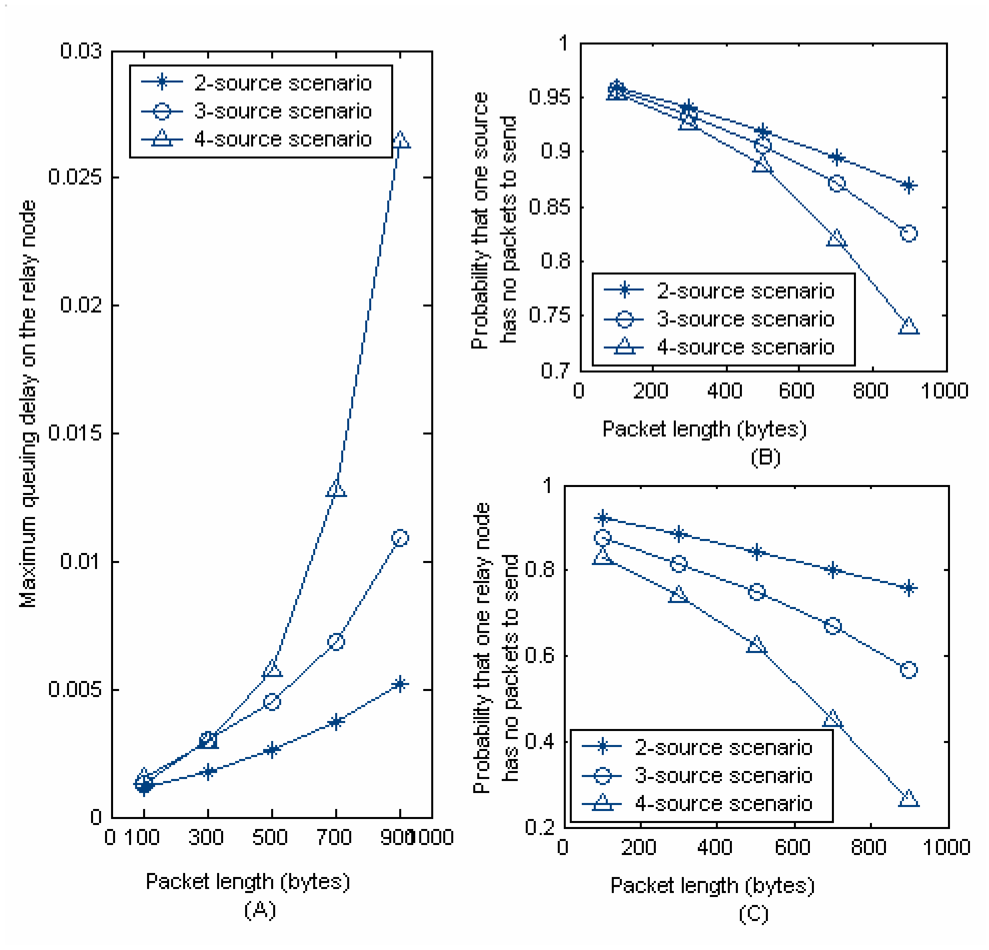


Figure 5.10: (A) Maximum queuing delay on the relay node obtained by algorithms under different packet length and number of sources; (B) The probability that a source does not have packets to send; and (C) The probability that the relay node does not have packets to send.

In Figure 5.10, (A) shows that with the increase of the packet length, the maximum queuing delay of packets on the relay node increases. Furthermore, the increase in maximum queuing delay is greater with more sources. (B) shows that the probability that a source node does not have packets to send decreases with the increase of the packet length. This is because, when the packet length is large, the service time of the packet is long and the time that a packet stays in a queue is long, such that the probability that this queue is empty is small. In addition, with the increase of the number of sources, the probability also decreases. With more sources active, more nodes compete the channel, hence the service time of a packet increases and the probability that the queue is empty decreases. (C) shows that the probability that the relay node does not have packets to send decreases with the increase of the packet length and increase of the number of sources as well. The reason is the same as that of (B).

We also change the packets generation interval to see the trend of the maximum queuing delay under different packets generation intervals. The packet length is 500 bytes and generation interval is set from 0.05s to 0.2s. The results are shown in Figure 5.11.

In Figure 5.11, (A) shows that with the increase of the packet generation interval, the maximum queuing delay of packets on the relay node decreases. Furthermore, with more sources, the faster the maximum queuing delay decreasing. (B) shows that the probability that a source node does not have packets to send increases with the increase of the packet generation interval. That is because, when the packet generation interval is large, the fewer packets are generated. Thus, the probability that this queue is empty is larger. In addition, with the increase of the number of sources, the probability decreases as more nodes compete the channel, such that service time of a packet increases and the probability that the queue is empty decreases. (C) shows that the probability that the relay node does not have packets to send increases with the increase of the packet generation intervals and decreases with the increase of the number of sources. The reason is the same as that of (B).

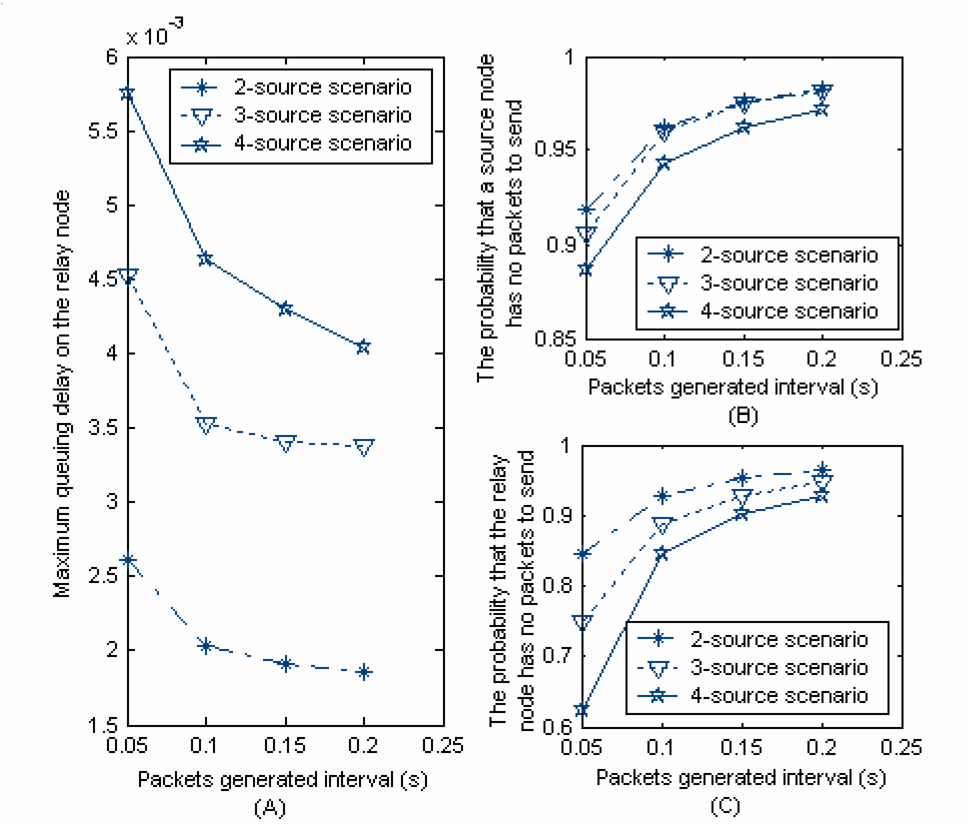


Figure 5.11： (A) Maximum queuing delay on the relay node obtained by algorithms under different packet generation intervals and number of sources; (B) The probability that a source does not have packets to send; and (C) The probability that the relay node does not have packets to send.

In this chapter, we derived the maximum queuing delay at each node in the ad hoc networks adopting IEEE 802.11 as MAC protocol. As we have introduced, the end-to-end delay of a flow is the sum of the hop-by-hop delays of each hop in the flow and each hop-by-hop delay itself is comprised of the queuing delay, service time and transmission time. The queuing delay is the key element in end-to-end delay, based on which we will discuss the calculation of end-to-end delay in next chapter.

Chapter 6 Analysis for End-to-End delay of a Path

6.1 Introduction

In this chapter, we concentrate the analysis of maximum end-to-end delay of flows in a general scenario where all elements are random, namely, random paths and random flows are chosen from random network topology. The analysis process is divided into two parts. We first analyze the maximum queuing delay at each node and then compute the maximum end-to-end delay of a path based on the results of first part.

6.2 Maximum queuing delay analysis

After deriving the maximum queuing delay analysis for the two-hop scenarios, we extend the method to analyze the maximum queuing delay for a more complex scenario – the general scenario. The generic scenario differs from the two-hop scenario in which each node is treated as a unique individual instead of a member of any group. As such, *pw* (the probability that a node has packets to send) and *py* (the probability that a node does not have packets to send) of each node are different from other nodes, unlike in the two-hop scenario where all sources have the same *pw* and *py*. Thus, in the two-hop scenario, equations comprise of components such as pw^n and $(1-pw)^n$ whereas in general scenario, they cannot. In order to meet the requirements in the general scenario, we define some additional parameters as shown in Table 9:

Table 9： The parameter's definitions

$\mathbf{A}(n \times n)$	One-hop adjacency matrix.
$\mathbf{B}(n \times n) = \mathbf{A} \times \mathbf{A}$	Block-neighbor matrix
$\mathbf{NB}(1 \times n)$	Block-neighbors number matrix.

Matrix **B** is defined according to the transmission property, which states that if a node is transmitting, its one-hop neighbors and two-hop neighbors cannot transmit simultaneously due to the collisions [14]. **B** contains all two-hop shortest paths in the network. In the other words, it contains all the one-hop neighbors and two-hop neighbors of each node. We call a node's one-hop and two-hop neighbors as Block-neighbors of this node and **B** as Block-neighbor matrix because it shows all the Block-neighbors of each node. Matrix **NB** sums up the number of the Block-neighbors for each node. In addition, the probability that the corresponding node transmits in a randomly chosen slot time (τ) is an important parameter in the analysis process, which can be obtained by two equations below:

$$\tau = \frac{2(1-2p)}{(1-2p)(CW_{\min}+1) + pCW_{\min}(1-(2p)^m)}$$

$$p = \frac{2CW_{\min}(n-1)}{(CW_{\min}+1)^2 + 2CW_{\min}(n-1)}$$

where $CW_{\min}$ is the minimum contention window in IEEE 802.11, and p is the probability of a collision seen by a packet being transmitted.

Thus, the probability τ of a node is determined by the corresponding n (number of nodes that share one channel). In the general scenario, n is the number of block-neighbors of the node of interest that can be obtained from the matrix **NB**.

As in the previous section, any node in the network, e.g. node *i*, has two probabilities py_i and pw_i. py_i is the probability that node *i* has packets in queue waiting to be delivered, and pw_i is the probability that this node does not have packets in its queue. For node *i*, we also assume the additional parameter definitions as shown in Table 10:

Table 10: The parameters' definitions

n	number of its Blocked-Neighbors, $n = \mathbf{NB}(1,i)$
m	number of its one-hop neighbors which send packets to it
λ_i	average arrival rate of node *i*
μ_i	average service rate of node *i*
td_k	average service time when there have *k* nodes compete channel

6.2.1 Average arrival rate (λ) and average service rate (μ)

If node *i* is a source, the average arrival rate equals to the average rate at which packets are generated, namely $\lambda_i = average\_packet\_generation\_rate$.

The average service rates of the sources can be calculated by analyzing the all possible cases (Table 11):

Table 11: All possible cases for the average service rate of source

Cases	Service rate	Probability
j block-neighbors of the source of interest have packets and (*n-j*) block-neighbors do not have packets.	$\frac{1}{td_{j+1}}$	$C_n^j * \left(\prod_{\substack{all\ nodes \\ have\ packets}}^{j} py_{\substack{selected \\ node}} \right) * \left(\prod_{\substack{all\ nodes\ have \\ no\ packets}}^{n-j} pw_{\substack{selected \\ node}} \right)$

In Table 11, the expression for the probability is a summation of a series of possible cases. In each case, *j* nodes with packets to send are selected. As each node has its own *pw* and *py*, the probability that these *j* nodes have packets to send is given by the product of the corresponding *py* of the selected nodes. For example, the probability that node *x*, node *y* and node *z* have packets to send is $py_x * py_y * py_z$. Thus, for a certain number *j*, there are C_n^j possible scenarios that *j* nodes have packets to deliver.

In conclusion, the average arrival rate (λ_i) and the average service rate (μ_i) of a source can be expressed as:

$$\lambda_i = average\_packet\_generation\_rate$$

$$\mu_i = \sum_{j=0}^{n} C_n^j * \left(\prod_{\substack{all\ nodes \\ have\ packets}}^{j} py_{\substack{selected \\ node}} \right) * \left(\prod_{\substack{all\ nodes\ have \\ no\ packets}}^{n-j} pw_{\substack{selected \\ node}} \right) * \frac{1}{td_{j+1}}$$

The probability that the queue of a node is empty is $pw_i = 1 - \frac{\lambda_i}{\mu_i}$.

If node *i* is not a source, all the packets it has come from its neighbors whose next hop destination is node *i*. Thus, the average arrival rate of node *i* is determined by the number of nodes which have packets to send to it. Therefore, if node *i* wants to receive packets, there must have at least one neighbor with packets to send to it.

$$\lambda_i = \sum_{j=1}^{m} \sum_{k=0}^{(n-m)} \left\{ \begin{array}{l} C_m^j * \left(\prod_{\substack{all\ nodes \\ have\ packets}}^{j} py_{\substack{selected \\ node}} \right) * \left(\prod_{\substack{all\ nodes\ have \\ no\ packets}}^{m-j} pw_{\substack{selected \\ node}} \right) \\ * C_{(n-m)}^k * \left(\prod_{\substack{all\ nodes \\ have\ packets}}^{k} py_{\substack{selected \\ node}} \right) * \left(\prod_{\substack{all\ nodes\ have \\ no\ packets}}^{n-m-k} pw_{\substack{selected \\ node}} \right) \\ * (py_i * \frac{j}{td_{j+k+1}} + pw_i * \frac{j}{td_{j+k}}) \end{array} \right\}$$

where, n is the number of the block-neighbors of node i and m is the number of one-hop neighbors of node i which are sending packets to it.

The average service rate is the same as the analysis and expression when the node of interest is a source, which is

$$\mu_i = \sum_{j=0}^{n} C_n^j * \left(\prod_{\substack{all\ nodes \\ have\ packets}}^{j} py_{\substack{selected \\ node}} \right) * \left(\prod_{\substack{all\ nodes\ have \\ no\ packets}}^{n-j} pw_{\substack{selected \\ node}} \right) * \frac{1}{td_{j+1}}$$

Similarly, $pw_i = 1 - \frac{\lambda_i}{\mu_i}$

From the equations above, we can prove that all the parameters have unique solutions.

6.2.2 Variance of inter-arrival time and variance of service time

We use the same method as that in the two-hop scenario to obtain the variance of arrival time and variance of service time. We analyze the probabilities of all possible events occurring in one time slot to estimate the time interval required to send out a packet successfully. Firstly, we derive the formulas for variance of the inter-arrival time. Some relevant parameters are defined in the Table 12, where $p_c + p_i + p_{sa} + p_{ss} = 1$.

For the variance of inter-arrival time, we concentrate on the arrival part. In this scenario, the three events that may occur between two successful transmissions at the node of interest are:

i) Channel is idle;

ii) Collision;

iii) Successful transmission to other nodes.

Table 12: Definition of parameters used to find variances of inter-arrival time and service time

n	number of its Blocked-Neighbors, $n = \mathbf{NB}(1,i)$
m	number of its one-hop neighbors which send packets to it
p_c	probability that a collision occurs in a given time slot
p_i	probability that channel is idle in a given time slot
p_{sa}	probability that a successful transmission occurs in a given time slot on arrival part
p_{ss}	probability that a successful transmission occurs in a given time slot on service part
t_s	average time of successful transmission
t_i	average time of idle slot
t_c	average time of collision
$p_c(x)$	$\{p_c \mid x$ nodes share channel$\}$
$p_i(x)$	$\{p_i \mid x$ nodes share channel$\}$
$p_{sa}(x,y)$	$\{p_{sa} \mid (x+y)$ nodes share channel; x nodes send packets to the node of interest and y nodes send packets to other nodes$\}$
$p_{ss}(x,y)$	$\{p_{ss} \mid (x+y)$ nodes share channel; x nodes send packets to the node of interest and y nodes send packets to other nodes$\}$
$T1 = ii * t_i + jj * t_c + kk * t_s + t_s$	inter-arrival time when there are *ii* idle time slots, *jj* collision time slots and *kk* service part's successful transmissions time slots between two successful transmissions of arrival part
$T2 = ii * t_i + jj * t_c + t_s$	inter-arrival time when there are *ii* idle time slots and *jj* collision time slots between two successful transmissions of arrival part
$P1 = p_i^{ii} p_c^{jj} p_{ss}^{kk} p_{sa}$	probability that the inter-arrival time is $T1$ with the same *ii, jj, kk*
$P2 = p_i^{ii} p_c^{jj} p_s$	probability that the inter-arrival time is $T2$ with the same *ii, jj*

$P1(x,y)$	$\{P1 \mid (x+y)$ nodes share channel; x nodes send packets to the node of interest and y nodes send packets to other nodes$\}$
$P2(x)$	$\{P2 \mid x$ nodes share channel$\}$

Therefore, all possible cases and the corresponding probabilities of inter-arrival time (using node *i* as an example) are as described in Table 13.

Table 13： All possible cases and the corresponding probabilities of inter-arrival time

Cases		Inter-arrival time	Probability (P)
At least one node has packets to send to node i. Among m nodes whose next-hop destination is node i, j ($1 \le j \le m$) nodes have packets in their queues, and (m-j) nodes do not have.	**(s1)** Among (n-m) nodes whose next-hop destinations are not node i, k $(0 \le k \le (n-m))$ nodes have packets in their queues, and (n-m-k) nodes do not have. Node i has packets in its queue.	*T1*	$C_m^j * \left(\prod^{\substack{j\ \ nodes\\ have\ \ packets}} py_{\substack{selected\\ node}}\right) * \left(\prod^{\substack{(m-j)\ \ nodes\\ have\ \ no\ \ packets}} pw_{\substack{selected\\ node}}\right) * C_{n-m}^k * \left(\prod^{\substack{k\ \ nodes\\ have\ \ packets}} py_{\substack{selected\\ node}}\right) * \left(\prod^{\substack{(n-m-k)\ \ nodes\\ have\ \ no\ \ packets}} pw_{\substack{selected\\ node}}\right) * py_i * P1(j,k+1)$
	(s2) Among (n-m) nodes whose next-hop destinations are not node i, k $(1 \le k \le (n-m))$ nodes have packets in their queues, and (n-m-k) nodes do not have. Node i does not have packets in its queue.	*T1*	$C_m^j * \left(\prod^{\substack{j\ \ nodes\\ have\ \ packets}} py_{\substack{selected\\ node}}\right) * \left(\prod^{\substack{(m-j)\ \ nodes\\ have\ \ no\ \ packets}} pw_{\substack{selected\\ node}}\right) * C_{n-m}^k * \left(\prod^{\substack{k\ \ nodes\\ have\ \ packets}} py_{\substack{selected\\ node}}\right) * \left(\prod^{\substack{(n-m-k)\ \ nodes\\ have\ \ no\ \ packets}} pw_{\substack{selected\\ node}}\right) * pw_i * P1(j,k)$
	(s3) All (n-m) nodes whose next-hop destinations are not node i do not have packets to send. Node i does not have packets in its queue.	*T2*	$C_n^j * \left(\prod^{\substack{j\ \ nodes\\ have\ \ packets}} py_{\substack{selected\\ node}}\right) * \left(\prod^{\substack{(n-j)\ \ nodes\\ have\ \ no\ \ packets}} pw_{\substack{selected\\ node}}\right) * \left(\prod^{\substack{all\ \ (n-m)\ \ nodes\\ have\ \ no\ \ packets}} pw_{\substack{selected\\ node}}\right) * pw_i * P2(j)$

No nodes have packets to send to node *i*. **<u>After that</u>**, among *m* nodes whose next-hop destination is node *i*, *j* ($1 \le j \le m$) nodes have packets in their queues, and (*m-j*) nodes do not have.	**(*s*4)** Among (*n-m*) nodes whose next-hop destinations are not node *i*, *k* $(0 \le k \le (n-m))$ nodes have packets in their queues, and (*n-m-k*) nodes do not have. Node *i* has packets in its queue.	$T1+\left(\sum_{n}(1/\lambda)\right)/n$	$\left(\overset{\substack{all\ m\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * C_n^j * \left(\overset{\substack{j\ nodes\\ have\ packets}}{\prod} py_{\substack{selected\\ node}}\right) * \left(\overset{\substack{(n-j)\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * C_{n-m}^k * \left(\overset{\substack{k\ nodes\\ have\ packets}}{\prod} py_{\substack{selected\\ node}}\right) * \left(\overset{\substack{(n-m-k)\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * py_i * P1(j,k+1)$
	(*s*5) Among (*n-m*) nodes whose next-hop destinations are not node *i*, *k* $(1 \le k \le (n-m))$ nodes have packets in their queues, and (*n-m-k*) nodes do not have. Node *i* does not have packets in its queue.	$T1+\left(\sum_{n}(1/\lambda)\right)/n$	$\left(\overset{\substack{all\ m\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * C_m^j * \left(\overset{\substack{j\ nodes\\ have\ packets}}{\prod} py_{\substack{selected\\ node}}\right) * \left(\overset{\substack{(m-j)\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * C_{n-m}^k * \left(\overset{\substack{k\ nodes\\ have\ packets}}{\prod} py_{\substack{selected\\ node}}\right) * \left(\overset{\substack{(n-m-k)\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * pw_i * P1(j,k)$
	(*s*6) All (*n-m*) nodes whose next-hop destinations are not node *i* do not have packets to send. Node *i* does not have packets in its queue.	$T2+\left(\sum_{n}(1/\lambda)\right)/n$	$\left(\overset{\substack{all\ m\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * C_n^j * \left(\overset{\substack{j\ nodes\\ have\ packets}}{\prod} py_{\substack{selected\\ node}}\right) * \left(\overset{\substack{(n-j)\ nodes\\ have\ no\ packets}}{\prod} pw_{\substack{selected\\ node}}\right) * \left(\overset{\substack{all\ (n-m)\ nodes\\ have\ no\ packets}}{\prod} pw\right) * pw_i * P2(j)$

The phrase "After that" has the same meaning as that in Chapter 5, i.e. after the interval that all nodes whose next-hop destinations are node of interest do not have packets to send.

In cases (*s*4), (*s*5) and (*s*6), a value $\left(\left(\sum_n (1/\lambda)\right)/n\right)$ is added to each inter-arrival time. This is because, after $\left(\left(\sum_n (1/\lambda)\right)/n\right)$ from the time at which all sources have no packet in their queues, at least one source will have packets to send to the relay nodes. The term $\left(\left(\sum_n (1/\lambda)\right)/n\right)$ is the average time needed by the nodes whose next-hop destination is the node of interest, to get packets.

Based on the six instances in Table 6, and the definition of variance,

$$var(x) = E\big(E(x) - x\big)^2$$

the variance of inter-arrival time can be derived as follows:

$$\begin{aligned}
\text{var\_int}er - arrival\_time = & \sum_{j=1}^{m}\sum_{k=0}^{n-m}\left[\sum_{ii=0}^{\infty}\sum_{jj=0}^{\infty}\sum_{kk=0}^{\infty}\left(P_{s1} * \left(T1 - \frac{1}{\lambda_i}\right)^2 + P_{s4} * \left(T1 + \frac{\sum_n (1/\lambda)}{n} - \frac{1}{\lambda_i}\right)^2\right)\right] \\
& + \sum_{j=1}^{m}\sum_{k=1}^{n-m}\left[\sum_{ii=0}^{\infty}\sum_{jj=0}^{\infty}\sum_{kk=0}^{\infty}\left(P_{s2} * \left(T1 - \frac{1}{\lambda_i}\right)^2 + P_{s5} * \left(T1 + \frac{\sum_n (1/\lambda)}{n} - \frac{1}{\lambda_i}\right)^2\right)\right] \\
& + \sum_{j=1}^{m}\left[\sum_{ii=0}^{\infty}\sum_{jj=0}^{\infty}\left(P_{s3} * \left(T2 - \frac{1}{\lambda_i}\right)^2 + P_{s6} * \left(T2 + \frac{\sum_n (1/\lambda)}{n} - \frac{1}{\lambda_i}\right)^2\right)\right]
\end{aligned}$$

where P_{s1}, P_{s2} etc. are the probabilities of corresponding cases.

For the variance of service time, we concentrate on the service part. In this scenario, the three events between two successful transmissions are:

i) Channel is idle;

ii) Collision;

iii) Successful transmission of other nodes.

Besides the parameters defined for the variance of inter-arrival time, two more parameters are defined for the variance of service time in Table 14.

Table 14： Two parameters' definitions for variance of service time

$PP1 = p_i^{ii} p_c^{jj} p_{sa}^{kk} p_{ss}$	probability that the service time is $T1$ with the same *ii, jj, kk*
$PP1(x,y)$	$\{PP1 \mid (x+y)$ nodes share channel; x nodes send packets to the node of interest and y nodes send packets to other nodes$\}$

The possible cases of service time are very simple (Table 15).

Table 15： All possible cases and the corresponding probabilities of service time

Cases	Inter-arrival time	Probability
Among *n* block-neighbors of node *i*, *j* (1≤*j*≤*n*) nodes have packets to send, and (*n-j*) nodes do not have packets. Node *i* is not taken into consideration.	$T1$	$C_n^j * (\prod^{j} py) * (\prod^{n-j} pw) * PP1(j,1)$
All *n* block-neighbors of node *i* do not have packets to send. Node *i* is not taken into consideration.	ts	$\prod^{n} pw$

Thus, the variance of service time can be expressed as the following:

$$\text{var}\_service\_time = \sum_{j=1}^{n} C_n^j * (\prod^{j} py) * (\prod^{n-j} pw) * \sum_{ii=0}^{\infty}\sum_{jj=0}^{\infty}\sum_{kk=0}^{\infty}\left[PP1(j,1) * \left(T1 - \frac{1}{\mu_i} \right)^2 \right]$$
$$+ \prod^{n} pw * (ts - \frac{1}{\mu_i})^2$$

From the formulas derived for variance of inter-arrival time and variance of service time, we can see that the key to solve these formulas is the value of

$P1(x,y),\ PP1(x,y)\ \textit{and}\ P2(x)$. These three parameters can be obtained from the following formulas.

$$P1(x,y) = \left[p_i(x+y)\right]^{ii} * \left[p_c(x+y)\right]^{jj} * \left[p_{ss}(x,y)\right]^{kk} * \left[p_{sa}(x,y)\right]$$

$$PP1(x,y) = \left[p_i(x+y)\right]^{ii} * \left[p_c(x+y)\right]^{jj} * \left[p_{sa}(x,y)\right]^{kk} * \left[p_{ss}(x,y)\right]$$

$$P2(x) = \left[p_i(x)\right]^{ii} * \left[p_c(x)\right]^{jj} * \left[p_s(x)\right]$$

Thus, we turn to the problem of finding the conditional channel probabilities, represented here by $p_s(x)$, $p_i(x)$ and $p_c(x)$. For this purpose, let $P_{tr}(x)$ be the probability that there is at least one transmission in the considered time slot when *x* nodes share the channel. Since the probability that a station transmits in a randomly chosen slot time is τ, we have

$$P_{tr}(x) = 1 - \prod_{k=1}^{x}(1-\tau_k)$$

The probability $P_{suc}(x)$ that a successful transmission occurs on the channel is given by the probability that exactly one node transmits in the channel, conditioned on the fact that at least one node transmits, i.e.

$$P_{suc}(x) = \frac{C_x^1 * \tau * \prod^{x-1}(1-\tau)}{P_{tr}(x)}$$

Therefore, the probability that a successful transmission occurs in a given time slot is $p_s(x) = P_{tr}(x) * P_{suc}(x)$.

Accordingly, $p_i(x) = 1 - P_{tr}(x)$ and $p_c(x) = P_{tr}(x) * (1 - P_{suc}(x))$.

Assuming (*x*+*y*) nodes share one channel and among them, *x* nodes' next hop destination is the node of interest and *y* nodes' next hop destinations are other nodes. The probability that a successful transmission occurs in a given time slot ($p_s(x+y)$) in a certain channel consists of two components: the successful transmission to the node of interest ($p_{sa}(x,y)$) and the successful transmission to the other nodes ($p_{ss}(x,y)$). In other words, $p_{sa}(x,y)+p_{ss}(x,y)=p_s(x+y)$. The values of these two equations can be found from the following two equations:

$$p_{sa}(x,y)=C_x^1 * \tau * \prod^{(x-1)}(1-\tau) * \prod^{y}(1-\tau)$$

$$p_{ss}(x,y)=C_y^1 * \tau * \prod^{(y-1)}(1-\tau) * \prod^{x}(1-\tau)$$

Based on the parameters we have analyzed above, the maximum queuing delay of node *i* can be obtained by the following equation:

$$W_{qi}=\frac{\lambda_i(\mathrm{var}\,arritime+\mathrm{var}\,servtime)}{2(1-\frac{\lambda_i}{\mu_i})}$$

6.3 General expression for the end-to-end delay of a path

The end-to-end delay of a path is the sum of all the hop-by-hop delays of each hop in this path. Therefore, the basic expression of the end-to-end delay of a path is:

$$\begin{aligned}end-to-end\quad delay &= \sum_{i=1}^{\substack{hop\ count\\ of\ the\ path}}\left(hop-by-hop\quad delay\right)_i\\ &= \sum_{i=1}^{\substack{hop\ count\\ of\ the\ path}}\left[\left(queuing\quad delay\right)_i+\left(service\quad time\right)_i\right]\end{aligned}$$

where, *service time* is the sum of *service time* and *transmission time* mentioned in Chapter 5.

We derive each part of the expression for the end-to-end delay to obtain the general expression for random scenario. Firstly, we define some parameters required in the analysis as shown in Table 16.

Table 16： Parameters' definitions

$D_{i,n}$	hop-by-hop delay of node *i* which has *n* neighbors sharing the same channel
$Wq_{i,n}$	queuing delay of packet at node *i* which has *n* neighbors sharing the same channel
$Ts_{i,n}$	service time of a packet at node *i* which has *n* neighbors sharing the same channel

In Table 16, we have $D_{i,n} = Wq_{i,n} + Ts_{i,n}$.

By referring to an arbitrary node *A*, the possible numbers of neighbors which share the channel with *A* are 1, 2, 3…… n_A .

Assume P_{Ak} is the probability that node *A* has *k* active neighbors. (We call the neighbors which share the channel with node *A* as active neighbors).

The hop-by-hop delay at node *A* can be expressed as:

$$D_A = \begin{cases} Ts & A \ \ is \ \ the \ \ source \\ \sum_{i=1}^{n_A} P_{A,i} * D_{A,i} & A \ \ is \ \ the \ \ relay-node \\ 0 & A \ \ is \ \ the \ \ destination \end{cases}$$

For a path with length *L* hops, there are a source, a destination and *L*-1 relay nodes.

Build an *L*-dimension matrix $\mathbf{M}(n_1, n_2, ..., n_L)$, where n_j denotes the maximum possible number of active neighbors of node *j*. Each element $\mathbf{M}(i_1, i_2, ..., i_L)$ in the matrix denotes the probability that the nodes on the path from the source to the last relay node, have the corresponding number of active neighbors $i_1, i_2, ..., i_L$. The end-to-end delay of the path can be expressed as follows:

$$D_L = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} .... \sum_{i_L=1}^{n_L} \left[M(i_1, i_2, ..., i_L) * \sum_{j=1}^{L} D_j \right]$$

6.4 Simulations

In the simulation study, we use a string topology to validate our scheme for the end-to-end delay estimation (Figure 6.1). Each node has two neighbors except the two end nodes which have only one neighbor. However, each node has varying number of block-neighbors. The nodes in the middle have more block-neighbors than the nodes on the two ends.

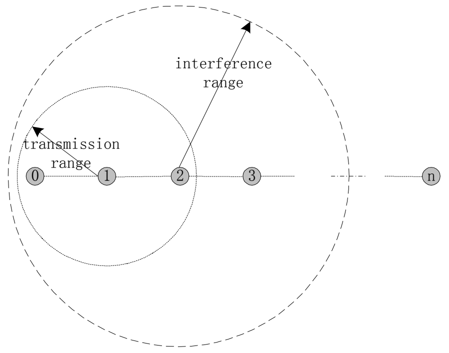


Figure 6.1： String topology used in simulations

In the simulations, we vary the length of the path from 3 hops to 8 hops. The traffic load on the path is 110kbps (packet length 700 bytes; packet generation interval: 0.05s).

Figure 6.2 presents the comparison of the end-to-end delay for paths of different hops obtained from our analysis and the simulations. Though two sets of results are very close to each other, the analytical results are slightly higher than that of the simulations, because, we only consider the neighbors and block-neighbors of a node when we compute the queuing delay of this node. Although we use number of block-neighbors of each node to compute the transmission probability (τ) of a node, we overestimate it since neighbors of each nodes are affected by their own neighbors. Therefore, we overestimate the channel competition situations which results in higher estimated queuing delay, so does the end-to-end delay. Figure 6.3 shows the analytical maximum queuing delay of each node on the strings.

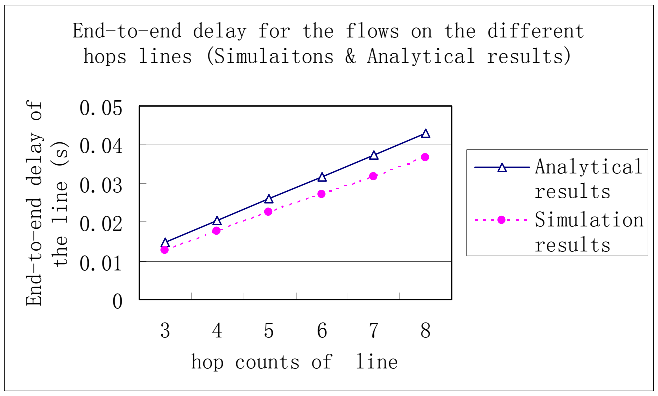


Figure 6.2: The end-to-end delay for flows (110kbps) on the different hops strings

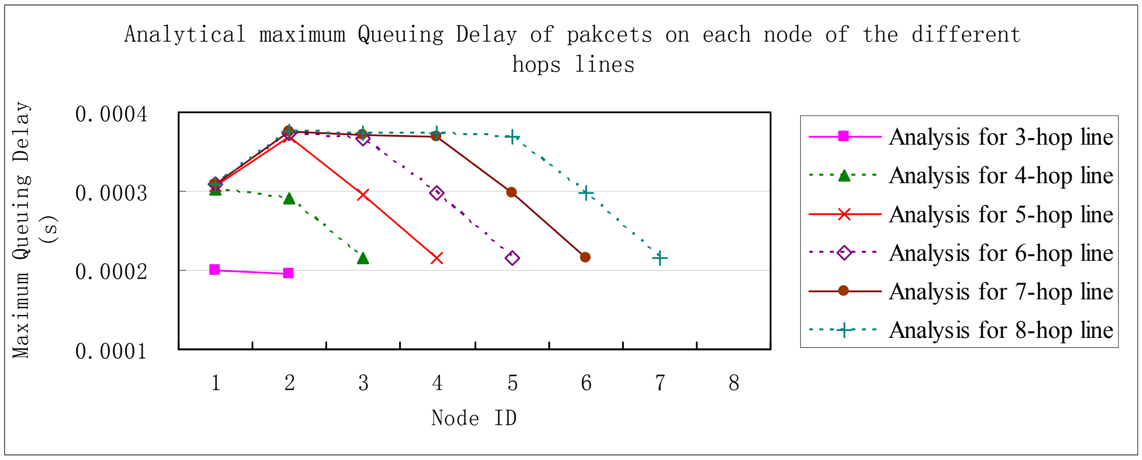


Figure 6.3: Analytical maximum queuing delay of packets on each node of the different hops strings (traffic load: 110kbps)

Figure 6.3 shows that curves for the paths with more than 3 hops increase first, keep almost unchanged, and then decrease. This is because the number of block-neighbors of the nodes at the fore-end of the path increases first, keeps unchanged for several mid nodes, and then decreases at the other end. However, the queuing delay for the node at the end of the path is smaller than the source because the average arrive rate of the node at the end of the path is smaller than that of the source.

Figure 6.4 to Figure 6.7 present the curves of the two parameters, average arrival rate and average service rate, obtained by our algorithm for each node on the string with 5 hops, 6 hops, 7 hops and 8 hops respectively. These four figures show that the two parameters for the nodes in the middle of path are smaller than those of nodes at the ends of path. This is because the numbers of interference nodes for the nodes in the middle are larger than those nodes near the ends. The figures also show that the curves are symmetrical, because the nodes, which have similar average service rate or similar average arrival rate, have the same numbers of the interference nodes.

We change the traffic load to 182kbps (packet length: 700 bytes; packet generation interval: 0.03s). The analytical end-to-end delay of the flows on different hops paths is the same as that of simulations as shown in Figure 6.8. Figure 6.9, which is the analytical maximum queuing delay of packets on each node on the string, has the same trend as that in Figure 6.3.

Figure 6.2 and Figure 6.8 show that our scheme can approximatively derive the maximum end-to-end delay of flows running on the string topologies. Due to the overestimated maximum queuing delay, the analytical maximum end-to-end delay of flows is larger than that in simulations.

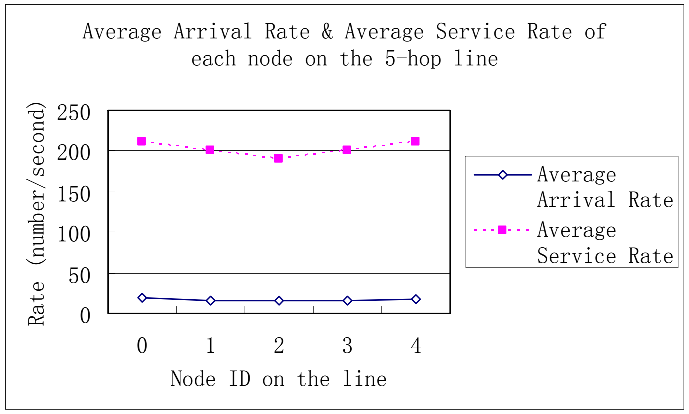


Figure 6.4: Average arrival rate and average service rate of each node on the 5-hop string (traffic load: 110kbps)

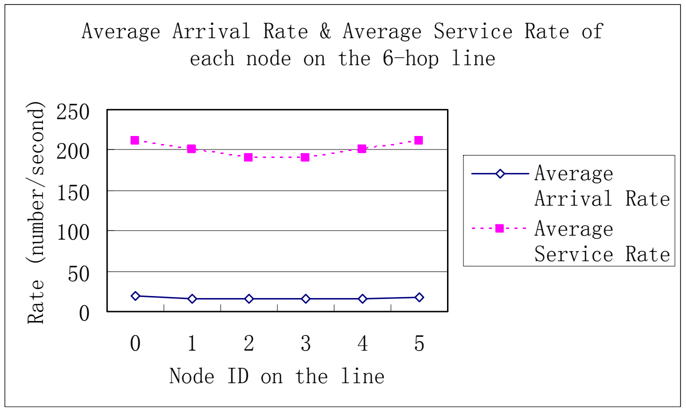


Figure 6.5: Average arrival rate and average service rate of each node on the 6-hop string (traffic load: 110kbps)

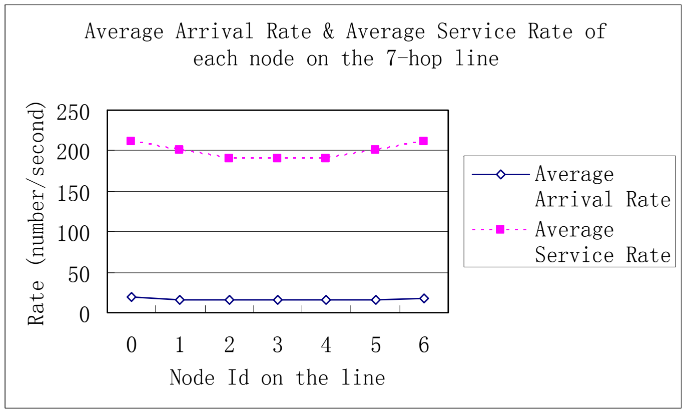


Figure 6.6: Average arrival rate and average service rate of each node on the 7-hop string (traffic load: 110kbps)

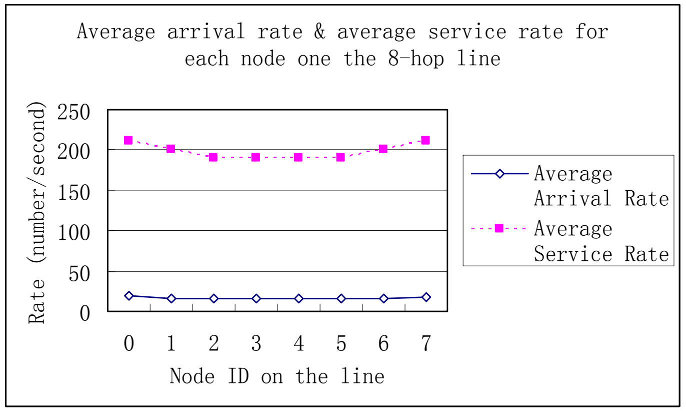


Figure 6.7: Average arrival rate and average service rate of each node on the 8-hop string (traffic load: 110kbps)

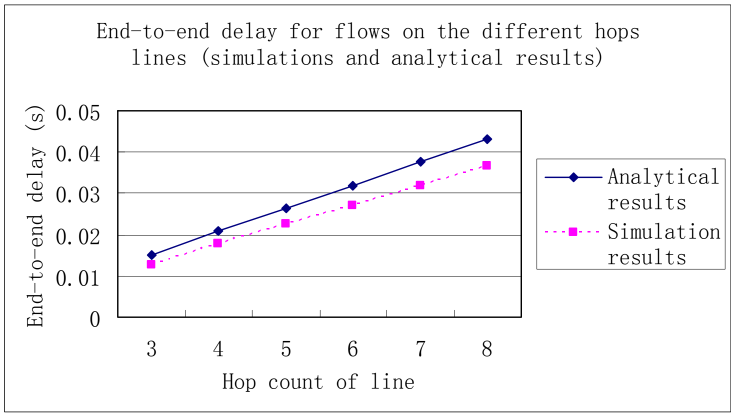


Figure 6.8： End-to-end delay for flows (182kbps) on the different hops strings

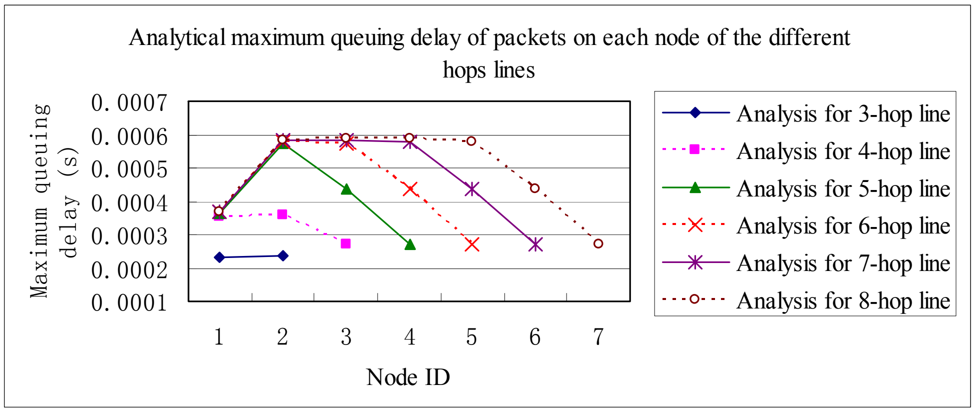


Figure 6.9： Analytical maximum queuing delay of packets on each node of the different hops strings (traffic load: 182kbps)

Furthermore, we extend the simulation scenarios to more general topologies besides string topologies. We add one additional node that acts as source and put two flows into system as shown in Figure 6.10. Two flows have the same destination. Except for the two source nodes, other nodes in this topology need to relay the packets for both flows. In first sets of simulations, we set bit-rate of each flow as 55kbps (packets length: 700 bytes; packets generation interval: 0.1s). The end-to-end delay of two flows obtained by simulations and analysis are shown in Figure 6.11 and Figure 6.12 respectively.

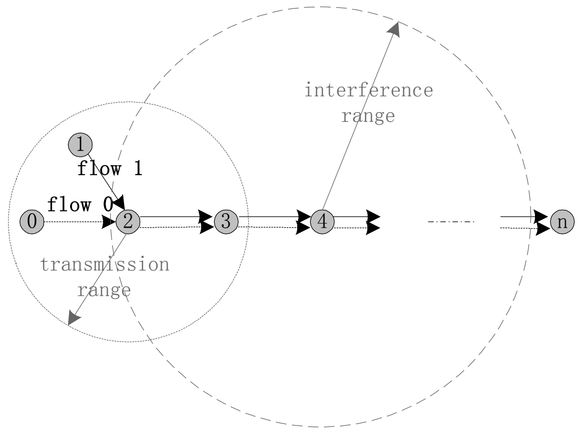


Figure 6.10：Simulation topology (two flows in the system)

We change the bit-rate of two flows to 78kbps (packets length: 700 bytes; packets generation interval: 0.07s). The simulations and analytical results are shown in Figure 6.13 and Figure 6.14 respectively.

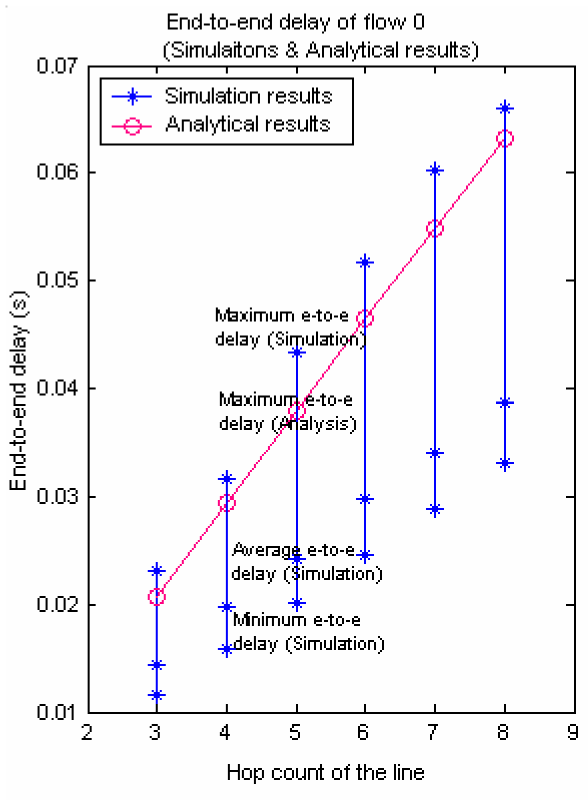


Figure 6.11：End-to-end delay of flow 0 (55kbps)

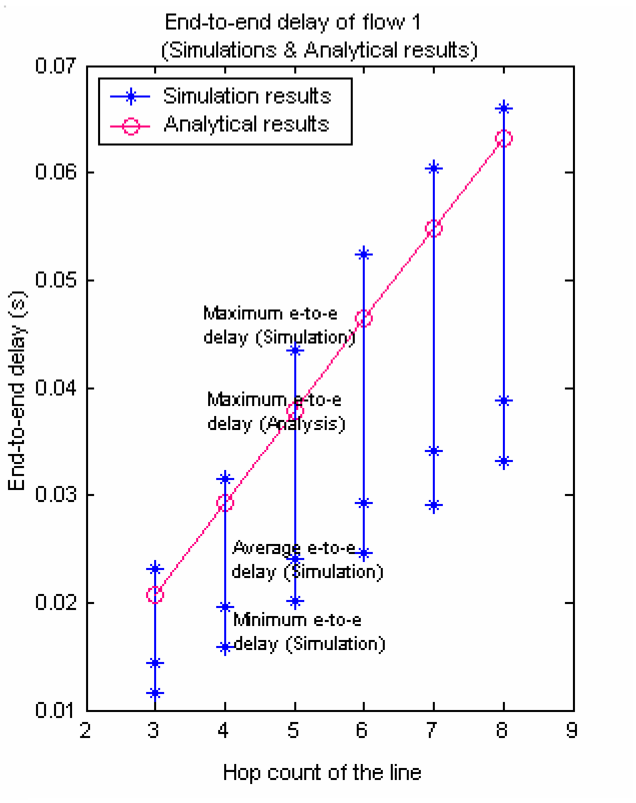
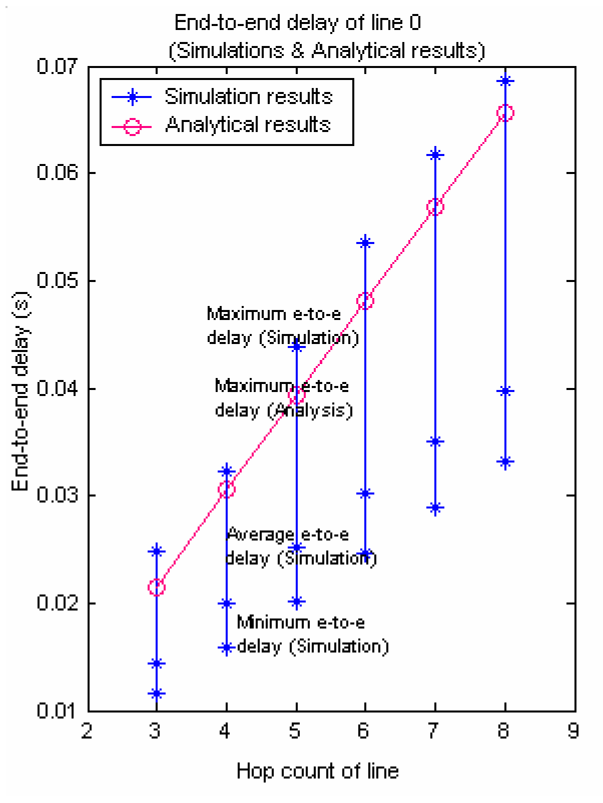


Figure 6.12：End-to-end delay of flow 1 (55kbps)



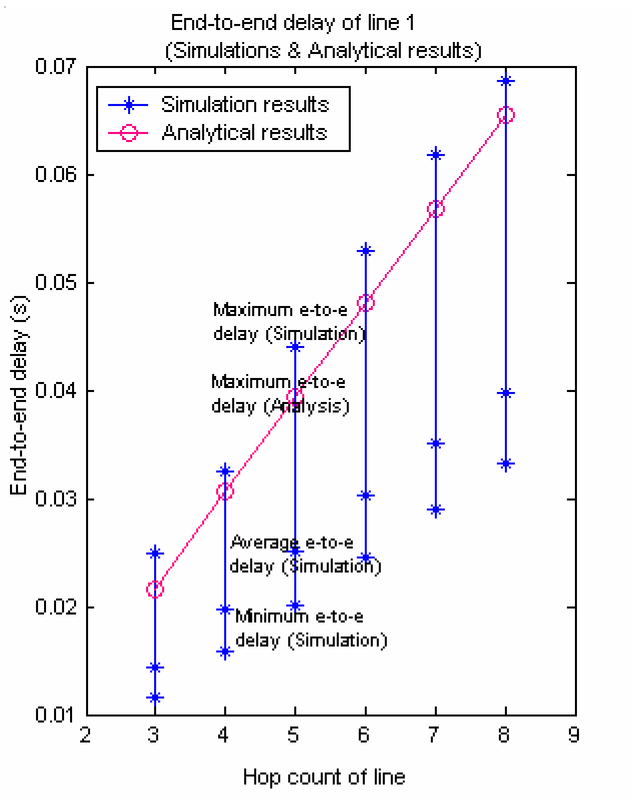


Figure 6.13：End-to-end delay of flow 0 (78kbps)

Figure 6.14：End-to-end delay of flow 1 (78kbps)

Figure 6.11, 6.12, 6.13 and 6.14 show that in the same scenario, the end-to-end delay of flow 0 and flow 1 is similar to each other. That is reasonable since both flows are in the same situations. In these figures, the curves of analytical maximum end-to-end delay of the flows are below that of simulations. This is because we use maximum queuing delay adding average service time to estimate the maximum end-to-end delay. However, the number of the packets whose actual end-to-end delay is larger than analytical results is a very small part of more than ten thousands packets (<2%), which can be seen by the values of average end-to-end delay of the flows and minimum end-to-end delay of the flows shown in the figures. Hence, the analytical end-to-end delay is acceptable.

Chapter 7 Lower Bound of Network Capacity

7.1 Introduction

In Chapter 4, we derive the upper bound of network capacity by ignoring the queuing delay of packets transmission at each node. Chapter 5 and Chapter 6 propose the estimation method of maximum end-to-end delay of flows. Hence, in this chapter, we derive the lower bound of network capacity based on the algorithms in Chapter 4 and the end-to-end delay analysis in Chapter 6. We continue using the number of one-hop sessions as the capacity metric. Lower bound of network capacity is defined as the minimum number of one-hop sessions that can transmit in the network simultaneously subjecting to particular end-to-end delay constraints. We assume each one-hop session belongs to only one flow, and all the flows, whose hop count equals to or is less than the maximum allowed hop counts (which is decided by the end-to-end delay constraints) and contain these one-hop sessions, meet the predefined end-to-end delay constraint. At the same time, in this topology and mobility situation, if one more flow is added in the network, the end-to-end delay constraint will be violated.

7.2 Algorithms description

The fundamental idea of our method used to obtain the lower bound of network capacity is that: according to the predefined end-to-end delay constraints and the analytical formulas derived in Chapter 6, calculate the maximum hop count of a flow that monopolizes the channel subjecting to the end-to-end delay constraint. Compute maximum end-to-end delay for the single flow in string scenarios from one hop to maximum allowed hops. Count all the paths whose hop counts equal to or less than maximum allowed hop count from the information included by multi-hop adjacency matrix of the network. Compute the average delay a packet in the network needed to reach the destination. According to the computed end-to-end delay for the flows with different hops, judge average hop count a packet needs to go through before it reaches the destination. Change this hop count to the number of flows sharing a one-hop link that is feasible in our scenarios and is proved behind. If we know the minimum number of one-hop links that can transmit simultaneously in the network without considering how many flows share it, we can obtain the minimum number of one-hop sessions that can transmit simultaneously by multiplying it with the number of .flows sharing a one-hop link. According to the process described above, two algorithms are required to obtain two important parameters respectively: (i) the number of single one-hop links and (ii) the number of the flows sharing a one-hop link.

7.2.1 Minimum same-hop Links Select Algorithm (MLSA)

In this section, an algorithm, named Minimum same-hop Links Select Algorithm (MLSA), is designed to compute the minimum number of same-hop links that can transmit simultaneously in the network without taking the end-to-end delay constraints into consideration. Figure 7.1 presents the detail of MLSA.

This algorithm is very similar to the Matrix Select-Delete Algorithm (MSDA) introduced in Chapter 4 except for the selection methods of source and destination.

We use the same network topologies example as that in MSDA simulations to calculate the results of MLSA. In the simulations, we compare the results obtained by

MLSA with those by brute-force search algorithm. The results are shown in the Figure 7.3 and Figure 7.5.

```
Begin [Minimum same-hop Links Select Algorithm (MLSA)]
    Input number of nodes (n) and one-hop adjacency matrix ($\mathbf{A}(n,n)$)
    Input link hop count of interest (k)
    Compute $\mathbf{B}(n,n) = \left[ \underbrace{\mathbf{A}(n,n) \times \mathbf{A}(n,n) \times \cdots\cdots \times \mathbf{A}(n,n)}_{k} \right]^*$
    Store all the paths in PathSet;
    SelectedPaths := ∅;
    While (PathSet <> ∅)
    {   source := select the node with the most one-hop neighbors;
        dest := select the node, which is one of k-hop neighbors of the source, and
             and has the most one-hop neighbors;
        AddPath(SelectedPaths, source, dest): Add path originating from source to
            dest, to SelectedPaths;
        B(n,n) := delete all the columns and rows of the source destination, relay
            nodes and their one-hop neighbors;
        PathSet := delete all corresponding paths according to the deletion of B(n,n)
            from PathSet;
    }
    Output (SelectedPaths);
End
```

Figure 7.1: Minimum same-hop Links Select Algorithm (MLSA)

Figure 7.3 and Figure 7.5 show that our algorithm can obtain results approximate to those of the brute-force search algorithm with much less time complexity. The results also allow us to use MLSA to calculate the minimum number of links the network can support simultaneously.

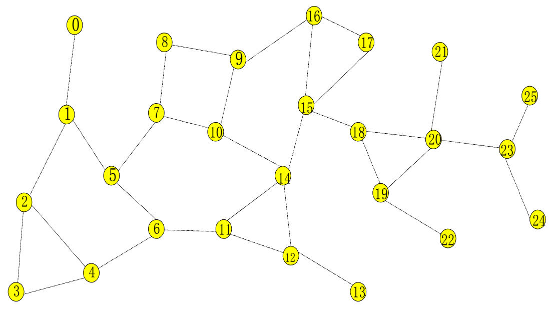


Figure 7.2：Simulation Topology (I)

(26 nodes)

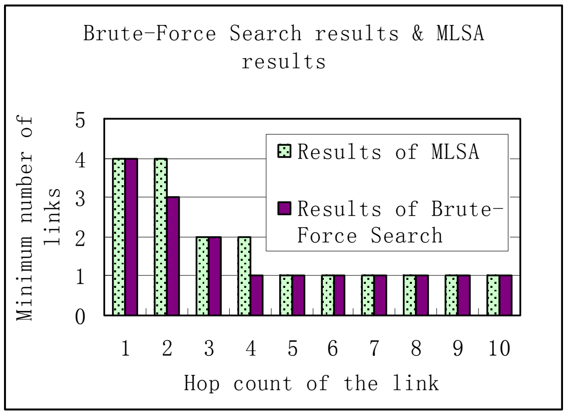


Figure 7.3 : Minimum number of links the network can support simultaneously for simulation topology (I)

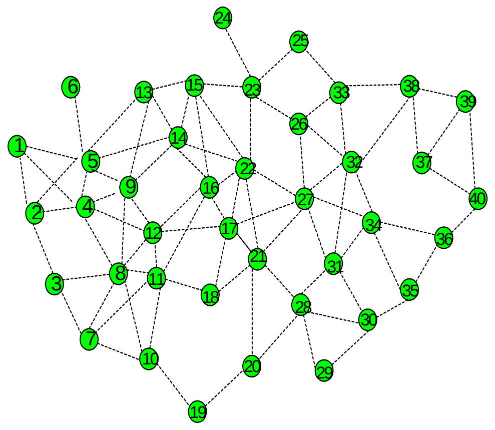


Figure 7.4：Simulation Topology (II)

(40 nodes)

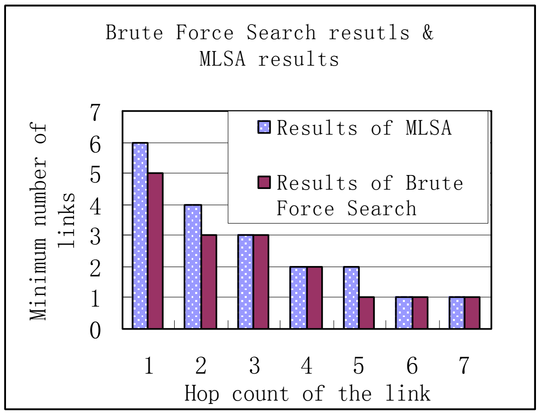


Figure 7.5 : Minimum number of links the network can support simultaneously for simulation topology (II)

7.2.2 Minimum one-hop Session Capacity Algorithm (MSCA)

Based on the results obtained from the MLSA, we propose Minimum one-hop Session Capacity Algorithm (MSCA) to obtain the lower bound of the network capacity that is the minimum one-hop session the network can support simultaneously with end-to-end delay constraints, as shown in Figure 7.6.

```
Begin [Minimum one-hop Session Capacity Algorithm (MSCA)]
{
(1)  Input number of nodes ($n$) in the network;
(2)  Input one-hop adjacency matrix $\mathbf{A}(n,n)$;
(3)  Input end-to-end delay constraint ($TC$);
(4)  Calculate the maximum hop count ($k$) that a flow can go through with the
(5)          maximum end-to-end delay less than delay bound;
(6)  Calculate the maximum delay for a flow on a path with hop count from 1 to $k$,
(7)          respectively and store them in array $Mdelay[k]$;
(8)  Calculate the $k$-hop adjacency matrix $\mathbf{B}(n,n) = \left[ \underbrace{\mathbf{A}(n,n) \times \mathbf{A}(n,n) \times \cdots\cdots \times \mathbf{A}(n,n)}_{k} \right]^*$;
(9)  Calculate the number of paths with the same hop count from $\mathbf{B}(n,n)$ and store
(10)         them in array $Numpath[k]$;
(11) Calculate the average end-to-end delay in the network that a packet requires to be
(12)         sent from source to the destination:
             $ave\_ete\_delay = \frac{\sum_{i=0}^{k-1} (Mdelay[i] * Numpath[i])}{\sum_{i=0}^{k-1} Numpath[i]}$;
(13) For all items in the array $Mdelay[]$
     {
(14)     if (( $ave\_ete\_delay < Mdelay[i]$ ) && ( $ave\_ete\_delay > Mdelay[i+1]$ ))
(15)         $num := i + 1$;
     }
(16) $count$ := number of one-hop links calculated by algorithm MLSA;
(17) $N_{one-hop-sessions} = count * num$;
}
End
```

Figure 7.6: Minimum one-hop Session Capacity Algorithm (MSCA)

In MSCA, line (4) and line (5) can be carried out according to the analysis process and formulas in Chapter 6. Line (13), (14) and (15) compute the average hop count a packet experiences before reaching the destination. We convert this average hop count to the number of packets that are transmitted one hop on the same link because the end-to-end delay a packet experiences in the n hops is not less than that n packets are sent one hop on the same one-hop link. Figure 7.7 illustrates the relationship for a two-hop scenario. The validity of this conversion is proven in Figure 7.8.

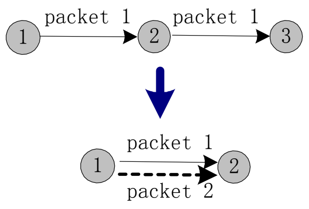


Figure 7.7：Conversional process from a packet runs two hop to two packets run one hop on the same link

Proof (using the scenario in Figure 7.7 as an example):

When packet 1 runs two hops, from node 1 to node 3, if we count from the time that packet 1 is on the header of the queue of node 1 and node 1 is using the channel, the end-to-end delay of it can be expressed as:

$$\begin{aligned} delay1 = {} & transmission\_time\_on\_first\_hop \\ & + channel\_competetion\_time\_at\_node2 \\ & + transmission\_time\_on\_\sec ond\_hop \end{aligned}$$

Correspondingly, the total delay needed for packet 1 and packet 2 running from node 1 to node 2 can be expressed as:

$$\begin{aligned} delay2 = {} & transmission\_time\_for\_packet1 \\ & + transmission\_time\_for\_packet2 \end{aligned}$$

Since $(channel\_competition\_time \geq 0)$, we have

$$delay2 \leq delay1 .$$

If using the end-to-end delay of a packet transmitting *n* hop to denote the total delay of *n* packets transmitting one hop from the same node, the total delay of the latter is overestimated. This guarantees our results will be the lower bound.

Figure 7.8：Proof for the conversional process in Figure 7.7

7.2.3 Simulations

In simulation study, we adopt the same network topologies as those in Chapter 4. The flows with bit-rate 110kbps are put into two simulation scenarios.

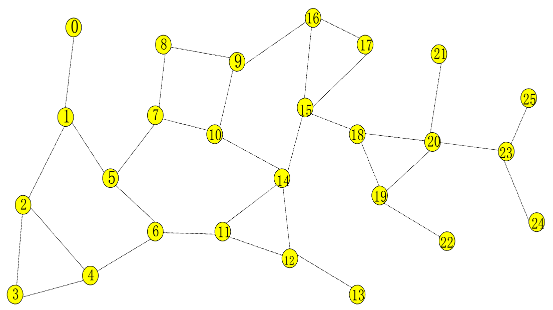


Figure 7.9: Simulation topology (I)

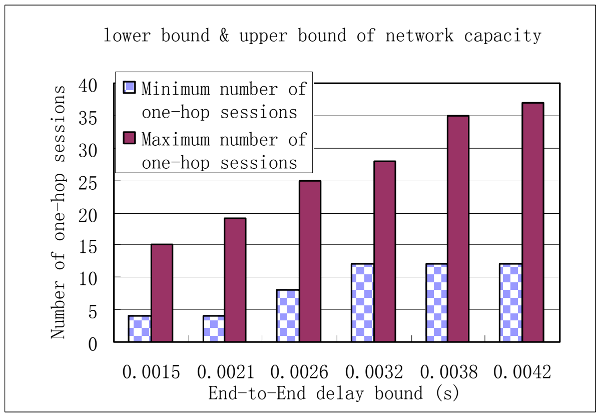


Figure 7.10: Lower bound and upper bound of network capacity for simulation topology (I)

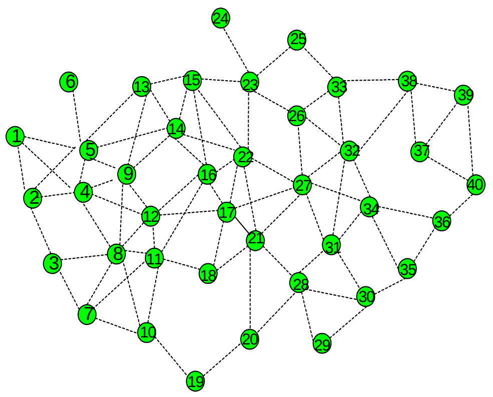


Figure 7.11: Simulation topology (II)

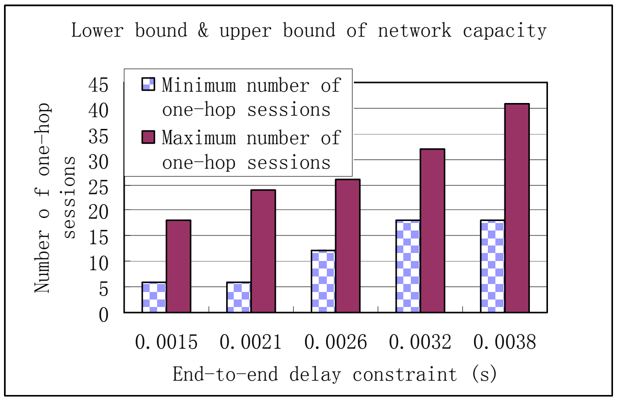


Figure 7.12: Lower bound and upper bound of network capacity for simulation topology (II)

Figure 7.10 and 7.12 present the lower bound and upper bound of network capacity for simulation topologies in Figure 7.9 and 7.11 respectively. The upper bound can be used to guarantee special QoS, and lower bound can be used to estimate the network resources utilization. Both of them are important in ad hoc networks.

Figure 7.10 and 7.12 show that though both the low bound and the upper bound are both increasing with the enhancement of end-to-end delay constraints, the upper bound increases faster than the lower bound. However, they cannot increase unlimitedly due to the limitation of bandwidth. Therefore, the upper bound will be fixed when the end-to-end delay constraint is larger than certain threshold, while the lower bound will keep fixed when the end-to-end delay constraint is large enough to allow a flow transmitting on the longest path in the network reserved by this flow. The conclusion is that the bandwidth restricts the upper bound of the network capacity finally and the network diameter restricts the lower bound of the network capacity.

Chapter 8 Conclusions and Future Work

In this thesis, we have designed an adjacency matrix based network model, explained the algorithms to obtain the upper bound and lower bound of the network capacity with delay constraints, described the analytical process and formulas for maximum queuing delay estimation and end-to-end delay estimation.

In the next section, we review the major contributions of the thesis. Subsequently, section 8.2 discusses the additional work that could be done in the future.

8.1 Contributions

The objective of the study in this thesis is to determine the network capacity with predefined end-to-end delay. The capacity metric is defined as the number of one-hop sessions an ad hoc network can support simultaneously, which not only depicts the network capacity, but also can be adopted to guarantee certain QoS. This is a special feature of our metric that may not be supplied by most other capacity metrics. The upper bound and lower bound of our metric are derived for different purpose in QoS issues: upper bound can be adopted in admission control algorithm to guarantee the end-to-end delay of the flows and the lower bound can be applied to judge if the network resources have a good utilization.

The major contributions of this thesis are as follows:

- Build a network model based on adjacency matrixes which denote the network topology.
- Define a network capacity metric according to the purpose that can be applied to

guarantee certain QoS.

- Propose two algorithms to derive the upper bound of network capacity for two scenarios: non-channel-sharing scenario and channel-sharing scenario based on an average hop count computation algorithm.
- Derive the formulas for the maximum queuing delay experienced by a packet in the queue of a node and end-to-end delay of a flow under different network scenarios. During this process, we derive the first and second moments of the service time and inter-arrival time for each node.
- Develop an algorithm to infer the lower bound of network capacity.

8.2 Future Work

This thesis has explained the purpose and necessity of the network capacity metric definition. Therefore, in the future, we will develop admission control algorithms to prevent the flows in the network from violating the end-to-end delay constraints. This can be used in scenarios where real-time services are supported.

The thesis also described the method and formulas to infer the maximum queuing delay and end-to-end delay. As we have pointed out before, the unique features of ad hoc networks such as no centralized infrastructure, limited power, multi-hop transmission, etc. result in complex correlation among nodes in the network. The IEEE 802.11 MAC protocol further increases this correlation. Therefore, the delay estimation process should take all these into consideration. In current stages, in order to simplify the derivation of the queuing delay and end-to-end delay, we assume a couple of independent relation among nodes and packets, which result in the imprecise results. In the future, we can use correlation queuing theory to revise our scheme to get more precise results.

References

[1] S. Chakrabarti, and A. Mishra, “QoS Issues in Ad Hoc Wireless Networks”, IEEE Communications Magazine, Vol. 39, No. 2, Feb. 2001, pp. 142-148.

[2] Z. J. Haas et al., “Guest Editorial,” IEEE JSAC, Special Issue on Wireless Networks, Vol. 17, No.8, Aug.1999, pp. 1329-32.

[3] R. Ramanathan, and J. Redi, “A Brief Overview of Ad Hoc Networks: Challenges And Directions”, IEEE Communications Magazine, Vol. 40, No. 5, May 2002.

[4] ZJ. Haas, “Panel Report on Ad-Hoc Networks”, Mobile Computing and Communications Review, Vol. 2, No. 1, 1997, pp. 15-18.

[5] C.-K.Toh, “Ad Hoc Mobile Wireless Network: Protocols and Systems”, Published by Prentice Hall ISBN 0130078174, Chapter 8, Dec. 2001, pp. 117-119.

[6] J. Zhang, and W. K. G. Seah, “Topology-based Capacity Analysis for Ad Hoc Networks with End-to-End Delay Constraints”, IEEE Frontiers of Mobile and Wireless Communication (MWC’04), 2004.

[7] P. Gupta and P.R. Kumar, “The Capacity of Wireless Networks”, IEEE transactions of Information Theory, Vol. 46, No. 2, Mar. 2000, pp. 388-404.

[8] S. Toump is and A. Goldsmith, “Ad Hoc Network Capacity”, Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, Vol. 2, 2000, pp. 1265 –1269.

[9] P. Gupta and P. Kumar, “Internets in the Sky: The Capacity of Three Dimensional Wireless Networks”, Communications in Information and Systems, Vol. 1, No. 1, 2001, pp. 39–49.

[10] P. Gupta and P.R. Kumar, “Critical Power For Asymptotic Connectivity in Wireless Networks”, in Stochastic Analysis, Control, Optimization and Applications, Eds. WM.McEneany et al., Birkhauser, Boston, 1998, pp. 547-566.

[11] M. Grossglauser, D. Tse, “Mobility Increases the Capacity of Ad-Hoc Wireless Networks”, INFOCOM 2001. Proceedings. IEEE, Vol. 3, 2001, pp: 1360 -1369.

[12] C. E. Caicedo, “Analysis of Results on the Capacity of Wireless Ad-hoc Network”, http://www.ece.utexas.edu/~caicedo/wireless/project_report ccaicedo.PDF.

[13] M. Gastpar and M. Vetterli, “On the Capacity of Wireless Networks: The Relay Case”, in Proc. IEEE Infocom, 2002.

[14] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, “Capacity of Ad Hoc Wireless Networks”, in Proceedings of the 7th ACM International Conference on Mobile Computing and Networking, Rome, Italy, Jul. 2001, pp. 61–69.

[15] IEEE Computer Society LAN MAN Standards Committee, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, New York, IEEE Std. 802.11, 1997.

[16] E. Uysal-Biyikoglu, A. Keshavarzian, “Throughput Achievable With No Relaying In A Mobile Interference Network”. Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC’03), June 30-July 03, 2003, pp.641.

[17] J. Li, Z. J. Haas, and M. Sheng, “Capacity Evaluation of Multi-Channel Multi-hop Ad Hoc Networks”, IEEE International Conference on Personal Communications (ICPWC’02), New Delhi, India, Dec. 2002.

[18] C. Comaniciu, H.V. Poor, "On the Capacity of Mobile Ad Hoc Networks with Delay Constraints", IEEE CAS Workshop on wireless communications and networking, Pasadena, CA, Sept., 2002.

[19] E. Perevalov and R. Blum, “Delay Limited Capacity of Ad Hoc Networks: Asymptotically Optimal Transmission and Relaying Strategy”, IEEE Proceedings of INFOCOM, April 2003.

[20] G. Bianchi, “Performance Analysis of the IEEE 802.11 Distributed Coordination Function,” IEEE Journal on Selected Area in Communications, Vol. 18, No. 3, 2000.

[21] H. Wu, Y. Peng, K. Long, J. Cheng, J. Ma, “Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement”, IEEE INFOCOMM, 2002.

[22] S. Rahman, “Throughput Analysis of IEEE 802.11 Distributed Coordination Function in Presence of Hidden Stations”, Submitted to IEEE Infocom, Hong Kong, 2004.

[23] L. Kleinrock, F. Tobagi, “Packet Switching in Radio Channels, Part II - The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution”, IEEE Transactions on Communication, 1975, pp. 1417-1433.

[24] S. Khurana, A. Kahol, S. K. S. Gupta, and P. K. Srimani. “Performance Evaluation of Distributed Co-Ordination Function for IEEE 802.11 Wireless Lan Protocol in Presence of Mobile and Hidden Terminals”, In MASCOTS, 1999, pp. 40–47.

[25] M.M.Carvalho, J.J.Garcia-Luna-Aceves, "Delay Analysis of IEEE 802.11 in Single-Hop Networks", Proc. 11th IEEE International Conference on Network Protocols (ICNP’03), Atlanta, GA. Nov. 2003. pp. 146 – 155.

[26] J. MATOUŠEK and J. NEŠETŘIL, “Invitation to Discrete Mathematics”, Clarendon Press, Oxford. 1998, pp. 109-110.

[27] A. V. Levitin, “Introduction to the Design & Analysis of Algorithms”, Villanova University, ©2003 ISBN: 0-201-74395-7, pp. 97-120, 113-119.

[28] The network simulator- ns-2. http://www.isi.edu/nsnam/ns/.

[29] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, “Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement”, In Proceedings of IEEE INFOCOM 2002, (New York, New York), June 2002.

Appendix: List of Publications

- J. Zhang, and W. K. G. Seah, “Topology-based Capacity Analysis for Ad Hoc Networks with End-to-End Delay Constraints”, in the proceedings of *the 6th IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication (MWC’04),* May 31 – Jun 2, 2004.

- J. Zhang, and W. K. G. Seah, “Capacity Evaluation for Ad Hoc Network with End-to-End Delay Constraints”, Submitted to *Issue of IEEE Journal on Selected Areas in Communications on Wireless Ad Hoc Networks,* 2004.