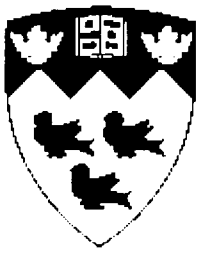
MPLS based State-Dependent Optimal Routing in IP Networks

Zhibing Wang



Department of Electrical and Computer Engineering
McGill University
Montreal, Canada

July 2003

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

ABSTRACT

A new QoS routing mechanism is proposed for MPLS capable networks to achieve global optimal route selection. The functional capabilities, which are necessary for optimizing bandwidth allocation and route selection, are identified and modeled. We focus on the design of an appropriate algorithm to combine label-switching technologies with State-Dependent routing. The algorithm, which takes into account resource utilization cost (RUC) and all available routes, is believed to be able to accomplish superior network utilization and quality of service. We devise both a centralized and a decentralized routing mechanism. The performances of both routing mechanisms are simulated and the results are analyzed. In addition, the expressions of RUC for packet-switched networks are proposed and validated.

SOMMAIRE

Un nouveau mécanisme de routage pour atteindre une sélection de chemin globale optimale avec qualité de service pour réseau de commutation multi protocole par étiquette est présenté. Les capacités fonctionnelles qui permettent une optimisation de l'allocation de la largeur de bande et de la sélection du chemin sont identifiées et modélisées. Nous orientons le travail sur un algorithme qui combine la commutation par étiquette et le routage dépendant de l'état. Nous croyons que l'algorithme, qui prend en compte le coût d'utilisation des ressources (CUR) et tous les chemins disponibles, est capable d'accomplir une meilleure utilisation du réseau et de qualité de service. Nous développons un mécanisme de routage centralisé et décentralisé. Les caractéristiques des deux mécanismes sont évaluées par simulations. Les résultats obtenus sont présentés et analysés. De plus, les expressions du CUR pour les réseaux à commutation par paquets sont proposées et validées.

Acknowledgments

This thesis would not be possible without the guidance and help of my supervisor, Professor Jean Regnier. I would like to thank him for his support throughout my graduate study at McGill University.

Grateful acknowledgment is expressed to Nortel Networks for sponsoring our research and particularly to Dr. James Yan from Advanced Network Performance, Nortel Networks, for his supervision and helpful suggestions.

I also want to thank my friends, Peng He, Xiaojian Lu, Benoît Pelletier, Marc-Antoine Parent and all of the people in the TSP Lab for their friendship and timely help.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ATM	Asynchronous Transfer Mode
BB	Bandwidth Broker
CRC	Centralized Routing Control
CRDN	Centralized Routing Decision Node
DCR	Dynamically Controlled Routing
DF	Direct First Policy
DiffServ	Differentiated Service
DRC	Decentralized Routing Control
DS	Differentiated Services
FEC	Forwarding Equivalence Class
IETF	Internet Engineering Task Force
IP	Internet Protocol
LDP	Label Distribution Protocol
LLP	Least Loaded Path
LLR	Least Load Routing
LSP	Label Switched Path
LSR	Label Switching Router
DNHR	Dynamic Non-Hierarchical Routing
MC	Minimum Cost Policy
MDP	Markov Decision Process
MPLS	Multiprotocol Label Switching
NS2	Network Simulator II
OSI	Open System Interconnect Reference Model
QoS	Quality of Service
RSVP	Reservation Protocol
RUC	Resource Utilization Cost
SDO CLASSIFIER	State-Dependent Optimal Classifier
SDOR	State-Dependent Optimal Routing
SLS	Service Level Specification
SPF	Shortest Path First
TCP	Transmission Control Protocol
VCC	Virtual Channel Connection

Chapter 1

Introduction

The Internet has grown from a research network to a world-wide network, which consists of thousands of heterogeneous networks and is used daily by millions of people. The increasing popularity is accompanied by increasing demands on consumer-oriented services such as telephony and video streams, known from circuit-switched networks. This type of applications can experience quality degradation in the traditional Internet, which was originally designed as a data packet network with best-effort packet forwarding capabilities. At that time the focus was on connectivity. If no special provisions are taken, no guarantees can be made on the delay that a packet will experience on its way to its destination, and thus no Quality of Service (QoS) can be assured.

In order to guarantee a customer's QoS requirements, a network should be able to shape and control network traffic and to reserve network resources. Several frameworks have been proposed to provide QoS on the Internet. In particular, the Differentiated Service (DiffServ) framework [2, 12], which supports aggregate traffic classes rather than individual flows, is seen as the key technology to achieve QoS. In addition, DiffServ must work concurrently with a forwarding technology that supports the aggregation scheme. MPLS [3] offers the means to accomplish such a task, as already proposed in [26, 27]. Obviously, existing routing protocols need to be enhanced or replaced by QoS-aware routing algorithms if the potentials of DiffServ and MPLS are to be realized. Unfortunately, no specific QoS routing system is provided in the DiffServ and MPLS frameworks.

This thesis identifies QoS routing and traffic engineering problems faced by modern IP networks, investigates the existing routing algorithms, and with the objective of global optimized resource utilization in mind, proposes to use State-Dependent Routing (SDR) [15, 16] as the QoS routing algorithm in MPLS capable IP networks and examines its performance with several implementations.

1.1 Scope and Objectives

In this thesis, we seek a routing algorithm that provides QoS-guaranteed services and, at the same time, can achieve traffic engineering objectives. From a service provider's perspective, the goal is to maximize resource utilization and hence, maximize the revenue. We call such a routing algorithm "the optimal routing".

Routing issues have been studied extensively for all kinds of networks, Circuit-Switched networks, Packet-Switched networks, and even public transportation networks. Many different kinds of solutions have been proposed and used for IP networks, from static to dynamic, from distance-vector to link-state, from single path to multipath, from best-effort delivery to QoS constrained and from pre-computed to adaptive. Previous studies (which will be discussed in next chapter) have demonstrated the pros and cons of those solutions, which are used as guidelines of this research and can provide us rich references.

Note:

1. Without further notice, in this thesis, "*traffic*" is referring to IP packet flows.
2. The words "*connection*" and "*call*" will be used interchangeably in this thesis.
3. The words "*route*" and "*path*" will be used interchangeably in this thesis.

The optimal routing should be able to take global resource utilization into account, automatically balance load among feasible routes, and control the admission of a call as a function of available resources. In addition, it should be able to reserve resources for connections requiring QoS, and monitor and measure resource utilization. With the feedback provided by the monitoring and measuring

processes, the routing algorithm should be able to adjust itself to give its optimal performance in different load conditions. The traditional independent hop-by-hop IP routing decision paradigm can not guarantee unified treatment for packets of a connection, since the packets that belong to the same connection may be forwarded through different links. In that case, it is impossible to guarantee QoS requirements such as delay and delay jitter. Therefore, a path has to be set up from the ingress node to the egress node (or using an existing one); and resources have to be reserved along the path (either explicitly or implicitly), before a QoS connection is admitted to the network. To fulfill those processes, some signaling functions [4] provided by the MPLS framework will be necessary. In summary, we are building a dynamic, multipath, QoS constrained, adaptive and load balancing routing algorithm. With the advent of the DiffServ and MPLS framework, the optimal routing becomes possible.

Given the complexity of QoS routing and traffic engineering, this thesis doesn't try to accomplish a total solution that takes care of all the situations, but focuses on a solution for a single service class that can be classified with the same effective bandwidth [8] and mean holding time. We call it homogenous IP traffic. This study will establish a well-grounded basis for expanding the research to the multi-service class case.

1.2 Chapter Contents

The rest of this thesis is divided into 5 chapters. The second chapter, SDR and Related Work, introduces some important frameworks and definitions, and then discusses State-Dependent Routing and concerned issues in detail. The third chapter, System Model, describes how we model important network elements such as IP traffic, routing functions, traffic load distribution and network topology. The fourth chapter, Implementation, presents how we implement the routing algorithm in a simulation environment to generate more realistic results. The fifth chapter, Results and Analysis, tests the routing algorithm against our design

objectives, and presents the simulation results and the experience gained. The sixth chapter, Conclusions and Further Studies, recapitulates the major results and findings, and enumerates some interesting directions that could be further explored.

Chapter 2

SDR and Related Work

This chapter discusses existing IP networking technologies, and related IP routing and QoS routing algorithms. Those existing frameworks and previous studies establish the platform, on which we base our work. They provide us with necessary networking functionalities and guidance on weaknesses and strengths of different approaches. SDR is introduced later in this chapter and argued to be a promising candidate for QoS routing. Its background theory and some implementation considerations are also presented.

2.1 A QoS Framework: DiffServ on MPLS

With the prospect of becoming the ubiquitous all-service network of the future, the Internet needs to evolve to support services with guaranteed QoS characteristics. The Internet Engineering Task Force (IETF) has proposed the DiffServ Model, which has been conceived to provide QoS in a scalable fashion.

2.1.1 DiffServ Model

DiffServ advocates a model based on different granularities at network edges and within the network. Instead of maintaining per-flow soft state at each router, packets are classified, marked and policed at the edge of a DiffServ domain. Core routers are only required to act on a few traffic aggregates that are meant to

offer a pre-defined set of service levels. A limited set of Per Hop Behaviors (PHBs) differentiate the treatment of aggregate flows in the core of the network, in terms of scheduling priority, forwarding capacity and buffering. Currently, there have been at least two types of treatments standardized: the assured forwarding (AF) group of PHBs [28], and the Expedited Forwarding (EF) PHB [29]. Service Level Specifications (SLSs) are used to describe the appropriate QoS parameters that the DiffServ-aware routers will have to take into account, when enforcing a given PHB. Thus micro-flow-based treatment is restricted at the DiffServ domain border while the core routers deal only with aggregate flows, according to the DiffServ Code-Point (DSCP) field of the IP header. This aggregate model of DiffServ is, therefore, highly scalable.

In DiffServ, Bandwidth Brokers (BBs) are responsible for collecting network information within one DiffServ domain, and distributing such information to edge routers for further processing. For example, BBs collect link state information including available bandwidth, link delay, etc. Edge routers use this information to compute routes for connection requests, and make admission control decisions.

In order to support DiffServ, we need to be able to differentiate traffic with respect to bandwidth, delay, delay jitter and packet loss. In this thesis, a network component Centralized Routing Decision Node (CRDN) that has similar functions as a BB is used to facilitate the routing mechanisms.

Several studies have involved traffic control algorithms for aggregate service levels, packet marking and policing and preferential treatment of marked packets in the network core. Reference [30] investigates the impact of aggregation on the performance of traffic-aware routing. It is found in [30] that most of the QoS routing benefits can be achieved using a small number of paths and relatively coarse traffic splitting. Furthermore, finer optimization gain may turn out to be detrimental to the short-term performance. In addition, while DiffServ solves the scalability problem, it also unavoidably introduces the problem of service performance deviation or performance loss. This is because it assumes that all flows, which belong to the same service level in one DiffServ domain, can only be

guaranteed in the form of an aggregate level. References [31-33] discuss the issue of performance loss in DiffServ networks in detail.

This performance loss can be adjusted by properly implemented call admission control and link utilization monitoring (with Bandwidth Brokers). The idea is simple: accept a connection only if at least one path that can satisfy its QoS requirement is available; otherwise, the connection is blocked immediately. In the proposed routing algorithms, the problem is addressed by combining a Call Admission Control (CAC) algorithm (originally developed for circuit-switching network) with the CRDN.

In DiffServ networks, traffic is classified into three service classes: premium, assured and best-effort. This research studies premium class homogenous IP traffic, which can be characterized by a single set of QoS requirement and flow parameters. This research can be seen as an effort to solve the QoS routing problem for one type of applications, e.g. voice over IP calls.

2.1.2 MPLS Framework

MPLS arises as a natural stage in the evolution of the label-swapping paradigm, offering improved performance in the organization and management of data streams. In MPLS, a short fixed length value called label is assigned to a packet, as the packet enters an MPLS domain. The packet is forwarded to its next hop together with this label. Labels are used to identify the forwarding equivalence classes (FEC) of data packets. A group of packets that are forwarded in the same manner are said to belong to the same FEC. FEC can be seen as an organized way of assigning and managing labels, to efficiently achieve the desired aggregation scheme. The value of a label depends on the FEC, to which the respective packet stream belongs, which is commonly inferred from its IP destination address. At each node the packet's label is used to perform an indexed search in a table that identifies the output port for the next hop and the new label to be used to replace the old one for forwarding the packet over the next hop. In addition, the packet's class of service information can also be

determined from its label (mapped to FEC). Packets are thus forwarded over the MPLS network based on labels attached to them in each hop, and no further analysis of the network layer headers is necessary.

Moreover, MPLS does not define a new layer in the classical open systems interconnection (OSI) communications model; instead, it can be seen as a bond between protocols operating at the network and the data link control (DLC) layers. These and other features give MPLS its superiority, and distinguish it as a promising new technology for supporting enhanced networking tasks, such as the design of virtual private networks (VPNs), traffic engineering, and explicit routing. A DiffServ and MPLS capable network is assumed in this thesis.

The fact that a packet is assigned a label as it enters the network allows the use of advanced forwarding techniques. A packet entering the network at a particular router can be labeled differently than the same packet entering the network at a different router. As a result, some kind of policy routing can be easily made. Since MPLS decouples forwarding from routing, it is able to support a large variety of routing policies that are difficult or impossible to implement with just conventional network layer forwarding. Explicit routing and multipath routing are the most interesting features offered by MPLS for QoS routing and traffic engineering.

An explicit route is specified as a sequence of hops rather than being determined by conventional layer 3 routing algorithms on a hop by hop basis. Thus, MPLS distinguishes from conventional IP routing by evolving a routing technique called explicit routing [5, 11]. An explicit route needs to be specified at the time that labels are assigned and does not have to be specified with each IP packet. Explicit routes can be used to support policy routing and traffic engineering. The ability to setup explicit routes allows network administrators to control how traffic flows through their networks. In MPLS an explicitly routed LSP (Label Switched Path) is considered as a tunnel. When a packet enters the network, its path, QoS, and forwarding class are already fully determined.

Multipath routing can also be supported in MPLS. Specifically, Label Switched Routers (LSR) may support multiple routes for a particular FEC. MPLS assigns

multiple labels to the FEC, one for each route. There are a number of obvious reasons why it may be desirable to use explicit and multipath routing instead of conventional IP routing. For example, this allows selection of routes based on administrative policies and careful route design to allow traffic engineering. As we show here, multipath and explicit routing features can also be used to facilitate QoS support for multiple classes of services in MPLS capable networks. The end result will be a load balanced network optimized to maximize operators' revenue and customers' satisfaction.

2.2 Effective Bandwidth

In a packet network, sources do not require dedicated bandwidth (e.g., circuits) for the entire duration of a connection. The actual bandwidth (bit rate) needed by each connection is uncertain and fluctuates over time. The actual required bandwidth fluctuates between some minimal level, perhaps 0, and a peak rate, which is typically determined by the speed of the access line. With packet multiplexing, the bandwidth of a link could be segmented into many different sized shares. The flexibility of packet networks also makes it more difficult to effectively control the admission of connections seeking to enter an existing network, and to plan the capacity of future networks when they are designed.

2.2.1 The Definition

The problems of admission control and capacity planning in a packet network may be addressed by a concept known as the effective bandwidth of a connection. The following is the definition of effective bandwidth given by Frank Kelly in [8]. "*Let* $X[0,t]$ *be the amount of work that arrives from a source in the interval* $[0,t]$. *Assuming that* $X[0,t]$ *has stationary increments.*

$$a(s,t)=\frac{1}{st}\log E\left[e^{sX[0,t]}\right] \quad 0<s,t<\infty"$$

When employing this concept, an appropriate effective bandwidth is assigned to each connection; and each connection is treated as if it required this effective bandwidth throughout the active period of the connection. The feasibility of admitting a given set of connections may then be determined by ensuring that the sum of the effective bandwidths is less than or equal to the total available bandwidth (i.e., the capacity). Kelly presents this additive property [8] as follows. "*If* $X[0,t]=\sum_i X_i[0,t]$, *where* $(X_i[0,t])_i$ *are independent, then* $a(s,t)=\sum_i a_i(s,t)$."

By using effective bandwidth in this manner, the problems of admission control and capacity planning are addressed in a fashion similar to that employed in circuit-switched networks. Moreover, with MPLS LSP tunnel, before a connection is admitted to a network, its forwarding path and FEC treatments have been fully determined. These characteristics inspire us to adopt those well-studied and well-tested call admission control and routing algorithms developed for Circuit-Switched networks to solve QoS routing problems. SDR, which is discussed later in this chapter, is one of those algorithms that are deemed to be ideal to support QoS routing.

In addition, this concept simplifies the QoS constraints to only one parameter when considering constructing a QoS routing algorithm [10, 14]. In emerging multi-service telecommunication networks (e.g. Based on ATM technology) QoS is defined in terms of delay, jitter, throughput, service availability, or any other service specific measurements that are applicable. With this definition, a QoS routing decision is a multiple-objective problem that does not have a uniform measure standard and perhaps even contradict each other. The concept of effective bandwidth can be seen as a generalization of the QoS definition, which leaves us with only one objective to focus on, the bandwidth. In another word, QoS is assumed to be guaranteed, if a connection is allocated with its effective bandwidth. This assumption is easy to verify. Considering a MPLS capable network (which is assumed in this thesis), a connection is assigned its effective bandwidth and forwarded along a LSP tunnel according to a FEC. Every packet

in this connection will follow the same route; propagation delay is supposed to be negligible; and the queuing priority is fine-tuned for its FEC. Hence, the delay, jitter, throughput and service availability are all fixed in the network; the QoS is then fixed.

2.2.2 Estimating the Effective Bandwidth

Different approaches [19, 20] have been proposed to estimate the effective bandwidth of sources. Often, the amount of traffic, produced by a source, as a function of time is characterized. The ultimate goal of this kind of traffic characterization is to determine the resources required to carry the source through the network, which directly depends on the queuing priority of the source. However, for this research, the concept of effective bandwidth is only used as a unit measurement for homogenous IP traffic. It is evident that the effective bandwidth of a connection should be some value between its average rate and its peak rate. Any particular value that is used is necessarily an approximation, but potentially a very useful approximation. We picked a value, by observations, between the average rate and the peak rate of traffic generators as the effective bandwidth. For homogenous IP traffic, there is only one kind of traffic generator (discussed in Chapter 4) used in the simulations.

Traffic shaping and conditioning are not the focus of this thesis. However, we assume that traffic sources will obey their predetermined traffic level, won't flood the network; and the network can enforce its Service Level Agreements (SLAs) with its customers, and won't allow misbehaving traffic to enter.

2.3 Markov Decision Process for State-Dependent Routing

2.3.1 Background

The introduction of electronic switching with stored-program control has made it possible to consider sophisticated call-routing algorithms [34 - 44] for implementation. Among those are routing schemes that use real-time or near real-time network state information at the time of a call arrival to select a route for that call (e.g., DNHR [35, 45], DCR [46, 47], SDR [15, 37], FLR [41]); we refer to them as State-Dependent Routing (SDR) schemes.

AT&T Bell Laboratories developed the scheme of Dynamic Non-Hierarchical Routing (DNHR) [45]. In DNHR, the sequence in which a call hunts for an available route within its permitted set of routes is made a function of the hour of day. The intention is to make use of spare capacity that exists in portions of the network because of non-coincidence in the occurrence of peak traffic demands in the different time zones of the country. In addition, there are features designed to make the routing adaptive, in real time, to actual network conditions. This was followed by studies at Bell Northern Research [34] and AT&T Bell Laboratories [35] on real-time routing schemes responsive to the actual state of the network, monitored at each call arrival (in the ideal case) or at intervals of a few seconds.

Studies on SDR began at Bellcore right from its inception in 1984. These investigations were directed at using the insights of Markov decision theory to develop a theory of SDR and formulate SDR schemes [37, 41] that offer impressive performance.

In [48] and [37], Markov decision theory was employed to compute a state dependent routing policy off-line by executing a single policy iteration step. The Markov Decision Process (MDP) approach has been tested favorably against a sequential routing rule similar to a basic version of DNHR [45], and against a "Least-Load Rule" similar to that proposed for the Trunk Status Map [35], a

precursor to the Real-Time Network Routing (RTNR) scheme that is now used in AT&T's long distance network [1]. The policies derived from the MDP approach require the evaluation of different cost functions, at least one of which can be estimated efficiently [15]. More recently, Dziong et al. [18] introduced the maximum revenue criterion and link shadow prices (first introduced by Kelly [21]) into this approach and demonstrated even further improvements. Such SDR schemes go beyond adjusting to predictable traffic variations and seek to exploit even the statistical fluctuations exhibited by traffic streams, i.e., they seek to make the most of even the random fluctuations of spare capacity that might be observed in the network.

A state-dependent scheme seeks to route each call so as to minimize the risk of blocking future calls, and thus responds to the current state of the network on the basis of certain assumptions about future traffic demands. Thus, it is an application of feedback control in the network. Since feedback control, in general, offers the possibility for automatic adjustment to actual conditions, SDR has the potential for a degree of robustness in network response to varying load patterns and failure configurations. This flexibility is one of the main benefits of SDR, and is of increasing importance in modern networks that plan to offer a variety of new services, for which reliable demand forecasts are not yet available.

2.3.2 SDR Overview

Let us reduce the network to its essential structural features—a set of nodes (switches), N, interconnected by a set of links, L, and a set of possible origination-destination (O-D) pairs, W. Assume that at O-D pair w, a call will require one unit of bandwidth (as effective bandwidth) on each link ℓ of the path at the time of call origination. The selected route ϕ^w consisting of one unit of bandwidth on each link of the path, is then occupied for the duration of the call. At the end of the call, the occupied bandwidth on each link becomes available for the connection of other calls. Thus, calls arrive at random and are routed on

suitable paths whenever possible; rendering those paths unavailable to other calls for the duration of the routed calls. Thus, the traffic produces random variations in the link occupancies in the network. The aim of a routing scheme is to influence, to the extent possible, these random variations in the pattern of link occupancies so as to minimize network blocking (the probability of an arriving call finding no suitable idle path for its connection).

The routing of a call arriving in a given configuration of link occupancies can thus be viewed as a choice of state transition from the given state of the network. It thus becomes evident that the routing problem can be characterized as a MDP [49, 50], i.e., as the probabilistic control of a system whose working can be modeled by the evolution of a Markov chain. This is a well-established branch of study in stochastic control; and there exist (in principle) well-defined methods for finding an optimum solution. V.E. Benes [51] presented a formal treatment of the traffic routing problem as a MDP with an average-reward criterion in 1966.

The basic assumptions regarding the stochastic nature of the network are as follows:

1. The incoming calls are Poisson distributed with rate λ;
2. The holding times of all calls are exponentially distributed with unit mean time, μ^{-1};
3. Blocked calls do not retry;
4. Call set-up times are negligible;
5. All routes have at most two links;
6. When a call is routed on a j-link path, it becomes j independent calls, each with an independent holding time.

Assumptions 1-4 are fundamental for most teletraffic modeling. Assumption 5 pertains to networks with dynamic routing, since it has been established that more than two hops is the point of diminishing returns for the benefits of sophisticated routing and an invitation for instability. Assumption 5 & 6 relate to the treatment of the traffic routing problem as a MDP and will be discussed in the next section.

2.3.3 Formulating the Routing Problem as a MDP

In treating traffic routing as a Markov decision process, the exact characterization of the network state would require the specification of the number of calls in progress on each possible route in the network. Such a detailed specification of state is computationally infeasible for any realistic network due to the huge state space of the Markov decision process.

Numerous studies, e.g., [18, 15], have proposed the link independence assumption and a consequent decomposition of the path cost into a set of separable link costs, referred to as the path cost separability assumption. In the following, we first introduce these two assumptions. We then describe the decomposed Markov decision process that results form these two assumptions.

- **The link independence assumption:**

This assumption assumes that ([18, 53, 54, and 15]):

1. Calls arrive to any link ℓ according to independent Poisson process with rate λ^{ℓ}.
2. A call carried on a path consisting of n links behaves like n independent calls, i.e., the link holding times for the call on each of the n single links are statistically independent.

- **The path cost separability assumption:**

This assumption was first introduced in [15]. The idea is to assume that the cost functions associated with adding a call on path ϕ_j^w to the network are separable, i.e., $C_{\phi_j^w}(\mathbf{x}) = \sum_{\ell \in \phi_j^w} C^{\ell}(\mathbf{y}_{\ell})$, where ϕ_j^w is the j[th] route of O-D pair w, $C_{\phi_j^w}(\mathbf{x})$ is the cost of accepting a call under the network state $\mathbf{x}$, $C_{\ell}(\mathbf{y}_{\ell})$ is the cost of using link ℓ when there are already $\mathbf{y}$ calls on the link.

By assuming statistical independence of the links in the network, we arrive at a simpler definition of network state as the set of bandwidth occupancies (with the effective bandwidth as unit), i.e., the vector of link states. This amounts to regarding a multiple-link call in progress as several independent calls on those links, with independent departures. This is a reasonable approximation whenever

the number of calls common to two links is a small fraction of the total number of calls in progress on either link — a condition that should hold if most calls are direct-routed or if each link is used by calls of many different node pairs. Now suppose that we want to route a call on a path with more than one link, while still restricting all other calls to direct routes. The path cost for adding this call is thus simply the sum of the individual link costs. The link independence assumption and the path cost separability assumption are satisfied if direct routing (discussed in the next section) is used and each O-D pair has its own direct link.

If n is the number of links, and x_i is the occupancy of link i, then the network state is given by the vector $x=(x_1,x_2,\cdots,x_n)$. The state transitions are rather simple: the departure of a call in progress on link i changes the state to $(x_1,x_2,\cdots,x_i-1,\cdots,x_n)$. The arrival of a call either produces a new state if the call is carried, with the occupancies of each link on the chosen route increased by 1, or leaves the state unchanged if the call is blocked.

The routing rule or policy can be described by a function $R(x,w)$, which specifies the network state attained as a result of the routing decision when a call of node pair w arrives in state x. We define an optimum routing rule as one that minimizes the expected steady-state blocking rate in the network. The blocking rate and the routing policy R are connected by an equation for each network state [49, 50]. Appearing in these equations are the so-called Howard relative cost functions [49, 50], $\{V_R(x)\}$, which allow us to attach a cost to each state such that $[V_R(y)-V_R(x)]$ is the difference in the expected total number of calls lost in the network under policy R, starting from the initial state y and starting from the initial state x. The cost $\{V_*(x)\}$ under optimum routing, it is clear, must be consistent with the following routing rule:

When a call arrives in state x, we should choose, from the possible actions, the one that leads to the state with the smallest cost.

Noting that an outright rejection of a call leaves the state unchanged but incurs an immediate loss of one call. We can now describe the optimum SDR rule as follows, in terms of $\{V_*(x)\}$, the corresponding relative costs:

When a call arrives in state x, let y be the state, among those that can be reached by feasible routing choices, with the smallest cost. Then,

(a) If $|V_*(y) - V_*(x)| < 1$, chose the route corresponding to y.

(b) If $|V_*(y) - V_*(x)| \geq 1$, reject the call. (2.3.3.1)

The criterion of the second part of equation 2.3.3.1 is an important, built-in congestion control feature of SDR. It weighs the risk that the current call (if carried on a multiple-link route) might cause more than one future call to be blocked during its lifetime. It is a natural form of trunk reservation [45] that acts to limit the extent of multiple-link routing during overloads, thus preventing congestion from spreading in the network.

The exact determination of the relative costs $\{V(x)\}$ for optimum routing is a hopeless task for practical networks, since the number of equations that must be solved equals the size of the state space [49]. However, if the cost $\{V_R(x)\}$ for some routing scheme R are used in equation 2.3.3.1 to derive a new routing scheme $\overline{R}$, then it is known [49] that $\overline{R}$ is superior to R (unless, of course, R is optimum, in which case $\overline{R} = R$). This procedure of improving upon an initial policy is known as "policy improvement" [49, 50]; and it offers us a ray of hope. If we could derive the relative costs from some reasonable initial routing scheme without having to solve a system of equations the size of the state space, then we can use equation 2.3.3.1 to improve upon it. We now show how the relative costs can be determined from the case of direct routing.

2.3.4 Relative Costs for Direct Routing

Consider that a Poisson stream of intensity λ, which is normalized for the choice of the mean holding time as the unit of time, is offered to a single link with capacity of s units of (effective) bandwidth. The state of the link is defined by the number, p, of occupied bandwidth units.

We define, for $p = 0,1,\cdots(s-1)$, $D_p = V(p+1) - V(p)$ as the expected number of additional calls blocked (in the long run) when we start with $(p+1)$ busy bandwidth units rather than with p busy bandwidth units.

Consider a start from the initial state p. Sooner or later, the link occupancy makes its first passage to the state $(p+1)$; the subsequent course of events for this case corresponds to the case of a delayed start from the initial state $(p+1)$, the delay being the first-passage time from state p to state $(p+1)$. Considering the effect of the delayed start on the expected asymptotic difference in blocked calls in the two cases, we obtain the formula

$$D_p = M_p E_b(s,\lambda) \qquad (2.3.4.1)$$

where M_p is the expected number of call arrivals in an interval of length equal to the mean first-passage time from state p to state $(p+1)$, and $E_b(s,\lambda)$ is Erlang-B blocking of the link.

By examining the possible state transitions from state p, it is easy to see that M_p satisfies the first-order difference equation

$$M_p = \frac{\lambda}{\lambda + p} + \frac{p}{\lambda + p}\left|M_{p-1} + M_p\right| \qquad (2.3.4.2)$$

with the initial conditions $M_0 = 1$.

The solution of equation 2.3.4.2 is given by $M_p = \frac{1}{E_b(p,\lambda)}$ and using this in equation 2.3.4.1, we obtain the following simple formula for the relative cost of adjacent states $(p+1)$ and p:

$$D_p = \frac{E_b(s,\lambda)}{E_b(p,\lambda)} \qquad (2.3.4.3)$$

D_p can also be seen as the possibility of blocking one future call when the link moves from state p to state $(p+1)$.

In a direct-routing network, the links are, in fact, independent; and, as a result, the relative cost of a network state $x = (x_1, x_2, \cdots, x_n)$ is separable and additive in the component states of the individual links. Equation 2.3.3.1 is used as the call admission control criterion for direct routing. Suppose that a call arrives in state x and has to choose from the following three available routes:

Route 1: direct route consisting of link 1

Route 2: two-link route consisting of links 2 and 3

Route 3: two-link route consisting of links 4 and 5

The cost $\{C_j\}$ associated with the corresponding state transitions are obtained by using equation 2.3.4.3, using link subscripts on the variables:

Route 1: $C_1 = \dfrac{E_b(s_1,\lambda_1)}{E_b(x_1,\lambda_1)}$

Route 2: $C_2 = \dfrac{E_b(s_2,\lambda_2)}{E_b(x_2,\lambda_2)} + \dfrac{E_b(s_3,\lambda_3)}{E_b(x_3,\lambda_3)}$

Route 3: $C_3 = \dfrac{E_b(s_4,\lambda_4)}{E_b(x_4,\lambda_4)} + \dfrac{E_b(s_5,\lambda_5)}{E_b(x_5,\lambda_5)}$

Let $C = \min(C_1, C_2, C_3)$. Then, one of the routing rules can be:

If $C < 1$, route the call on the minimum-cost route;

If $C \geq 1$, reject the call.

Let's call this rule the minimum-cost (MC) policy. We note that, since $C_1 < 1$, whenever $x_1 < s_1$, a call that can be carried on a direct link will never be rejected; however, the call may not be necessarily carried on the direct link, e.g. when $1 < C_1 < C_2$. In this thesis, we will also examine a slightly different policy. Still using the above example:

If $(C_1 \leq 1)$, then $\{$route the call on link 1;$\}$

else if $(\min(C_2, C_3) \leq 1)$, then $\{$route the call on the minimum-cost route;$\}$

else $\{$reject the call;$\}$

Let's call it direct-first (DF) policy. The first line of the policy gives the direct link priority over two-link routes. This modification leads to more efficient network utilization and the same time, saves some cost computation, especially when the number of admissible paths is large. In addition, with the cost still less or equal to 1, the blocking rate will not be compromised. In fact, we found that DF gives better performances, because of its "direct first" policy. We will compare the two routing policies in chapter 5. The discussion in the rest of this thesis will be based on the DF policy.

2.4 Resource Utilization Cost (RUC)

Based on the independence assumptions introduced in section 2.3.3, we are going to examine the cost of admitting a call to a network from a different perspective: define the cost as the probability of blocking future calls, and then formulate the average cost function by averaging the probability over its (the call's) mean holding time.

2.4.1 First Passage Times

Consider the utilization of a link with resource $r = 0, \cdots, N$ as a birth and death Markov process with transition rate λ_r and μ_r.

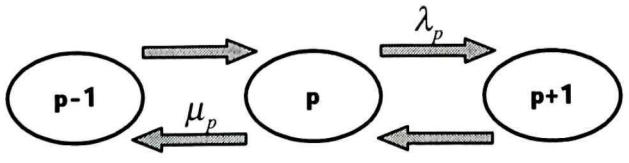


Figure 2.4.1.1: State diagram

At state $p,(p=0,\cdots,N-1)$ with call arrival rate λ_p and call departure rate μ_p, the total rate of leaving state p is $v_p=\lambda_p+\mu_p$. With probability λ_p/v_p, $p+1$ is the next state; with probability μ_p/v_p, $p-1$ is the next state. In the later case, the process still has possibility to reach state $p+1$ after it comes back to state p again. Let t_p^+ be the time taken by the process to reach state $p+1$ from state p. Let $s_p^+(x)$ be the P.D.F. of t_p^+. Then we have [52]

$$s_p^+(x)=\frac{\lambda_p}{v_p}v_p e^{-v_p x}+\frac{\mu_p}{v_p}v_p e^{-v_p x}*s_{p-1}^+(x)*s_p^+(x) \qquad (2.4.1.1)$$

where * devotes convolution.

Let $S_p^+(s)$ be the Laplace transform of $s_p^+(x)$: i.e.,

$S_p^+(s)=\int_0^\infty s_p^+(x)e^{-sx}dx$. From (2.4.1.1), we have

$$S_p^+(s)=\frac{\lambda_p}{s+v_p}+\frac{\mu_p}{s+v_p}S_{p-1}^+(s)S_p^+(s)=\frac{\lambda_p}{s+v_p-\mu_p S_{p-1}^+(s)} \qquad (p=1,\cdots,N-1)$$

For $p=0$, the time to reach state 1 is simply that of the next arrival, and is exponentially distributed with parameter λ_0. Thus: $S_0^+(s)=\frac{\lambda_0}{s+v_0}$.

Now, let t_{pq} be the time taken by the process to reach state $q>p$, starting from state p. We can write t_{pq} as: $t_{pq}=t_p^++t_{p+1}^++\cdots+t_{q-1}^+$, where $t_p^+,\ldots,t_{q-1}^+$ are independent.

Let $s_{pq}(x)$ be the P.D.F of t_{pq}, and $S_{pq}(s)$ be the Laplace transform of $s_{pq}(x)$. Hence,

$$\begin{aligned}S_{pq}^+(s)&=S_p^+(s)S_{p+1}^+(s)\cdots S_{q-1}^+(s)\\&=\frac{S_0^+(s)S_1^+(s)\cdots S_{q-1}^+(s)}{S_0^+(s)S_1^+(s)\cdots S_{p-1}^+(s)}\end{aligned} \qquad (2.4.1.2)$$

This is an important result that is used to formulate the average cost function.

2.4.2 Average Cost

We consider still the link with resource, $n = 0, \cdots, N$, and the same assumptions about the traffic as in section 2.3.4, i.e.,

1. Each call requires one resource for the duration of its holding time.
2. The call arrival process is Poisson with intensity λ.
3. Each call has an independent and exponentially distributed holding time with mean value μ^{-1}.

Suppose that a new request R arrives at time 0 and requires one resource for its holding time h, when $p < N$ resource is busy, or the link is in state p. The acceptation of R can lead to two scenarios.

In the first scenario, Figure 2.4.2.1, while the call R is holding a resource, no later calls are blocked. Hence, the request R induces no cost.

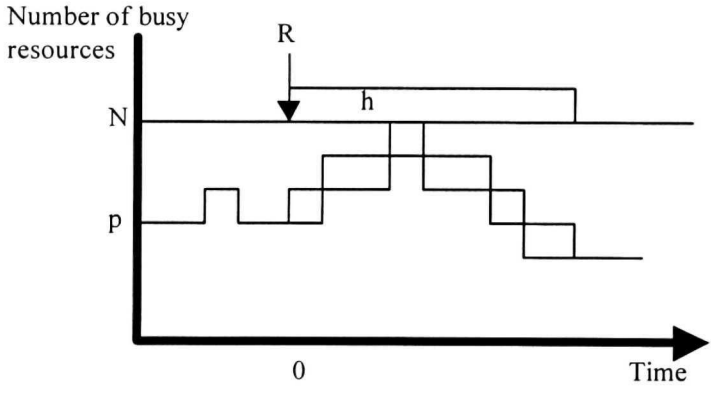


Figure 2.4.2.1: Scenario 1, request R induces no cost

In the above figure, the dotted line depicts the total number of busy resources while request R is being served.

In the second scenario, depicted in Figure 2.4.2.2, while the call R is holding one resource, a subsequent request arrives and finds all resource busy. Thus, the subsequent request is blocked. The subsequent call would have been served if R hadn't taken a resource. Therefore, the request R induces a cost of blocking one request. It is deserved to point out that request R only causes one blocking,

since after the first blocking, the link occupancy is the same as what the blocked request would have otherwise caused.

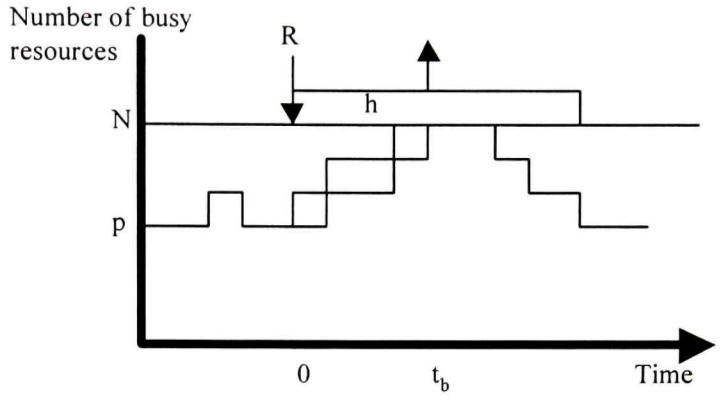


Figure 2.4.2.2: Scenario 2, request R induces the blocking of a subsequent request at time t_b

In the above figure, the dotted line depicts the total number of busy resources while request R is being served.

Let $\overline{C}_p$ be the average cost of assigning resources to a connection request R while p resources are busy. Specifically, $\overline{C}_p$ can be defined as the probability of blocking a subsequent request, while R holds its resources. In other words, $\overline{C}_p$ is the probability that the process transits from p to N during the holding time h of request R.

Averaging the possibility over the holding time h, we thus have:

$$\overline{C}_p = \int_0^\infty \Pr[t_{pN} \le t]\mu e^{-\mu t} dt = \int_0^\infty \int_0^t s_{pN}(x)dx\mu e^{-\mu t} dt = \frac{S_0(\mu)S_1(\mu)S_{N-1}(\mu)}{S_0(\mu)S_1(\mu)S_{p-1}(\mu)} \quad (2.4.2.1)$$

Combine equation 2.4.2.1 and 2.4.1.2, we can express $\overline{C}_p$ as:

$$\overline{C}_p = \frac{E_b(A,N)}{E_b(A,p)} \quad (2.4.2.1)$$

A is the traffic intensity, $A = \lambda/\mu$, p is the number of currently busy resources, N is the total number of resources and $E_b(x,y)$ is the Erlang-B blocking formula.

Since, $\overline{C}_p$ depends on p, A and N, we will refer to equation 2.4.2.1 as $\overline{C}_{A,N}(p)$.

Now, comparing equation 2.3.4.3 and 2.4.2.1, it is interesting to see that we have reached the same cost function. (Note that the equation 2.3.4.3 is using λ instead of A, because it is assuming each call has unit mean holding time, i.e. $\mu^{-1} = 1$.) This result proves that this cost function has solid theoretical basis. It can be verified from both the MDP approach and the probability approach. With this cost function, it is obvious that we can use the same routing policy as that introduced in section 2.3.4.

2.4.3 Understanding the RUC Function and the Erlang-B Blocking Formula

- **Erlang-B blocking formula**

The Erlang-B blocking formula is very important in routing studies and capacity planning. To get a better understanding of the formula, let's see some figures. Consider a typical trunk group in a well-engineered circuit-switched telecommunication network. Supposing a trunk group is offered a traffic of $A = 80Erl.$, and is designed for 1% of blocking rate with N=96. For this trunk group, the number of circuits (capacity) is determined most importantly from the expected traffic load. The figure 2.4.3.1 depicts the situation. Before the blocking rate reaches 1% (e.g. $A = 80Erl.$ when N=96), the curves are rather flat, which means that the blocking rate is not sensitive to traffic load fluctuations. However, the blocking rates begin to rise quickly after 1%. Therefore, 1% is a good threshold point for capacity planning to accommodate high traffic load and, the same time, keep the blocking rate low.

Figure 2.4.3.1 shows that after 1%, the blocking rates rise at different speed. The curve for N=96 is not as steep as the one for N=60, i.e. blocking rate for N=96 is increasing at slower speed than that for N=60. This can be explained as bigger trunk groups are more adaptable to overflow conditions.

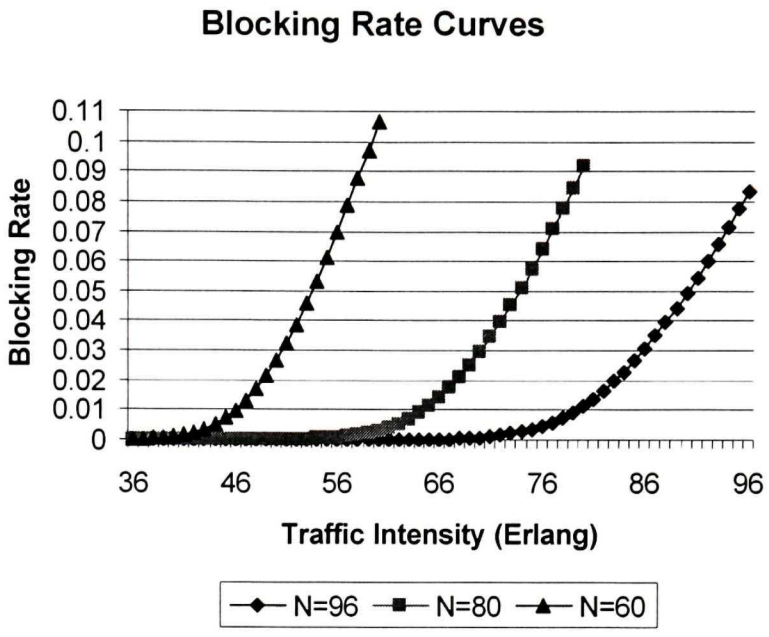


Figure 2.4.3.1: The blocking rate curves when N=96, N=80 and N=60

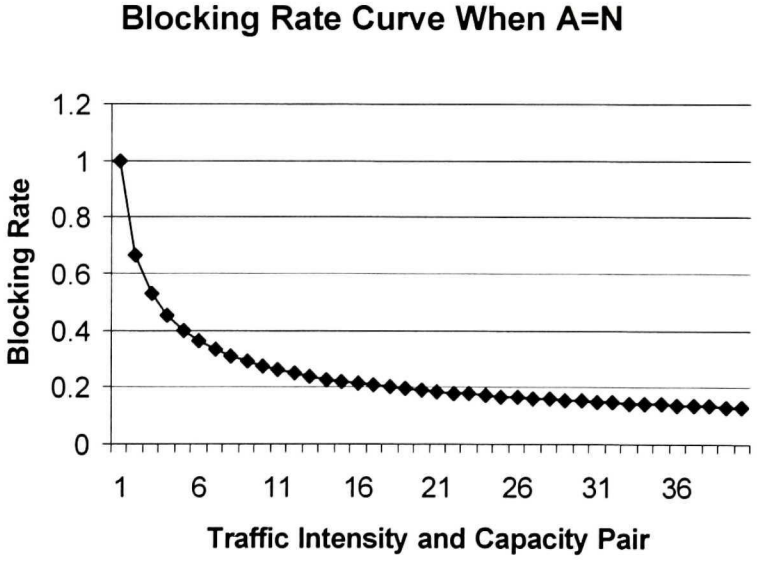


Figure 2.4.3.2: The blocking rate curve when "A=N"

Figure 2.4.3.2 delivers the same information from a different perspective. It shows the blocking rate curve for $N = 1, \cdots, 40$ under offered traffic $A = N$, (i.e. when the number of circuits is N=21, the offered load is $A = 21Erl$). As the traffic intensity and capacity pair grows, the blocking rate goes down monotonously, i.e.

as the trunk group becomes bigger, blocking becomes smaller, even in an overflow condition. Hence, in capacity planning, one big trunk group may have advantages over several small ones. However, Figure 2.4.3.3 shows that as link capacity grows, the 1% threshold point grows at a lower speed. The curve "x=y" is used to compare with the "1% Blocking Rate Curve" to show the trend.

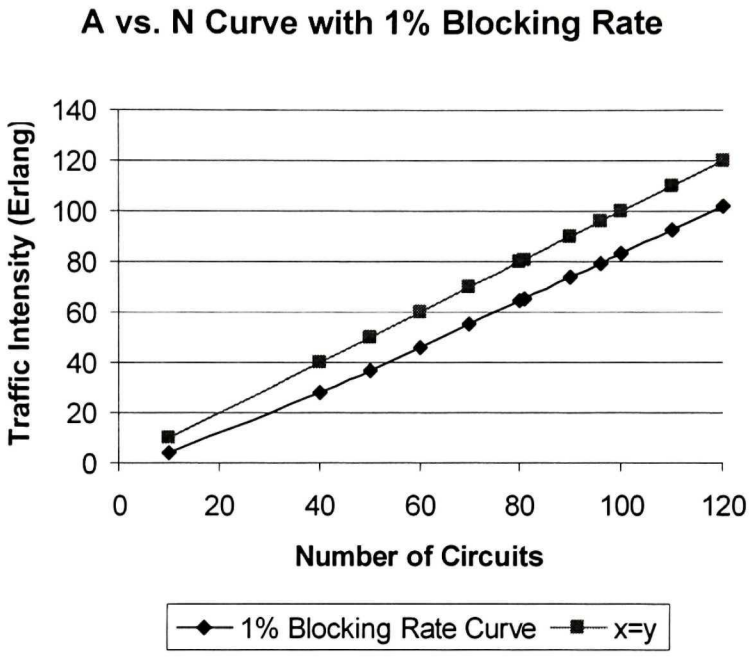


Figure 2.4.3.3: Traffic intensity vs. link capacity curve with 1% blocking rate

This result is consistent with Figure 2.4.3.1. As traffic intensity grows, the blocking rate of a high capacity link will start to move up earlier than a low capacity link, but with a lower initial slope. We will discuss more about capacity planning issues and present how we address them in our simulated network environment in chapter 4.

- **RUC function**

Figure 2.4.3.4 presents $\overline{C}_{A,N}(p)$, defined in equation (2.4.2.1), as a function of p when $A = 80Erl.$ and N=96.

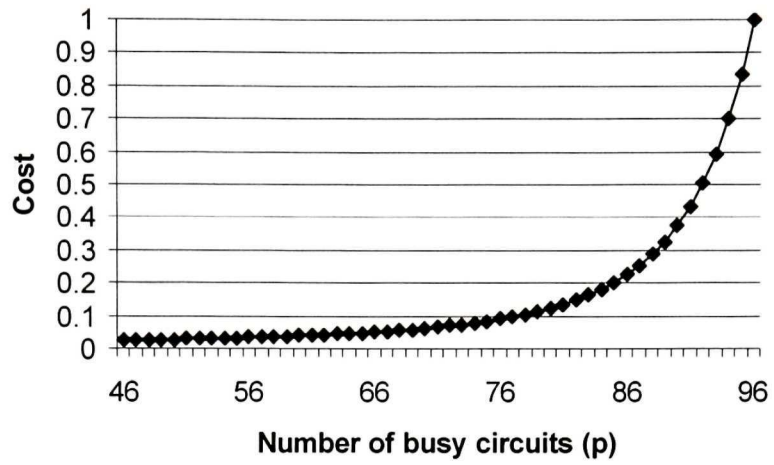


Figure 2.4.3.4: $\overline{C}_{A,N}(p)$ as a function of p for a trunk group

Figure 2.4.3.4 shows that $\overline{C}_{A,N}(p)$ monotonically increases in p, and is always less than or equal to 1. Hence, it is always profitable to assign a trunk to a call, if it only requires one trunk. Blocking a call will instantly cost 1 unit, while accepting it may at most entail 1 unit of cost. However, as the number of busy circuits rises, assigning a trunk to a call becomes increasingly costly. When the number of busy circuits approaches N, the cost increases sharply.

Now, still using above example, consider how the traffic intensity parameter λ affects the cost curves.

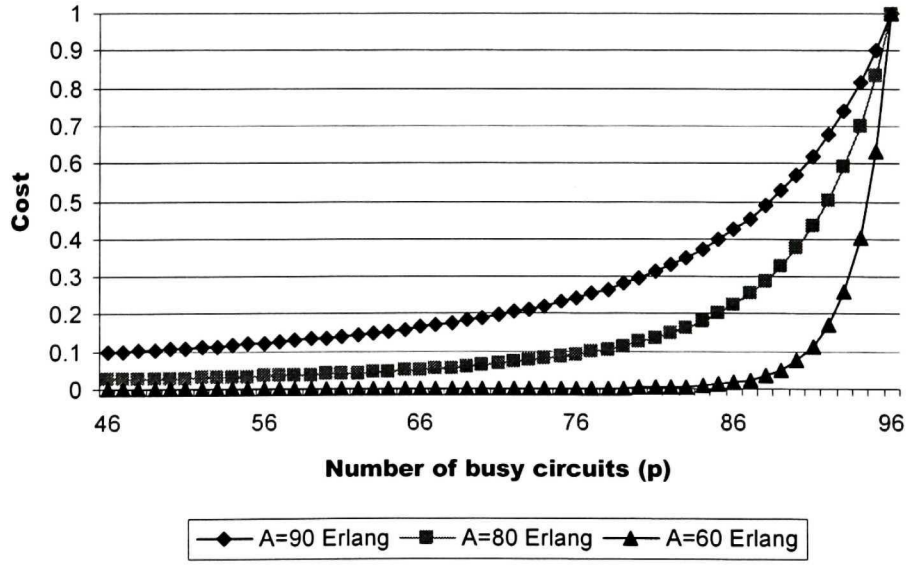


Figure 2.4.3.5: RUC cost under different traffic intensities

Comparing the three curves corresponding to $A = 90Erl.$, $A = 80Erl.$ and $A = 60Erl.$, we can see that when $p = 96$, they all reach 1. However, the cost curve of $A = 90Erl.$ is always higher than the other two curves (except when $p = 96$). Moreover, the higher the traffic intensity is, the smother the curve is. This will make the trunk reservation approximation less accurate (to be further discussed in section 2.5.2).

2.5 Detecting the Traffic Intensity

With equation 2.4.2.1, the RUC can be accurately calculated. However, $\overline{C}_{A,N}(p)$ requires the knowledge of both the real-time traffic intensity A and the network utilization p, which are difficult or even impossible to acquire. In this section, we introduce two approaches to either get the traffic intensity parameter A or remove the dependence on A with an approximation. And then, in the section 2.6, we will focus on solving the dependence on real-time network utilization p.

2.5.1 SDR-ADAPT

In general, determining the correct value of λ for each link requires solving a large non-linear program off-line [37]. It is therefore highly desirable to do away with the need for prior load information. Krishnan introduced adaptive SDR, SDR-ADAPT, in [54]. Algorithm SDR-ADAPT has no prior knowledge of the call arrival rates for each node-pair (and hence no knowledge of the offered load on each link). Instead, it collects data in the course of normal traffic measurement and adjusts routing to the traffic pattern. In SDR-ADAPT, the offered load on a link is initialized to its link capacity. For a network designed with 1% blocking rate, this initial traffic intensity is normally higher than the real intensity on the link. During the execution of the algorithm the carried loads (occupied capacities) are scanned every δ time units and the offered loads are updated every Δ scans.

Figure 2.5.1.1 shows the first update of λ. Other updates are similar.

n_ℓ : Carried load on link ℓ; λ_ℓ : Offered load on link ℓ;

B_ℓ : Blocking on link ℓ; h : Mean call holding time;

for $i = 1$ to Δ

$n_\ell(i) \leftarrow$ Carried lad on ℓ at time $i\delta$

$$I(i) \leftarrow \begin{cases} 0 & n_\ell(i) \neq C_\ell \\ 1 & n_\ell(i) = C_\ell \end{cases}$$

$$\hat{N}_\ell \leftarrow \sum_{i=1}^{\Delta} n_\ell(i) \Big/ \Delta$$

$$B_\ell \leftarrow \sum_{i=1}^{\Delta} I(i) \Big/ \Delta$$

$$\lambda_\ell \leftarrow \hat{N}_\ell \Big/ \left(h(1 - B_\ell)\right)$$

In our implementation: $h = 5$ seconds; $\Delta = 81$ seconds; $\delta = 1$ second;

Figure 2.5.1.1: The first update of the offered load for SDR-ADAPT

To be compatible with the RUC function (equation 2.4.2.1), we will use $A = \lambda / \mu$ instead of λ to describe the carried load. The traffic intensity of the whole network can be then described as a traffic intensity matrix, $\lfloor A_{i \times j} \rfloor$. With $\lfloor A_{i \times j} \rfloor$ being updated repeatedly, it will get closer and closer to the value that reflects the actual traffic pattern. This is in fact an "initial policy (direct routing) + policy improvement" process, which is an effective way to apply MDP to the routing problem.

An over estimated initial offered load won't have much impact on direct traffic, since direct routes always cost less than 1; however, it will reduce 2-hop traffic. When most (e.g. 90%) of the traffic can use direct routes, after the first update cycle, SDR-ADAPT can capture most (e.g. more than 90%, since there still will be some 2-hop traffic) of the actual offered load. During the second cycle, SDR-ADAPT may adapt to an intensity value smaller than the actual load, but very close. Hence, the blocking rate will decrease considerably, and the 2-hop rate will

increase, which leads to a higher traffic intensity. After two updates, SDR-ADAPT will have a very good estimation of the current load.

SDR-ADAPT doesn't require traffic demand knowledge in advance or in real-time. In stead, it calculates the traffic intensity matrix $\lfloor A_{i\times j} \rfloor$ once in every update cycle, Δ. This modification may lead to sub-optimal routing decisions, but it makes SDR-ADAPT a much more realistic algorithm. Moreover, after $\lfloor A_{i\times j} \rfloor$ converges to its target value, the routing decisions will also reach the optimal.

However, SDR-ADAPT still has to calculate the rather complicated RUC equation (equation 2.4.2.1) for every call admission. This could be a huge problem for a modern telecom network, which may have to respond to hundreds of newly arriving calls simultaneously. Hence, further simplification is necessary to cut down the computation.

2.5.2 Trunk Reservation (TR)

Since the cost function $\overline{C}_{A,N}(p)$ requires knowledge of the traffic demand A it is impractical to compute it in real-time. Trunk reservation [45], which eliminates the requirement for real-time traffic demand knowledge, is an effective way to approximate $\overline{C}_{A,N}(p)$.

Trunk reservation [23-25] was first developed in circuit-switched networks, in which a threshold is defined for each trunk group. If the number of idle circuits is greater than the threshold, the cost of accepting a call is considered zero. If the number of idle circuit is less than the threshold, the cost of using this trunk group is considered 1. When the cost of using a circuit for a call is less than or equal to 1, the circuit is assigned to the call. With the trunk reservation approximation, the total cost of a route is always a non-negative integer. Still using the example from section 2.4.3 ($A = 96Erl$ and N=96), Figure 2.5.2.1 illustrates the trunk reservation approximation with a threshold of five idle trunks.

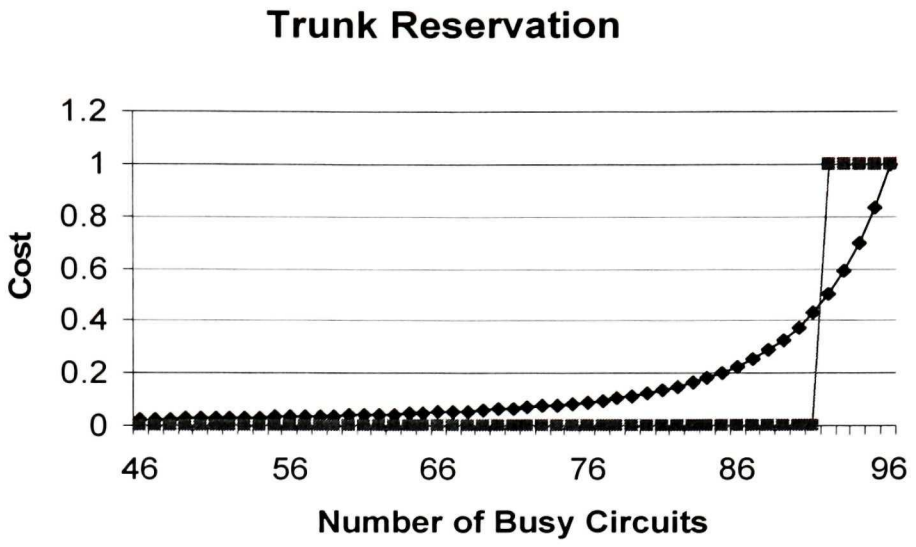


Figure 2.5.2.1: The trunk reservation approximation to $\overline{C}_{A,N}(p)$

From the discussion in section 2.4.3, we know that traffic intensities can greatly change the cost curves. Hence, under different traffic intensities, the threshold values should be different.

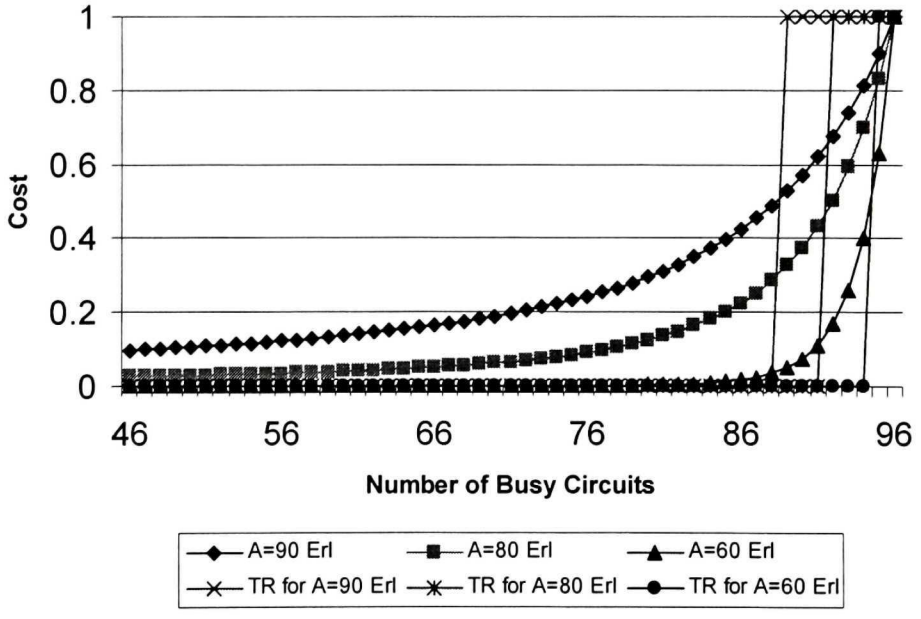


Figure 2.5.2.2: Threshold values under different traffic intensities

From Figure 2.5.2.2, we can see that the lower the traffic intensity is (in this example $A = 60$ Erlang), the more accurate the trunk reservation approximation

can be. It also shows that $A = 80$ Erlang (corresponding to 1% blocking rate) is a good threshold point for trunk reservation. When the blocking rate is higher than 1%, the TR will be less efficient. Hence, one of the network design objectives should be to limit the blocking rate on each link to 1% or less.

As an approximation to the RUC cost function, the TR reduces the cost on some link states, which are smaller than the threshold, to 0, and increases the cost of all others to 1. For a specific traffic intensity, the threshold on a link should be the link utilization state that has a cost around 0.5. Hence, when a 2-hop route costs more than 0.5 on both its links, its total cost becomes 2; when it has one link that costs less than 0.5, the total cost is 1, and the route is admissible. Since under TR, the exact cost of a given route can't be determined, i.e. the minimum cost route can't be found, and MC is devised to test SDR with exact cost, TR isn't implemented under MC. Under DF, this modification doesn't impact direct traffic directly, since direct traffic always has a less than or equal to 1 cost. However, it may cause some less optimal decisions when routing 2-hop traffic, since all 2-hop routes have costs 0, 1 or 2. Therefore, under TR, DF can only find a 2-hop route, whose links are below their thresholds. The route is not necessarily the optimal one.

The TR threshold does not need to correspond to an integer number of idle trunks. For packet-switched networks, the threshold should be some quantity of bandwidth, for example, 1 percent of total bandwidth, or several units of the effective bandwidth of a service class. In thesis, we define the threshold as follow,

$$x = Round\left(\frac{C \times \delta}{B}\right) \times B$$

Where x is the threshold; δ is a small number, e.g. 0.05 ; B denotes the effective bandwidth of the connections; C is the capacity of the concerned link; and $Round(s)$ is the function to round the variable s to the nearest integer.

$$\overline{C} = \begin{cases} 0, & \text{Available Bandwidth} \geq x \\ 1, & \text{Available Bandwidth} < x \end{cases}$$

x is affected by the total capacity of the link C, the effective bandwidth B and the traffic intensity on this link, where C is known and B is fixed. Used to reflect

traffic intensity, δ is the parameters that need to be investigated in this research. The following figure shows the relationship between the cost, $\overline{C}$ and δ.

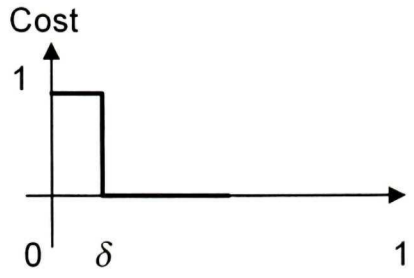


Figure 2.5.2.3: The trunk reservation approximation to $\overline{C}_{A,N}(p)$

Trunk reservation replaces the RUC equation with a simple threshold, hence, dramatically reduces the computations for SDR-ADAPT. Both SDR-ADAPT and trunk reservation are implemented in this research. The simulation results are presented and discussed in chapter 5.

2.6 Detecting the Network State

We have shown that with SDR-ADAPT, real-time traffic intensity A can be replaced by an averaged value over an update cycle; and after a few updates, the averaged value converges to its real value. With trunk reservation, the RUC function is replaced with a threshold, and no traffic intensity knowledge is required. However, there is another real-time information required in RUC, the link state, p. The discussion of SDR so far has assumed that instantaneous state information is available at each call arrival.

In practice, routing decisions can only be made based on near real-time state information, which is updated periodically. It is easy to see that a higher update frequency will lead to better routing decisions; nevertheless it also induces cost, e.g. bandwidth consumption. Clearly, there is a trade off between update frequency and the induced cost. In this section, we will introduce a method to find

out the optimal update frequency that can give the best balance. The performance of SDR under different update frequencies is studied in chapter 5.

2.6.1 Update (Sampling) Frequency

Considering the utilization of a link as a stochastic process, we can estimate a frequency range $[-f_0, f_0]$, which contains most of the energy of the process. For connection-oriented Poisson traffic with homogenous exponentially distributed holding time (with mean μ^{-1}), some related research [52] on telephone networks can be used. The following equation can help us to determine the frequency range that contains x percent of the power of the process.

$$f_0 = \frac{\mu}{2\pi}\tan\left(x \times \frac{\pi}{2}\right), \qquad (2.6.1.1)$$

Then, applying the Nyquist sampling criteria, we can well sample the process at two times the maximum frequency. The estimated update interval should be around $1/(2 \times f_0)$. From formula 2.6.1.1, we can see that the update frequency (interval) depends on the mean holding time of the offered connections. The longer the mean holding time is, the lower the update frequency is required.

In our simulations, the mean holding time is $1/\mu = 5$ seconds, we can have the following figure.

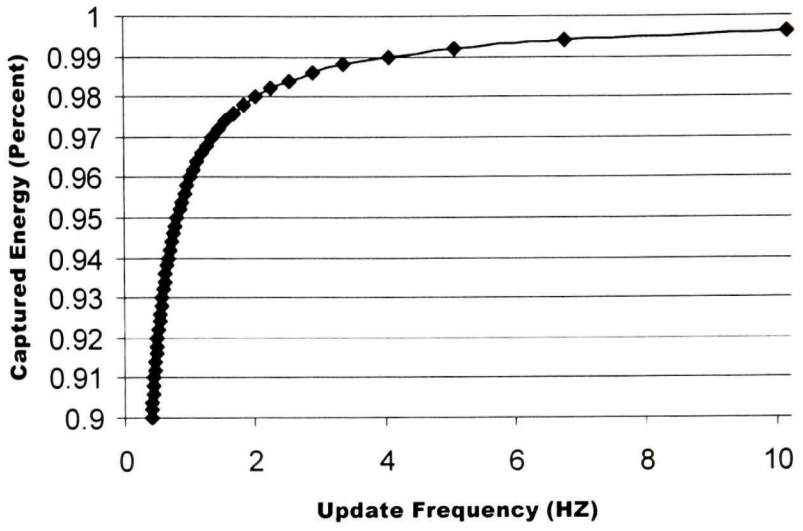


Figure 2.6.1.1: Captured energy and estimated update frequency

As shown in Figure 2.6.1.1, the captured energy is growing with the update frequency, but not linearly. After we have obtained 99% of the energy, it becomes increasingly costly to capture more. For example, to gain another 0.3% of the energy from 99.5%, the update frequency has to be more than doubled. Depending on specific traffic engineering goals, the sampling interval can be designed to capture around 99% of spectral energy.

Chapter 3

System Model

In this chapter, we identify the essential network functionalities and describe how they are modeled in the simulation.

3.1 Traffic Model

Traffic sources claim their service requirements at the ingress nodes of a network. The network then makes the decisions about admissions and resource allocations according to some SLAs. If a connection spans multiple networks, the negotiation needs to be carried in every network.

Some traffic sources contain a single packet, or very few packets. We call the traffic generated by these sources "datagram" traffic. Other traffic sources generate long series of packets over some time intervals. We call the traffic generated by these sources "connection" traffic. Datagram traffic can be viewed as connection traffic with very short holding time.
Connection traffic can be modeled with several characteristics:

- **Bandwidth requirement**

This can be either a static parameter negotiated in advance as part of an SLA, or negotiated upon admission. The bandwidth requirement of each traffic class is predefined with a specific effective bandwidth.

- **Origin in network**

This is fixed normally, and can be used as one of the parameters for assigning a label to the connection traffic.

- **Destination in network**

This can also be a fixed node, or can be determined by the network upon admission. This is also used for assigning a label to the connection traffic.

- **Holding time**

This may be a parameter provided by the traffic source (e.g., a 2-hour videoconference) or it can be inferred from other information associated with the traffic source (e.g., a TCP port).

- **Arrival rate**

This normally cannot be determined in advance. However, it can be estimated from traffic observations in similar conditions. Arrival rate and holding time together form the traffic intensity parameter in RUC.

For different connections, there can be lots of variations in the bandwidth and holding time. For instance, the holding time may last for days (e.g., VPN), hours (videoconference), minutes (VoIP call), or seconds (HTTP).

We use the notion of effective bandwidth [8] to define the bandwidth required by a connection. As described in chapter 2, the effective bandwidth of a connection provides a convenient measurement of its QoS requirement. The definition of effective bandwidth processes some desirable features that facilitate its utilization. We assume that, for a specified traffic type, e.g., VoIP, the value of the holding time and the arrival rate can be modeled as exponentially distributed random variables.

We assume that, in the network, the capacity of each link is large compared with the traffic generated by a single traffic source. In other words, each link may be

shared by a large number of concurrent traffic sources. This is a typical situation in telecom networks.

The traffic load may vary widely in different parts of the network and at different times during a day. This issue is discussed in section 3.3 in detail.

3.2 Routing Model

3.2.1 Route Selection

For datagram traffic, routing decisions apply on an individual packet basis.

For connection traffic, routing decisions apply on a connection basis. In this way, the packets of a connection are delivered along the same path in the network. Hence, packets arrive at their destination in the same order as they are sent. The label switching technique helps to satisfy this requirement. We assume MPLS is supported in the network.

The network that we study is a well connected and well engineered network. Many paths may exist between each pair of source and destination. Furthermore, the placements and the capacities of the links are closely associated with the direct traffic that they need to handle. In addition, there are several 2-link paths between each node pair in the network.

When a connection arrives at an edge node of the network, there are always several possible paths from the edge node to route it to its destination. Route selection is the process used by the node to make the decision.

The route selection process is state dependent. The result of this process may depend on the offered traffic and the load condition of the related links in the network. The central question that we want to investigate is the gain in performance that state-dependent route selection can provide, and the form that the underlying route selection algorithm should take.

Although each connection is tied to a particular path during its entire holding time, state dependent route selection can balance the load throughout the network. During its holding time, a connection won't be partitioned over multiple paths nor be rerouted. By considering resource allocation as a function of the resources currently available, unbalanced link utilization can be avoided.

3.2.2 Access Admission

For datagram traffic, admission decisions are made on an individual packet basis. These decisions depend on the origin and the destination of the packet, on its QoS requirements and on the state of the network.

For connection traffic, admission decisions are made on a connection basis. These decisions are made under an agreed traffic profile and QoS objectives. These decisions can be summarized as an effective bandwidth and service class required along some series of links. Once accepted, the network commits to respect the terms of agreement.

The cost of a traffic connection is evaluated at its ingress node or at some centralized routing control unit (e.g. bandwidth broker) in a network. If the cost of acceptance of this connection is greater than 1, the connection must be blocked.

We assume that the traffic connection source conforms to the traffic profile upon which it is accepted. This can be achieved through mechanisms such as traffic shaping and policing. If sources are allowed to increase their traffic beyond the level that the network agrees to support, the network cannot guarantee their QoS objectives.

If the network does not have enough resources to satisfy the QoS requirements of a connection, the connection must be refused.

3.2.3 Service Classes

QoS is provided by allocating a connection to a service class that is able to satisfy it. We are assuming that multiple service classes are supported by the network, for instance using DiffServ.
The different service classes can be viewed as independent of each other. Each class is engineered for a certain bandwidth. We assume that the bandwidth assigned to each service class is fixed. When we calculate the cost of a connection, we have to consider its effects on connections of all service classes, which may arrive later and request resources on the same link.

3.3 Traffic Distribution Model

To describe the traffic distribution model, let's use a typical continental U.S. IP network as an example. The traffic distribution model can be characterized by the following features:

- **Big nodes and small nodes**

The nodes in the network are normally located in different cities around the country. It is obvious that the bigger the city is, the more traffic it can generate, and the bigger the corresponding node is. The distribution of node sizes is just like that of city sizes. Huge nodes take up very a small percentage. The number of very small nodes is also relatively small. Most nodes are in the middle size class.

- **Same amount of in-out traffic**

We assume that the amount of traffic that is generated by a node and that is terminated at the node are in proportion. This is easy to justify. Assuming a person is sending out Christmas cards. The typical situation is that the more Christmas cards he sends out, the more he will receive.

- **Constantly changing traffic pattern**

The traffic pattern in telecom networks is strongly influenced by people's working timetable. During the weekend, the traffic volume is normally significantly lower than that in weekdays. In each weekday, the traffic volume usually increases dramatically in morning when people starts to work, and then drops down after working hours. At a given time, a node may be generating traffic with its peak rate, or with some percentage of its peak rate.

- **Non-coincidence of peak hours**

A network may span several different time zones, three for a continental U.S. network. Therefore, at a given time, some parts of the network may be generating traffic at their peak rates, while others not. For example, at 10 am Eastern Time on a weekday, New York will be at its peak rate, while Los Angeles will probably not be. The nodes in the network are allocated into different time zones; and they will be generating traffic according their local time.

- **Distance insensitivity**

Distance insensitivity is one of the properties of IP networks. Connections may be randomly distributed among the nodes in the network, although there may be a little more traffic aimed to local destinations than average.

- **Inter-network traffic**

We assume that there is no special limitation for connections to remain in one network. We consider several continental U.S. networks, and suppose that the data traffic has the same interest in going to each of these networks. Based on this assumption, the border nodes between the local network and other networks are going to handle large amount of incoming and outgoing traffic and become very big nodes.

3.4 Topology Model

The network topology should be built according to the traffic distribution, and satisfy the predefined design requirements. The networks that we study are well connected and well engineered, and can be described by three main characteristics:

- **Reasonable Size**

The size of a network is defined by its number of nodes. A too big network may take very long time to simulate; a too small network cannot generate useful result to help to analyze the performance of the routing algorithms. Considering that the backbone of a continental U.S. network typically consists of several dozens of nodes, a 30-node network is deemed sufficiently large to constitute a good topology for this research.

- **About Fifty Percent of Connectivity**

The connectivity of a network is defined as the ratio of its actual number of links related to the number of links that it would have if it was fully meshed. A fully meshed network is not necessary a good network. A well engineered network should provide committed quality of service and good scalability, and at the same time, minimize the building and maintenance cost. With the experience obtained from previous research, we believe that a network with about 50% connectivity should be a good starting point for our research.

- **About Ninety Percent of Traffic on Direct Links**

The network is built to carry as much traffic as possible on direct links. This is to avoid extremely unbalanced link utilization and lower resource utilization and maintenance cost.

- **One or Two Hop Path**

A small number of the connections in the network cannot reach their destinations through direct links. However, there are many two-link paths available for these connections. We assume that an optimal route can only be a one-hop path or a two-hop path. This assumption significantly reduces the complexity of the implementation of SDR mechanism.

A network with the features described above can be built according to the following principles.

- **There must be enough outgoing capacity from each node**

This means that the total outgoing capacity of a node has to be sufficient to carry the total outgoing traffic from this node determined in the traffic distribution model. The total outgoing capacity of a node is normally partitioned into several links that spread out from the node. The other ends of these links are also determined by the traffic distribution model. The bandwidth allocated to each of these links is, to a large extent, proportional to the volume of direct traffic on these links. The bigger the direct traffic is, the bigger the link capacity is.

- **There must be enough incoming capacity at each node**

This means that the total incoming capacity of each node has to be sufficient to support the total incoming traffic of this node determined in the traffic distribution model. The total incoming capacity of a node is normally partitioned into several links that terminate at the node. The other ends of these links are also determined by the traffic distribution model. The capacity allocated to each of these links is proportional to the volume of direct traffic on these links. The bigger the direct traffic is, the bigger the link capacity is.

Following the above procedures, we can build a well connected network, in which the placement and sizing of the links are highly correlated with the direct traffic that they need to support. With these procedures, we can satisfy two of the three

characteristics. However at this stage, we probably have a fully meshed network; all of the connections have direct paths to their destinations.

- **Connectivity Control**

We have to reduce the number of links to satisfy the fifty percent of connectivity. However, the links cannot be simply removed, because the total incoming or outgoing capacity of a node has to remain unchanged.

A simple way to solve the problem is:

1. Find out the number of incoming or outgoing links that a node can have;
2. Merge the surplus links into the remaining ones.

We must also guarantee that all origination - destination pairs have at least one 2-link path, if there is no direct link between them.

Chapter 4

Implementation

4.1 Overview

Based on NS2 [9], a simulator is built specially for this research. The operating system that we use for the simulation is FreeBSD 4.2, and the version of NS2 is 2.1b7a, which was the latest version available at the time.

- **Introduction to Network Simulator 2**

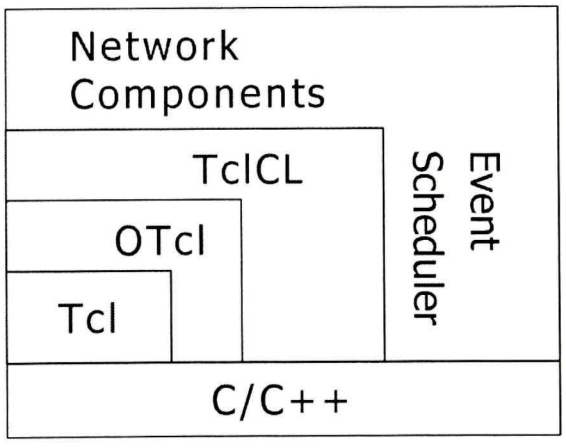


Figure 4.1.1: The Extended Tcl Interpreter of NS2

NS2 is an object-oriented simulator, written in C++, with an OTcl interpreter as the front-end.

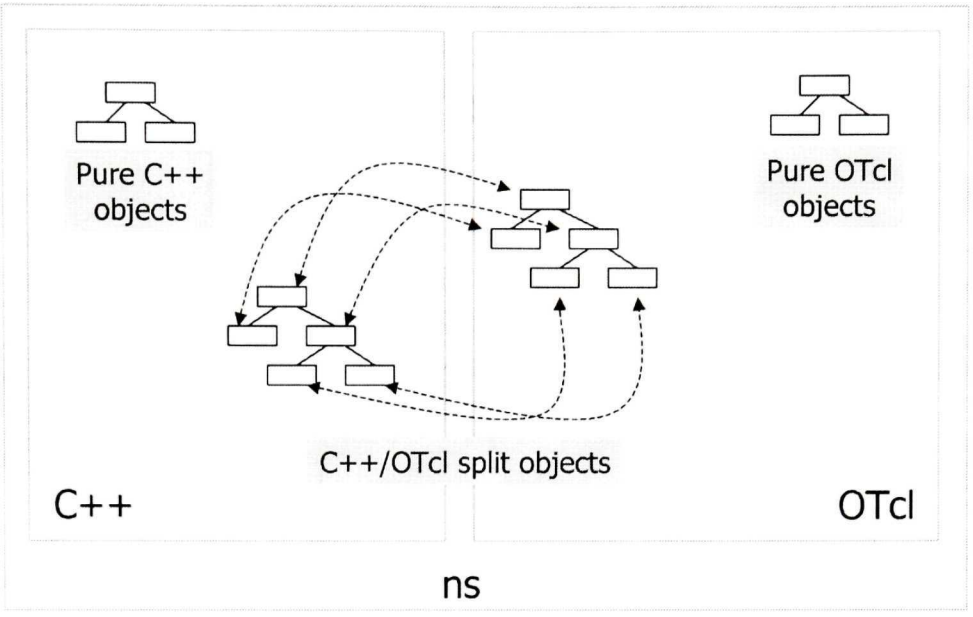


Figure 4.1.2: OTcl and C++: The Duality

The simulator supports a class hierarchy in C++ (also called the compiled hierarchy), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the *class TclObject*. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is automatically established through methods defined in the *class TclClass*. User instantiated objects are mirrored through methods defined in the *class TclObject*. There are other hierarchies in the C++ code and OTcl scripts; these other hierarchies are not mirrored in the manner of TclObject.

- **Concept Model**

In general, a network simulation can be divided into two parts: the applications and the network topology.

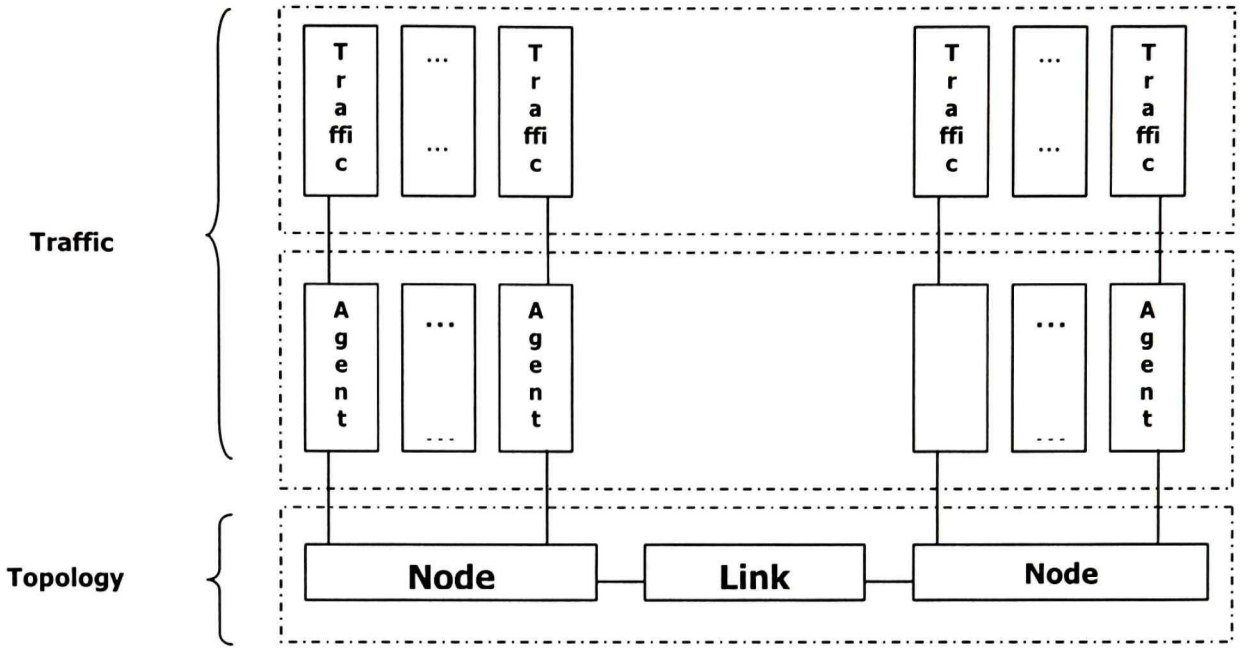


Figure 4.1.3: Concept model of a network simulation in NS2

- **Node**

A node is essentially a collection of classifiers and agents. In our case, a node contains one Sdo classifier and one port classifier. A Sdo classifier is a special classifier that is designed to simulate the explicit routing function of an LSR in this research.

Note that the words that are followed by under scores in the following figures are variables. These variables can be used as handles to access the corresponding fields in an object.

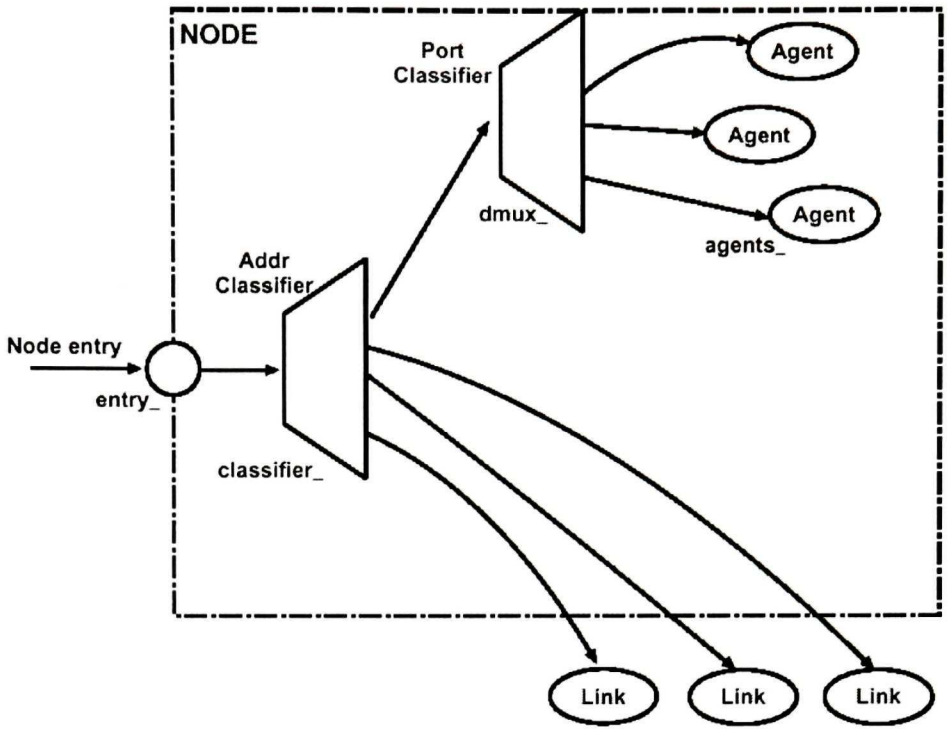


Figure 4.1.4: Structure of a Unicast Node

An Address Classifier reads IP addresses from packet headers and forwards the packets to next nodes or a local Port Classifier. A port Classifier forwards packets according to the destination port numbers in the packet headers.

- **Link**

A link is used to connect two nodes. A unidirectional link is called a simplex link in NS2. A simplex link connects a slot of a classifier to a corresponding node entry.

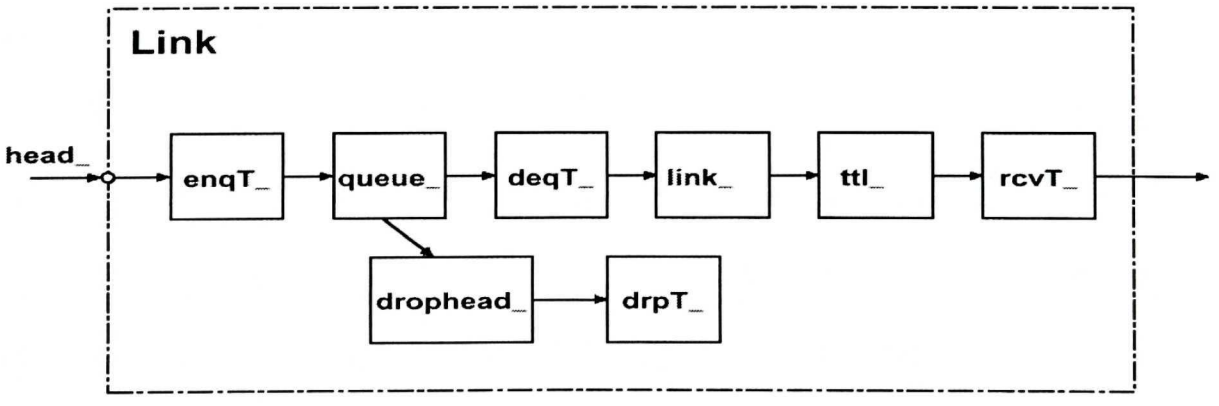


Figure 4.1.5: Structure of a unidirectional link

- **Topology**

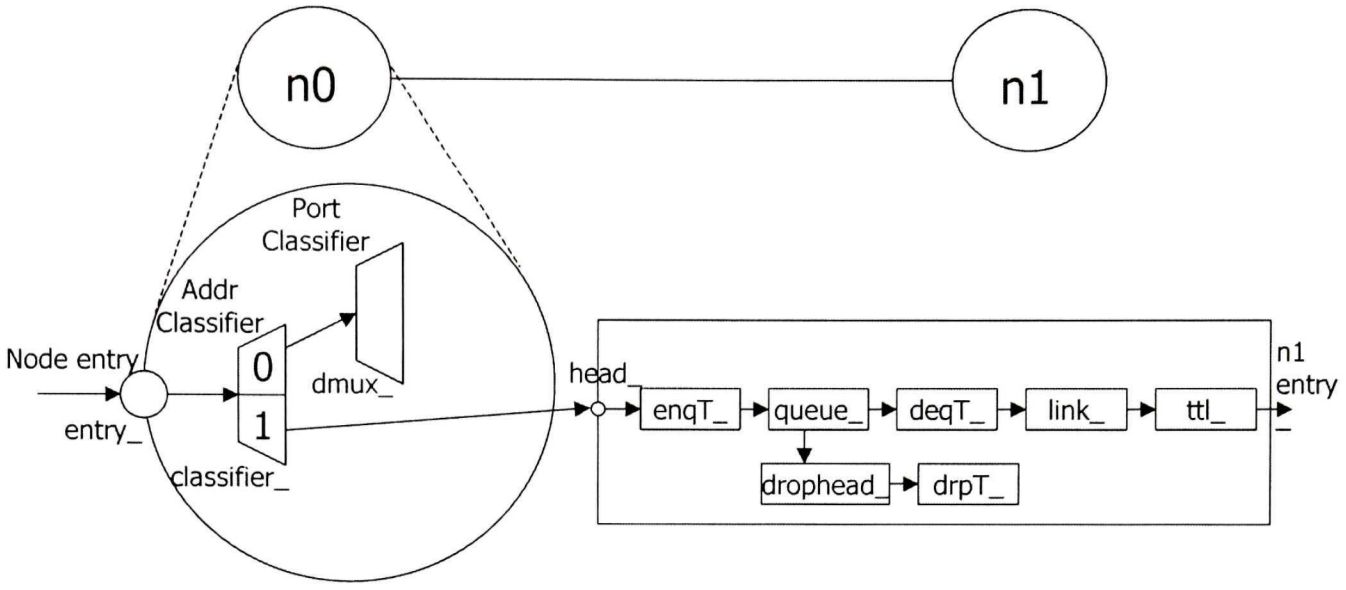


Figure 4.1.6: Unidirectional connection sample

- **Routing**

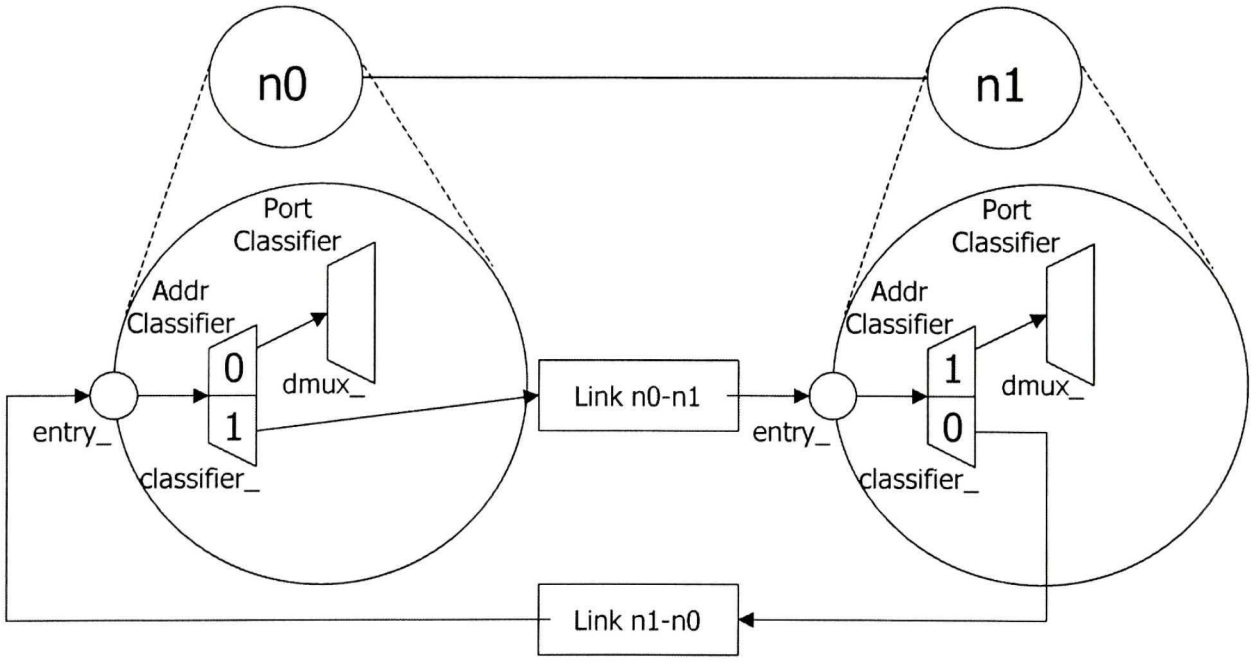


Figure 4.1.7: Duplex connection sample

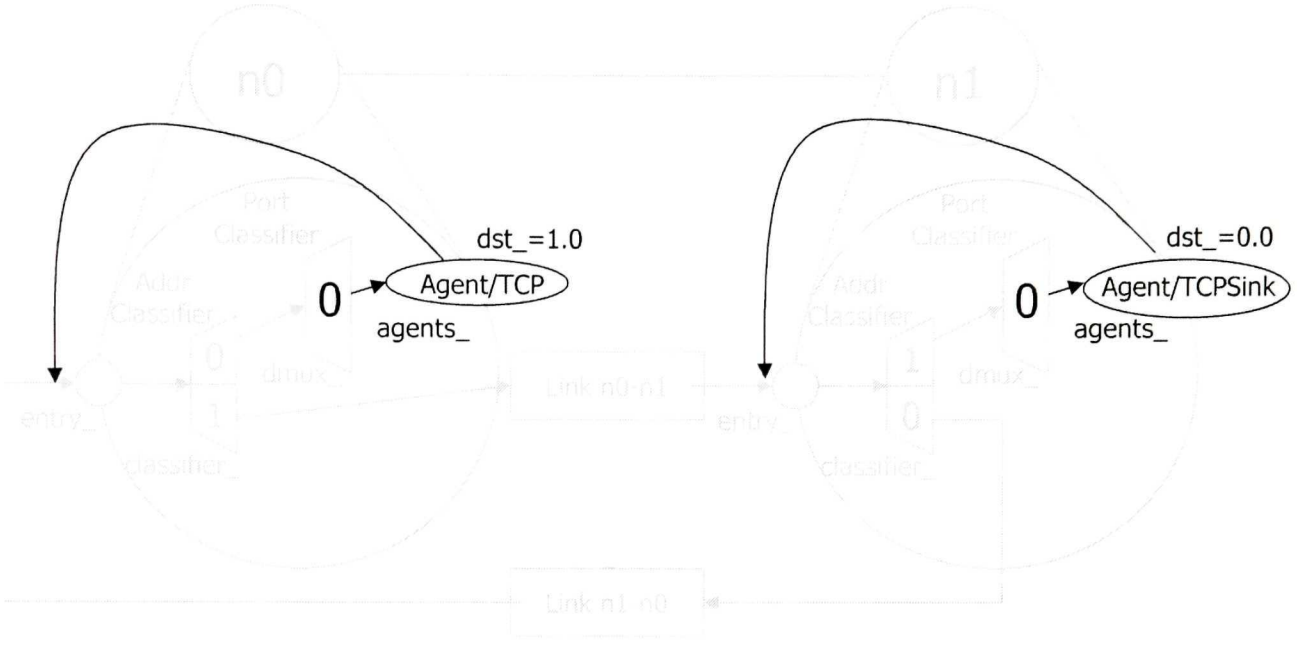


Figure 4.1.8: Adding transport agents

Transport agents act as the transport layer in the OSI 7-layer model. Both TCP and UDP agents are available. When a transport agent sends out a packet to its neighbor, the packet will go directly to the node entry (the variable "entry_" in the node object), as shown by the arrow; then the Address Classifier will forward it to the corresponding Link Head (the variable "head_" in the link object).

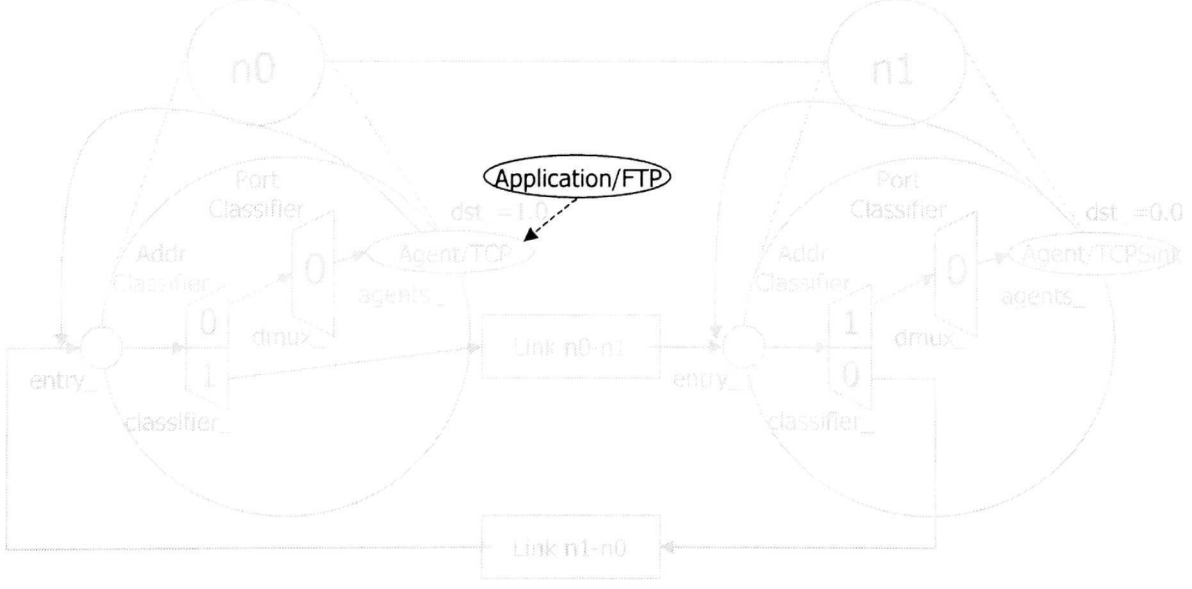


Figure 4.1.9: Adding a traffic generator

Traffic generators are applications at layer 5 and above in the OSI 7-layer model. Traffic generators must be attached to transport agents in order to send out packets. In figure 4.1.9, an application (FTP) is attached to a TCP agent. Node 0 is ready to send packet to the TCP Sink in node 1. A Sink is a null agent, which is responsible for destroying accepted packets.

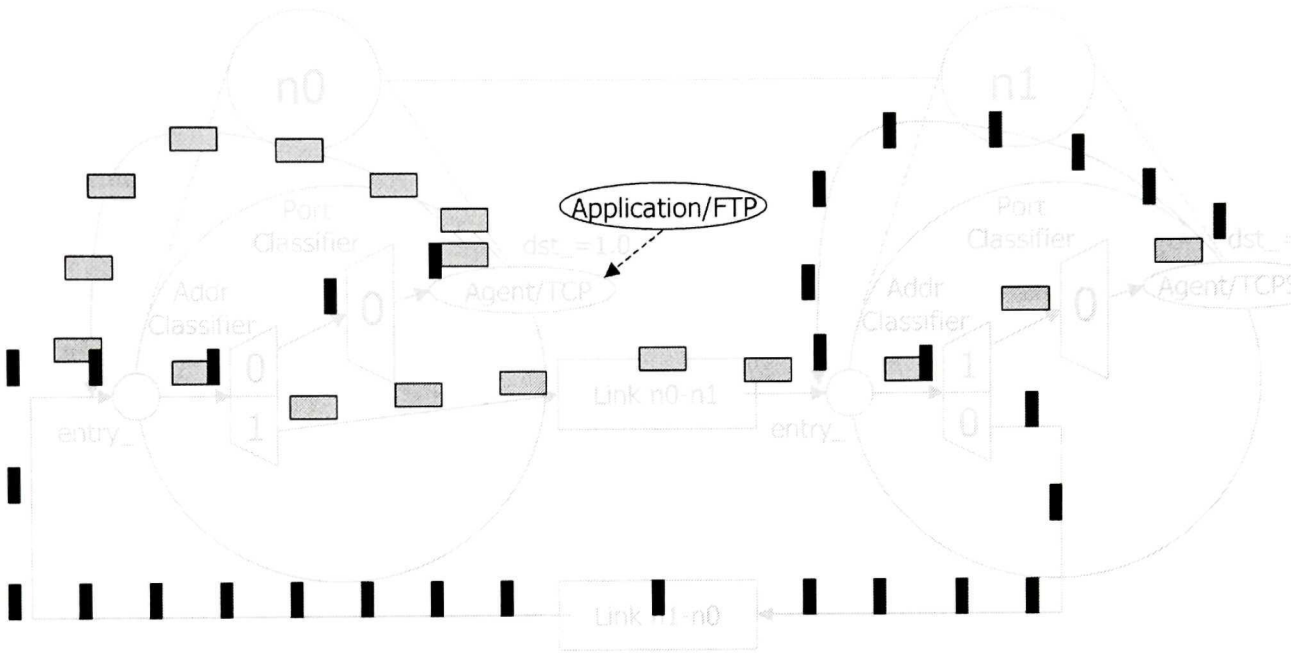


Figure 4.1.10: Packet flow

This figure shows how packets flow between two nodes. The gray packets are generated by the FTP application in node 0 (n0); and the black packets are returned from node 1 (n1) to node 0 (n0). The packets generated by FTP are passed to its TCP agent and then delivered to the entry of node 0. The address classifier forwards packets to corresponding slots according to the destination addresses in the packet headers. The packets generated by FTP are forwarded to slot 1 that points to node 1. When the packets arrive at the entry of node 1, the address classifier of node 1 again will check the destination addresses in the packet headers, and forward the packets to slot 1 that points to the port classifier of node 1. Finally, the packets are forwarded to the TCP Sink in node 1. The packets returned from the TCP Sink of node 1 experience the same procedure to reach the TCP agent of the node 0.

4.2 Traffic Agents

Agents represent endpoints where network layer packets are constructed or consumed and are used in the implementations of protocols at various layers. The *class Agent* has an implementation partly in OTcl and partly in C++.

Applications sit on top of transport agents, such as UDP agents and TCP agents, in NS2, as shown in Figure 4.2.1. There are two basic types of applications: traffic generators and simulated applications. The one on the left is used to generate UDP traffic. The Exponential On/Off generator is one of the applications in the UDP class. The other one is used to simulate TCP application traffic, such as FTP and HTTP.

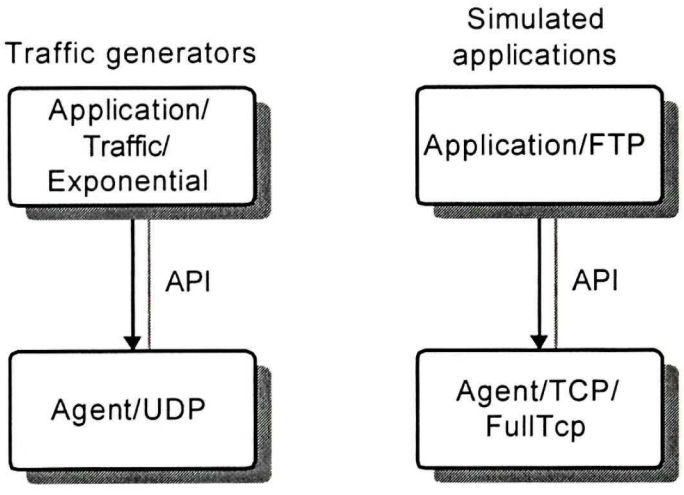


Figure 4.2.1: Example of Application Composition

To obtain valid simulation results, the generators should be capable of generating traffic with exponentially distributed burst time, holding time and bandwidth.

The Exponential On/Off generator is used as the traffic generator in this simulation. The output bandwidth of a traffic generator is specified by an exponential random number generator with a given mean value. Packets are sent at predefined rate during on periods, and no packets are sent during off periods.

The lengths of both burst time and holding time are drawn from exponential random distributions.

With the effective bandwidth model, it is not necessary to simulate packets. However, with the packet structure, we can easily store routing information, such as flow label, into packet header, which facilitates our simulations. All data flows in our simulations consist of packets.

4.3 Label-Switching

The Address Classifier shown in section 4.1 can only forward packets based on IP addresses. Hence, a node with only Address Classifiers acts like a conventional IP router that makes routing decisions on a hop-by-hop basis and provides best-effort delivery service. To support the label-switching technique, we need to devise a new classifier that can make routing decisions based on flow or label information. The Sdo Classifier is designed to perform label-switching functions. Before introducing the Sdo Classifier, let's take a closer look at Classifiers in general.

- **Classifiers**

The function of a node when it receives a packet is to examine the packet's header fields, usually its destination address and on some occasions, its source address. It should then map the value(s) to an outgoing interface object that is the downstream recipient of this packet.

In NS, a simple classifier object performs this task. Multiple classifier objects, each looking at a specific portion of the packet header, forward the packet through the node. A node uses many different types of classifiers for different purposes.

A classifier provides a way to match a packet against some logical criteria and to retrieve a reference to another simulation object based on the match result. Each

classifier contains a table of simulation objects indexed by *slot numbers*. The job of a classifier is to determine the slot number associated with a received packet and to forward that packet to the object referenced by that particular slot.

- **Hash Classifier**

One of the classifiers is particularly interesting, the Hash Classifier. A Hash Classifier can classify a packet as a member of a particular *flow*. As indicated by its name, a Hash Classifier uses a hash table internally to assign packets to flows. This object is used where flow-level information is required (e.g. in flow-specific queuing disciplines and statistics collection). Several "flow granularities" are available. In particular, a packet may be assigned to a flow based on its flow ID, its destination address, its source and destination addresses, or the combination of its source and destination addresses plus its flow ID.

The functions of the Hash Classifier make the label distribution and label switching processes much easier.

- **Sdo Classifier**

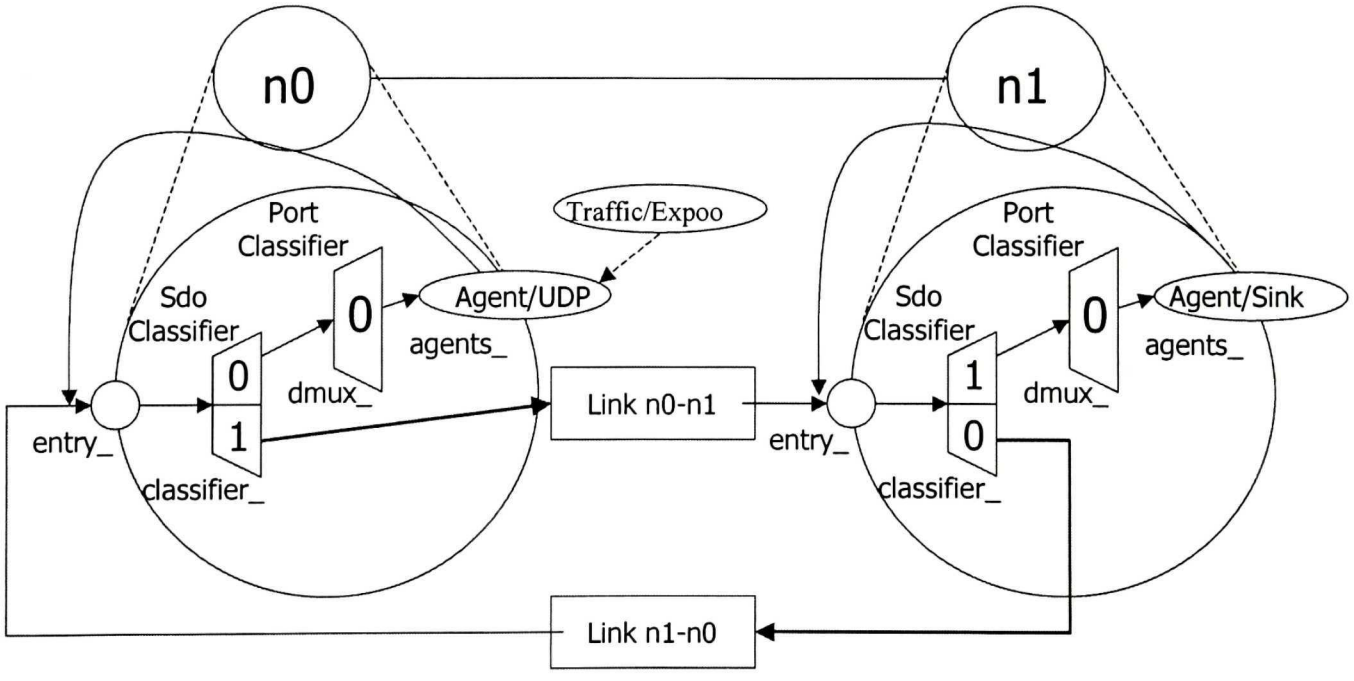


Figure 4.3.1: Sample system structure of SDOR nodes

Each of the classifiers that are original from NS2 maintains a slot table, which contains pointers to all of the node objects in the network. If the number of nodes in a simulation grows n times, the memory used for slot tables will grow proportionally to n^2. While in an SDO Classifier, the slot table contains only references to the node objects that are directly connected to the node, in which the SDO Classifier resides. The example shown in Figure 4.3.2 is an Sdo Classifier with five neighbor nodes. The Hash table maintains all of the flows that go through this node. The slot table contains five pointers to its five neighbors and the other one is to itself.

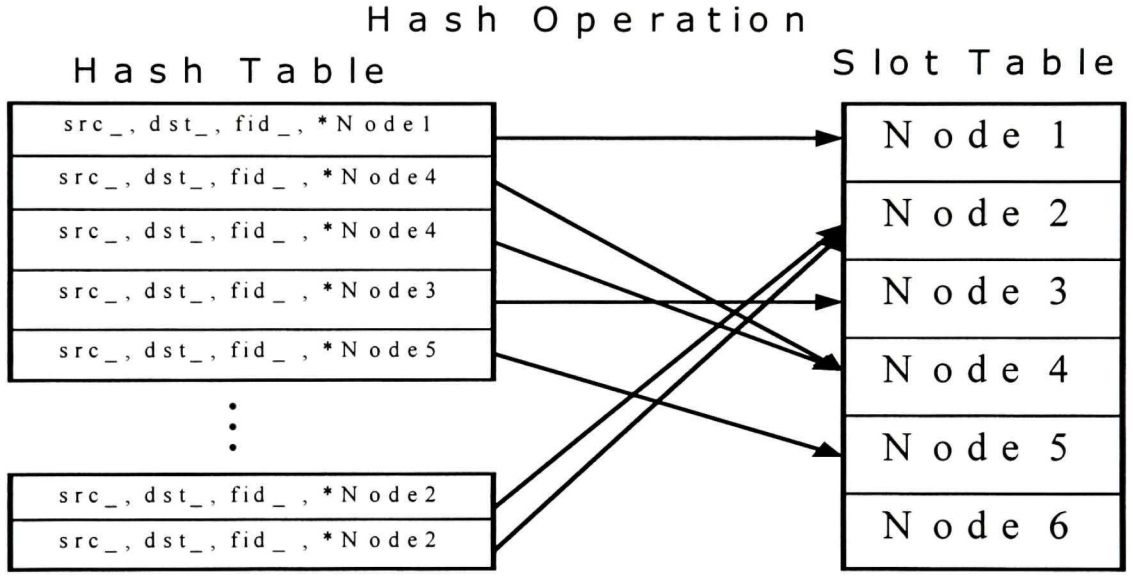


Figure 4.3.2: Hash Operation of an Sdo Classifier with five neighbor nodes

At the beginning of a simulation, each Sdo Classifier will look for directly connected nodes and install the corresponding entries in its slot table.

- **Establishing and Removing Label Paths**

With all of the above functions ready, the label path manipulation becomes a simple procedure.

Once a route is selected for a connection, the nodes along the route just add a new entry for this flow to their hash tables. A label path is built for this connection. Then, all the packets from this connection will be marked with a specific flow ID

and forwarded along the predefined label path. Similarly, when a connection terminates, the nodes on the path remove the hash entry from their hash tables. The label path is removed.

4.4 State-Dependent Optimal Routing

The State-Dependent Optimal Routing is implemented in two mechanisms: centralized and decentralized. MPLS also supports static multipath routing to avoid computing and setting routes up in real time. However, this implementation is left for further studies.

4.4.1 Centralized Routing Control (CRC) overview

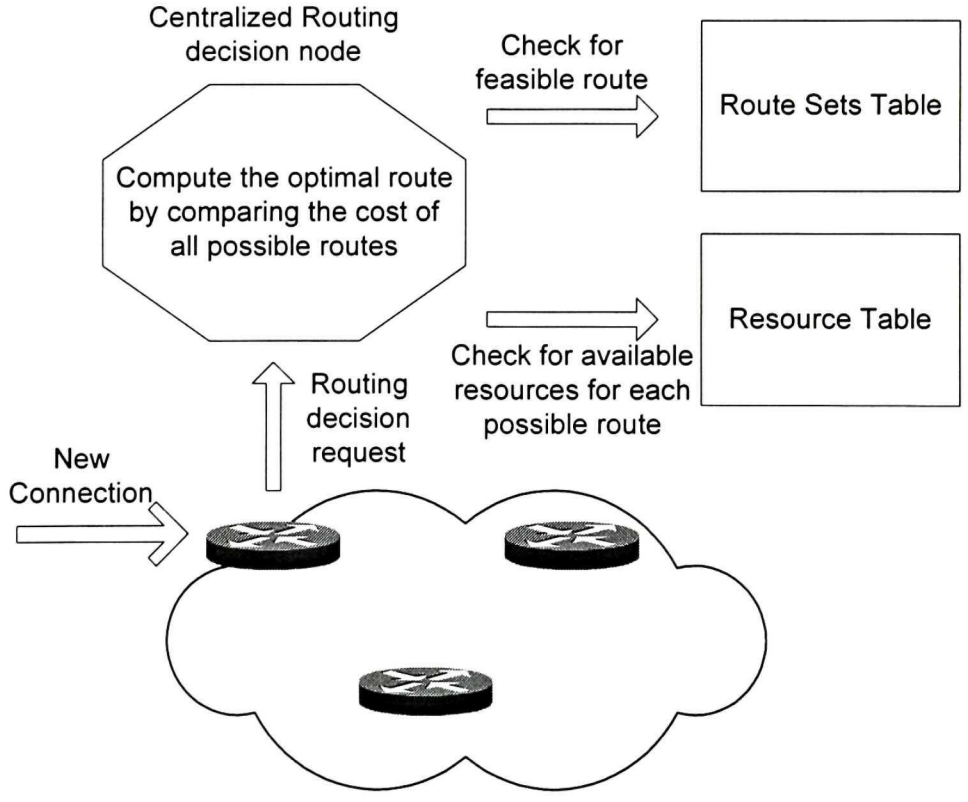


Figure 4.4.1.1: Centralized routing computation

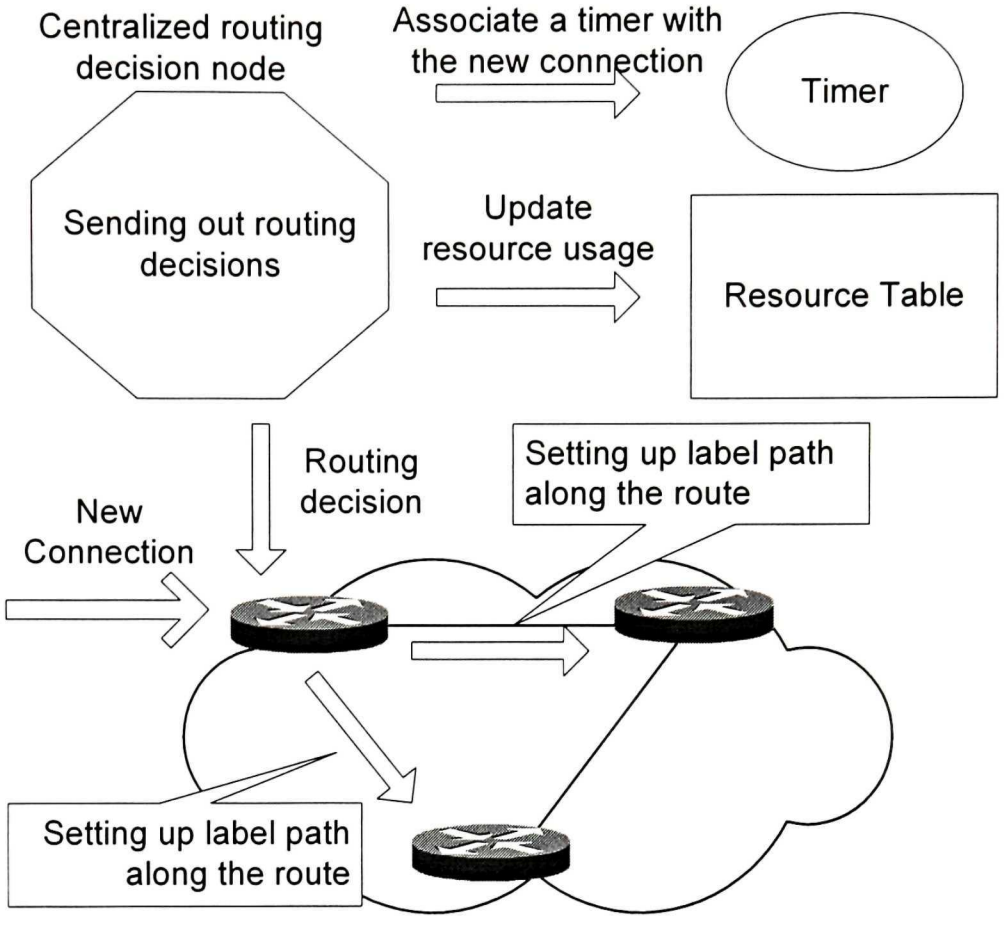


Figure 4.4.1.2: Label distribution and resource monitoring processes of CRC

CRC employs the DF routing policy and uses TR to calculate the cost of a connection. Whenever a new connection arrives, the ingress node contacts the centralized routing decision node (CRDN). The CRDN finds out all of the feasible routes for this connection from a route set table, then checks a resource table to determine the utilization of the concerned links, and computes the costs associated with these routes, and finally, selects the minimum cost route. The CRDN then provides the ingress node with the optimal route. The resource table is updated accordingly; a label path is established along the route; and a timer is associated with the connection. We are going to study the timer in detail later this section.

If there is no route that has enough resources, the connection will be blocked.

4.4.2 Decentralized Routing Control (DRC) overview

DRC is an extension of CRC with no assumption of real time link utilization information. It also employs DF and TR. In the decentralized implementation, each node holds a local resource table that is updated periodically by the CRDN. An ingress node makes routing decisions according to its local resource table without contacting the CRDN. Those routing decisions may not be globally optimal since the local resource tables do not contain real time link utilization information. The longer the resource table update interval is, the lower the possibility that an ingress node can make optimal routing decisions is. However, the blocking rate of a network is not a linear function of the update frequency. If we consider the utilization of the links of the network as stochastic processes, the periodic update is a sampling process. If our sampling is frequent enough, we should be able to capture most of the spectral energy of the process, and have an accurate tracking (as discussed in section 2.6). After that, increasing the frequency further will not improve much the performance, because we already capture most of the energy of the process (Figure 2.6.1.1). DRC implements its update procedure according the sampling analysis introduced in section 2.6.
With DRC, CRDN is not responsible for route computations; it only sends out link utilization updates. The DRC does not depend on real time information. This reduces significantly the computation load and signaling traffic of CRDN, and hence, makes the SDOR more practical.

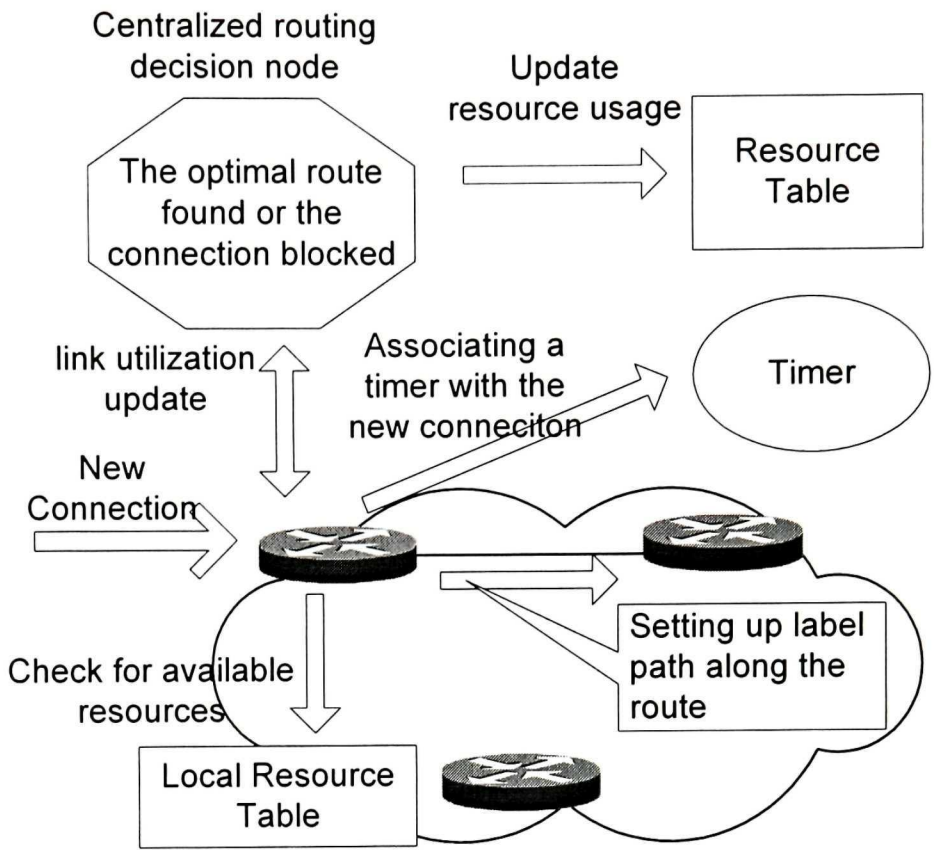


Figure 4.4.2.1: Decentralized routing control for direct traffic

4.4.3 The Route Set Table

A route set is a set of routes that contain all of the eligible routes between a pair of source and destination nodes.

This table is a three-dimensional matrix that contains the detailed topology information. Considering a 30-node network, the matrix can be $[30 \times 30 \times 30]$. Recall that all the links in this network are unidirectional, which means that the link from node i to node j is different from the one from node j to node i. The first dimension is used to contain the source node-IDs, the second one is for destination node-IDs, and the third one holds the intermediate node-IDs of all possible 2-hop routes. If the source and the destination are directly connected, the third dimension of the matrix will be filled with the destination node-ID as one of the routes.

The route set table is different from a traditional routing table that only records one feasible route for each pair of source and destination.

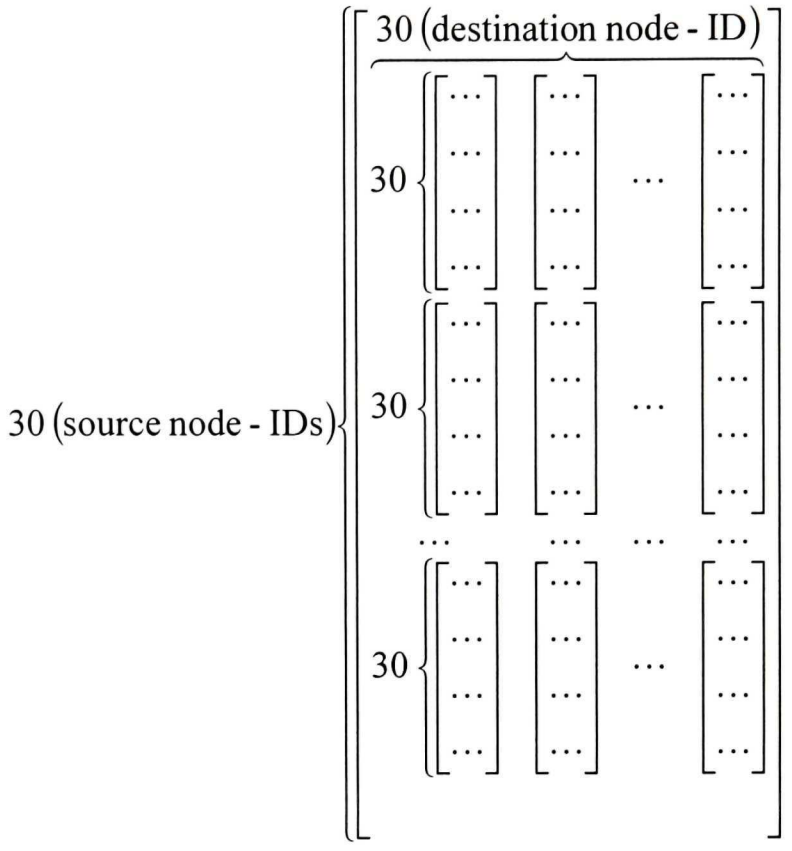


Figure 4.4.3.1: The structure of the route set table

4.4.4 The Resource Table

The resource table is used to keep the link utilization information of the network. For a 30-node network, the resource table will be a 30 by 30 matrix. In this implementation, each item in the matrix contains two numbers, the total link capacity (units of effective bandwidth) and the used capacity (units of effective bandwidth).

To make sure that the routing decisions are always optimal, the resource table has to be updated timely. In another word, whenever a label path is set up or removed, the resource table has to be modified. This is done by the cooperation of the Sdo Classifiers, the CRDN and the Timer objects.

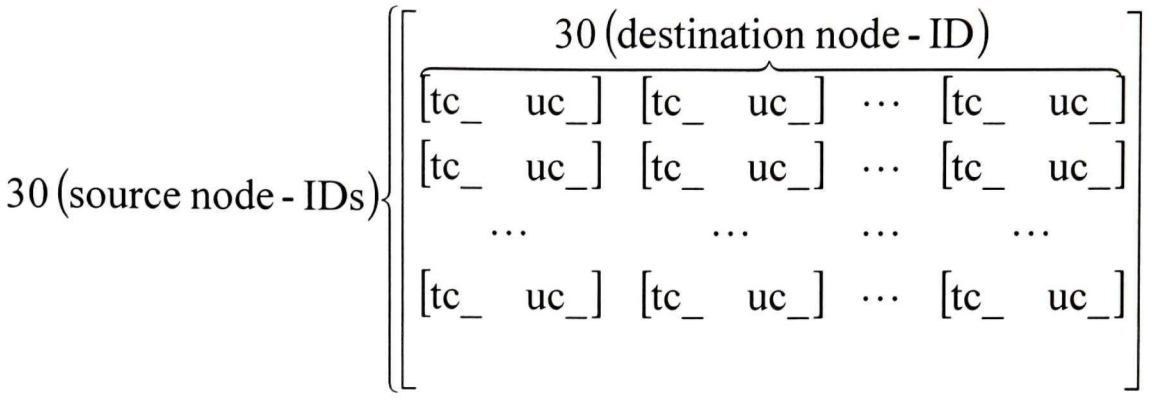


Figure 4.4.4.1: The structure of the resource table

4.4.5 The Timer

A timer is a tool to schedule an event, which will happen after a specified delay. In fact, when a user schedules an event with a timer, the timer inserts the event into the schedule list of the simulator. In addition, a timer object supports "cancel", and "reschedule" operations.

A Timer helps to monitor the status of a connection and to maintain the label path for it.

The duration of a connection cannot be predicted accurately. In this event-driven simulator, objects can only respond to events that have already happened or been triggered. For instance, a traffic generator starts to send out packets. Upon this event, all of the objects which have noticed this event can respond with some actions. However, a traffic generator stopping sending packets is not an event but the termination of an event. Hence, the simulator has no way to capture this exact moment. A timer object is used to capture this moment with limited error.

An IP data flow consists of a series of packets. When the first packet of this flow arrives, a Timer is attached to it and is started. Whenever a new packet of this flow arrives, the attached Timer is rescheduled for a predefined interval. If the Timer expires, the connection is considered over.

Therefore, a Timer is necessary for each IP flow. Every new packet is going to trigger a restarting of the Timer attached with its flow. To reduce the transactions

associated with rescheduling Timers, a Timer can be restarted for every three or more packets. The drawback is that the number of arrived packets since the last rescheduling has to be kept for every IP flow.

Although the interval between two consecutive packets in a flow is not constant (depending on the traffic rate and the packet size), the packet interval is always much smaller than the expiring interval of its timer. Hence, the Timer will never expire if the flow is still on.

However, this method does induce error. The biggest error between a timer expiring and a flow ending is the expiring interval of the timer. It is easy to keep the error in an acceptable range.

$$\begin{cases} Error < I_E & \text{Timer is rescheduled every packet} \\ Error < I_E - d \times I_P & \text{Timer is rescheduled every } x \text{ packets} \end{cases}$$

$x \times I_p << I_E$

I_E : the expiring interval of timers

I_P : the packet interval

d : Number of packets since the last reschedule

Due to the limitation of the simulator, a timer has to be used to catch the termination moment of a flow. This could be a scalability problem since the number of timers is linear in the number of flows. If the edge nodes could detect the event, the implementation would be more scalable.

4.5 Traffic Distribution and Topology

The objective of this section is to create a reasonable network for the purpose of assessing the performance of the routing mechanisms.

The following is a summary of the simulated continental U.S. network. Detailed descriptions of the reasoning and algorithms that are behind the results are presented in Appendix A.

4.5.1 Topology Generating Process

- **Node**

There are 30 nodes in the network. According to their abilities in originating traffic, the nodes can be divided into 4 scales or sizes (small, medium, large and very large). A node is assigned to a specific scale according to the following table.

Nodes size	Originating capability	Prob. of occurrence
Small	0.5	0.3
Medium	1	0.4
Large	1.5	0.2
Very large	2	0.1

Table 4.5.1.1: Node size and distribution

The network spans 3 time zones (Eastern, Central, and Western). There are 10 nodes in each time zone. For a given time, a node in the Eastern Time zone may be in its peak rate; a node in Western Time zone may be in its off peak rate. We use the following table to describe the traffic levels that a node may originate.

Traffic level	Fraction of peak traffic
Peak hour	1.0
Side peak hour	0.85
Off peak hour	0.70

Table 4.5.1.2: Traffic levels

This table presents the traffic levels of the nodes as a function of their time zone and of the time of the day, expressed in Eastern Standard Time (EST).

Hour	Region of the US where the node is located		
	East	Central	West
10 AM EST	Peak hour	Side peak hour	Off peak hour
12 PM EST	Side peak hour	Peak hour	Side peak hour
1 PM EST	Side peak hour	Side peak hour	Peak hour

Table 4.5.1.3: Peak hours

- **Border node**

The network is connected to three foreign networks. We assume that the three foreign networks have the same scale and throughput as our network. To each of the foreign networks, our network has two connections through two border nodes. Network traffic is assumed evenly distributed among the four networks. All of the inter-network traffic, which takes up 75% of the total volume, transmits through the six border nodes.

- **Traffic generator**

Traffic generators are distributed according to node sizes and peak hours. The bigger a node's originating traffic is; the more traffic generators are assigned to it. For example, a node may have 100 traffic generators at its peak hour and only have 85 traffic generators at its side peak hour. The following lists the total number of generators in the network in different traffic patterns.

Traffic Pattern	Number of Generators
AM (10 AM EST)	4856
Noon (12 PM EST)	5023
PM (1 PM EST)	5242

Table 4.5.1.4: Number of generators

- **Link**

The network has a connectivity of 49.2% and comprises 428 simplex links. Links are allocated according to traffic demand. About 90% of traffic can reach its destination in one hop; every pair of nodes in the network has several two hop routes available.

4.5.2 The Generated Topology Matrix and Traffic Matrix

The following is a topology matrix that is generated with the above process. The first column lists the originations and the first row lists the destinations.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	69	81	14	14	7	7	14	14	29	14	14	21	7	7	13	13	7	13	6	78	75	69	22	16	10	5	10	10	15
1	58	0	83	15	15	7	7	14	14	29	14	14	21	7	7	14	13	7	13	7	78	74	69	21	16	10	5	10	10	15
2	83	80	0	19	19	9	9	18	18	36	17	17	25	8	8	16	16	8	16	8	91	86	78	24	18	12	6	11	11	17
3	10	12	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	26	13	11	0	0	0	0	11
4	11	12	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	26	13	11	0	0	0	0	11
5	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	11	0	0	0	0	0	11
6	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	11	0	0	0	0	0	11
7	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
8	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
9	19	22	28	0	0	0	0	0	0	0	8	11	0	0	9	9	0	9	0	36	37	37	14	11	8	0	8	8	11
10	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
11	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
12	14	16	21	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	29	31	31	13	11	9	0	9	9	11
13	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	11	0	0	0	0	0	11
14	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	11	0	0	0	0	0	11
15	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
16	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
17	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	12	0	0	0	0	0	11
18	11	12	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	24	25	13	11	0	0	0	0	11
19	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	12	0	0	0	0	0	11
20	93	84	91	17	16	8	8	16	16	31	15	15	22	7	7	14	14	7	14	7	0	79	72	23	17	11	5	11	11	16
21	94	83	90	16	16	8	8	16	15	31	15	15	22	7	7	14	14	7	14	7	76	0	71	23	17	11	5	11	11	16
22	91	80	86	16	16	8	8	15	15	30	14	14	21	7	7	14	13	7	13	6	73	70	0	22	16	11	5	10	10	15
23	22	23	28	0	0	0	0	0	14	8	8	11	0	0	8	9	0	9	0	35	36	36	0	11	8	0	8	0	11
24	16	17	21	0	0	0	0	0	12	0	0	10	0	0	0	0	0	0	0	28	30	30	13	0	9	0	9	9	11
25	12	13	15	0	0	0	0	0	5	0	0	5	0	0	0	0	0	0	0	22	23	25	12	11	0	0	0	0	11
26	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	18	12	0	0	0	0	0	11
27	12	13	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	23	25	12	11	0	0	0	0	11
28	12	13	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	23	25	12	11	0	0	0	0	11
29	16	17	21	0	0	0	0	0	0	12	0	0	10	0	0	0	0	0	0	0	28	30	30	13	11	9	0	9	9	0

Table 4.5.2.1: Sample Topology Matrix

The unit of the numbers in table 4.5.2.1 is the effective bandwidth. The effective bandwidth of connections in our simulation is 20K.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0		13	26	13	13	7	7	13	13	26	11	11	17	6	6	11	11	6	11	6	1	14	14	19	14	10	5	10	10	14
1	14		27	13	13	7	7	13	13	26	11	11	17	6	6	11	11	6	11	6	14	1	14	19	14	10	5	10	10	14
2	28	28		14	14	7	7	14	14	28	12	12	18	6	6	12	12	6	12	6	24	24	1	20	15	10	5	10	10	15
3	13	13	14		1	1	1	1	1	2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1	1	1
4	13	13	14	1		1	1	1	1	2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1	1	1
5	7	7	7	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	1
6	7	7	7	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	1
7	13	13	14	1	1	1	1		1	2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1	1	1
8	13	13	14	1	1	1	1	1		2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1	1	1
9	26	26	28	2	2	1	1	2	2		2	2	3	1	1	2	2	1	2	1	19	19	19	3	2	2	1	2	2	2
10	13	13	14	1	1	1	1	1	1	2		1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1	1	1
11	13	13	14	1	1	1	1	1	1	2	1		2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1	1	1
12	20	20	21	2	2	1	1	2	2	3	2	2		1	1	2	2	1	2	1	15	15	15	2	2	1	1	1	1	2
13	7	7	7	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	1
14	7	7	7	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	5	5	5	1	1	1	1	1	1	1
15	13	13	14	1	1	1	1	1	1	2	1	1	2	1	1		1	1	1	1	10	10	10	2	1	1	1	1	1	1
16	13	13	14	1	1	1	1	1	1	2	1	1	2	1	1	1		1	1	1	10	10	10	2	1	1	1	1	1	1
17	7	7	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	5	5	5	1	1	1	1	1	1	1
18	13	13	14	1	1	1	1	1	1	2	1	1	2	1	1	1	1	1		1	10	10	10	2	1	1	1	1	1	1
19	7	7	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		5	5	5	1	1	1	1	1	1	1
20	1	47	61	14	14	7	7	14	14	27	12	12	18	6	6	12	12	6	12	6		38	38	19	15	10	5	10	10	15
21	52	1	66	14	14	7	7	14	14	27	12	12	18	6	6	12	12	6	12	6	41		41	19	15	10	5	10	10	15
22	56	56	2	14	14	7	7	14	14	27	12	12	18	6	6	12	12	6	12	6	44	44		19	15	10	5	10	10	15
23	26	26	28	2	2	1	1	2	2	4	2	2	3	1	1	2	2	1	2	1	19	19	19		2	2	1	2	2	2
24	20	20	21	2	2	1	1	2	2	3	2	2	2	1	1	2	2	1	2	1	15	15	15	2		1	1	1	1	2
25	13	13	14	1	1	1	1	1	1	2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1		1	1	1	1
26	7	7	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5	1	1	1		1	1	1
27	13	13	14	1	1	1	1	1	1	2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1		1	1
28	13	13	14	1	1	1	1	1	1	2	1	1	2	1	1	1	1	1	1	1	10	10	10	2	1	1	1	1		1
29	20	20	21	2	2	1	1	2	2	3	2	2	2	1	1	2	2	1	2	1	15	15	15	2	2	1	1	1	1	

Table 4.5.2.2: Traffic Matrix in the AM Pattern

Table 4.5.2.2 shows the traffic generators distribution for the AM pattern in the network.

Chapter 5

Results and Analysis

In this chapter, we are going to demonstrate the proposed routing mechanisms through presenting and analyzing the simulation results. Let us still use the continental U.S. network as an example to explain our findings.

5.1 Simulated Network Environment

We simulate a 30-node network and assume that another three equivalent networks coexist with it. The network has 3 pairs of gateway nodes that are linked to its 3 neighboring networks.

In this research, we focus on homogenous IP traffic, which can be characterized by a single set of traffic intensity and bandwidth parameters. These parameters are implemented in traffic generators that inject packets from nodes to the network. Each generator will produce traffic flows with exponentially distributed arrivals and holding times, with mean value $1/7$ HZ, and 5 seconds respectively. Thus, on each link, the call arrivals can be seen as independent Poisson process. If we consider the effective bandwidth of connections as unit, each traffic generator corresponds to $5/7 = 0.714$ Erlang.

Three different traffic patterns, the AM, noon and PM patterns, which are introduced in section 4.5.1, are simulated in the network. These three patterns offer similar traffic loads to the network. The AM pattern is the least loaded one among them; the PM pattern produces slightly more traffic than the other two. In

each traffic pattern, a specific number of traffic generators are installed in the network. The column "No. of Flows" in Table 5.1.1 lists the total number of connection requests that are offered to the network.

Traffic Pattern	No. of Generators	No. of Flows	Tot. Load (Erlang)
AM	4856	26321	3467
Noon	5024	27272	3587
PM	5242	28433	3743

Table 5.1.1: Offered load

It shows that only 7% more traffic is offered in the PM pattern than in the AM pattern. However, as presented in Table 4.5.1.3, these three traffic patterns have different traffic distributions. This provides us a chance to test our routing mechanisms under different load conditions.

Using effective bandwidth as unit, the total network capacity is 7755 units. The average outgoing capacity of gateway nodes is 720 units, and is 143 units for interior nodes. Gateway nodes are normally much more heavily loaded than interior nodes, since inter-network traffic make up ¾ of the total traffic. Therefore, more links and bandwidth are allocated to gateway nodes (see figure 4.5.2.1).

5.2 The Performance of SDR-ADAPT

This section presents the simulation results of SDR-ADAPT. SDR-ADAPT does not assume prior knowledge of the offered traffic; instead, it collects the load information during its operation and updates its measured traffic intensity periodically.

5.2.1 Determining the h and Δ

The update procedure is presented in Figure 2.5.1.1 with $\delta = 1$ second $\Delta = 81$ seconds and $h = 5$ seconds . The parameters h and δ are selected according to the simulated network scale and available hardware resources. Δ is picked by observing the behavior of the network under fixed traffic intensity matrices.

Figure 5.2.1.1 shows an example of the blocking curve under a randomly picked traffic intensity matrix. The blocking rate is calculated every second from the number of blocked connection requests and total requests in that second. From the 60th second to the 100th second, the blocking rate remains in between 2.30% and 2.50%. We can say that the curve is already in its steady state before the 80th second. Normally, a blocking curve should enter its steady state after ten times the mean holding time of connections, which is 50 seconds in this simulation. In addition, we also want to collect some data during the steady state period. Hence, we set the traffic intensity update period, Δ, to 81 seconds.

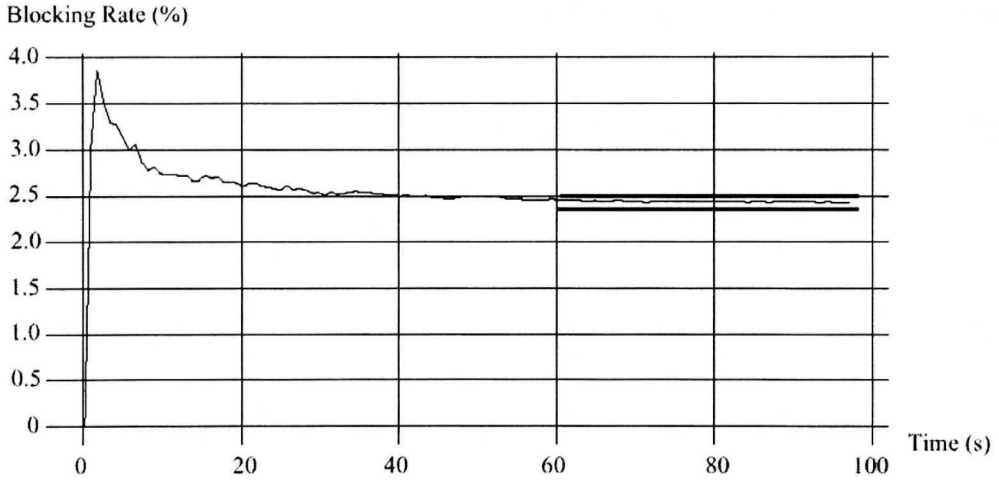


Figure 5.2.1.1: The blocking rate curve with a fixed intensity matrix

From the above example, we can see that the selection of Δ is affected by the mean holding time, h. When h goes up, it will be more costly to simulate every flow; and the update cycle, Δ, will also be longer. To make the simulations less time consuming, we pick a rather short mean holding time, which leads to a reasonable Δ.

5.2.2 Determining the Initial Traffic Intensity

With a good initial traffic intensity, SDR-ADAPT will take less update cycles to reach the actual traffic load. There are several alternatives to choose from. We study the issue based on the direct-first policy. However, both policies (minimum-cost and direct-first, see section 2.3.4 for their definitions) have similar responses to the initial traffic intensity. Hence, we don't discuss them separately.

- Initializing SDR-ADAPT with full link capacities.

For a network designed with 1% blocking rate, the actual offered load is much less than the full link capacities. The 2-hop rate will be reduced considerably because of the too high recorded traffic intensity (more detailed discussion in section 5.2.3). However, the direct traffic will always have a cost less than 1, hence, won't be affected. Note that the direct traffic is taking up 90% of the total traffic. Therefore, in this case, more than 90% (direct traffic + some 2-hop traffic) of the traffic will be properly routed in the first cycle. At the second cycle, the measured traffic intensity will be a bit lower than, but very close to, the actual value. In our simulations, SDR-ADAPT converges within 4 cycles under all traffic patterns (see Figure 5.2.3.1 and 5.2.3.2).

- Initializing SDR-ADAPT with the traffic intensity matrix $[0]_{30\times30}$.

In this case, there will be much more 2-hop traffic than what should be justifiable in the first cycle, which will use additional link resources, and turn some direct traffic further into 2-hop traffic. The network resource usage will then be inefficient. At the second cycle, the measured traffic intensity will be much higher than its real value. Then, a converging process similar to the one that is initialized with full link capacities will begin.

- Initializing SDR-ADAPT with an arbitrary traffic intensity matrix $[A]_{30\times30}$.

Under different traffic patterns, it may lead to unpredictable performance. Hence, this can't be the best choice.

We can see that initializing SDR-ADAPT with a zero matrix needs at least one more update cycle than with full link capacities. In general, we recommend initializing SDR-ADAPT with full link capacities.

5.2.3 Traffic Intensity Converging Process

The simulation begins with the initial offered load of full link capacities. Under the minimum-cost and the direct-first policies, SDR-ADAPT shows different converging processes and blocking rates.

As an example, Figure 5.2.3.1 shows the traffic intensity converging processes on link 2-16. Link 2-16, with 16 units of effective bandwidth, is the link between the gateway node 2 and the interior node 16. The node 16 has only 8 incoming links, which means all other 21 nodes need 2-hop routes to get to the node 16. The node 2 is a gateway node that can be reached directly from every node. Therefore, link 2-16 will be used as a candidate for second hops for many nodes to reach node 16.

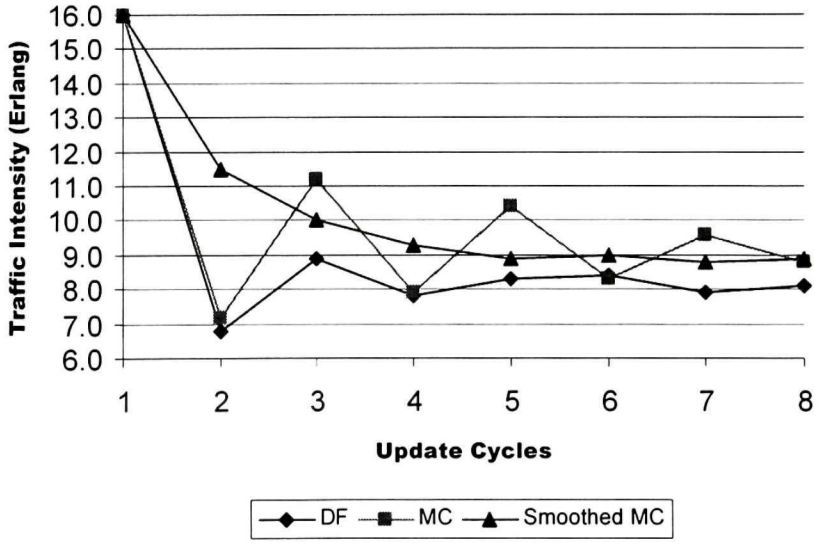


Figure 5.2.3.1: The averaged converging process on link 2-16

In Figure 5.2.3.1 and 5.2.3.2, "DF" stands for "Direct-First"; "MC" stands for "Minimum-Cost"; "Smoothed MC" stands for "Smoothed Minimum-Cost".

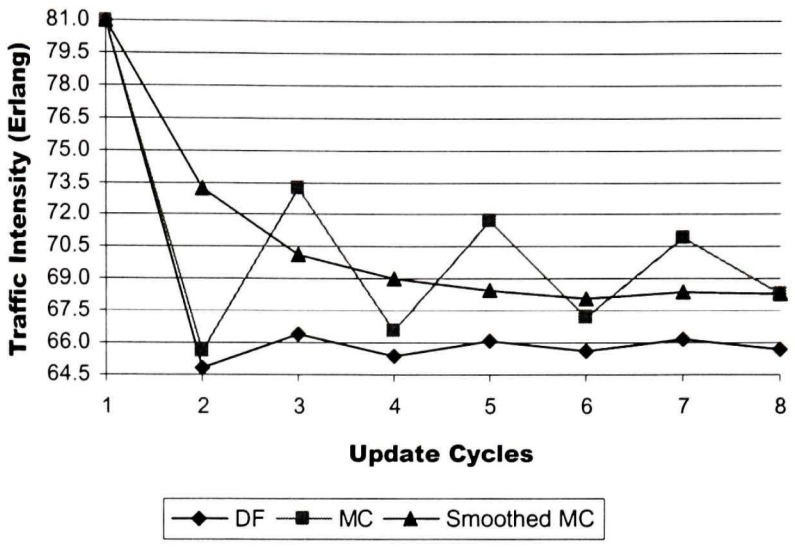


Figure 5.2.3.2: The averaged converging process on link 0-2

1) Minimum-cost (MC) policy

A brief discussion of the MC policy will be helpful to understand the traffic intensity converging process on individual links. Under the MC policy, the costs of all possible routes are calculated upon a connection's arrival. The connection will be assigned to the route with the minimum total cost. Hence, a call could be routed through a 2-hop path, even there is still enough capacity available on the direct link. Since, for every direct route, there are many 2-hop alternatives available (e.g. for the link 2-16, there are 7 alternatives), it is quite often found that a 2-hop route has lower cost than that of the direct route. The following example depicts the situation.

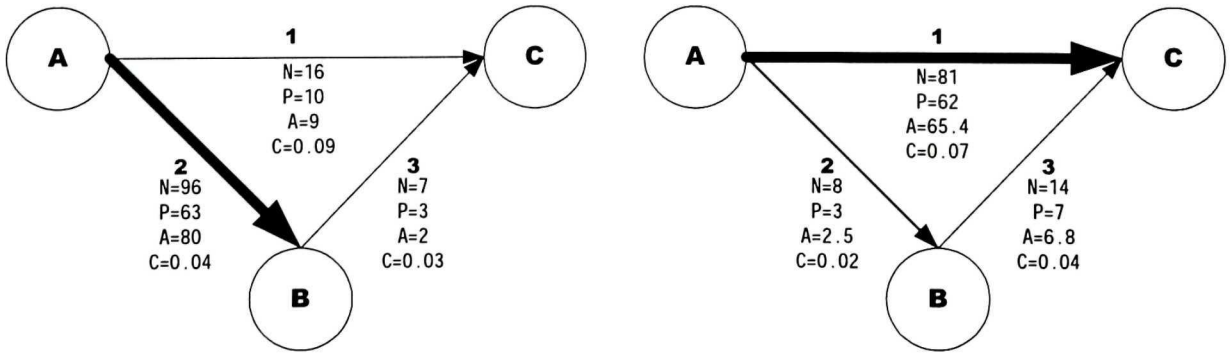


Figure 5.2.3.3: Examples of unwise selection of 2-hop route under MC

In Figure 5.2.3.3, three nodes A, B and C are connected by links 1, 2 and 3, whose status are shown beside the links. "N" is link capacity; "P" is the number of busy bandwidth units; "A" is traffic intensity; "C" stands for cost. For traffic from node A to node C, there are a direct link and a 2-hop path. Under MC, the traffic will be routed through 2-hop routes. Although the cost of 2-hop traffic is the sum of the costs on two links, the MC policy can't effectively control 2-hop traffic.

Through simulations, we found that, in most of the cases, 2-hop traffic is the reason for unstable traffic distributions, which makes the blocking rate hard to control. For example, too much 2-hop traffic can take up the resources planed for direct traffic, and force it to go through 2-hop path, which will further aggravate the problem. The results are higher traffic intensity and higher blocking rate.

During the simulation, within the first cycle, 2-hop traffic is depressed by the high initial traffic intensity. From Figure 5.2.3.1, we can see that the measured traffic intensity on the link 2-16 is around 7 after the first cycle. Then, in the second cycle, the 2-hop traffic grows significantly, which leads to a much higher measured traffic intensity. The 2-hop rate jumps up and down according to the measured traffic intensity, which, in turn, leads to the traffic intensity value oscillating around its target value. For example, if a link is assigned a bigger traffic intensity value than it should have, then the link will be discharged of 2-hop traffic and even direct traffic may put on 2-hop alternatives. Therefore, the recorded traffic intensity of this link will become too small at the end of this cycle and vice versa.

These oscillations lead to unstable traffic distributions and a long converging time. To solve the oscillation problem, we use the following smoothing formula to update the traffic intensity values. The number of updating cycles needed to reach the steady states is cut down almost a half. Within five updates, the traffic intensity values converge to their real values.

$$I = {(I' + I'')}/{2}$$

I' : newly recorded traffic intensity value

I'' : traffic intensity value used in the last period

Figure 5.2.3.1 shows that, under MC, the traffic intensity on the link 2-16 get stable at a higher level than that under DF. We will discuss this observation later.

2) Direct-first (DF) policy

As simulations start, SDR-ADAPT assigns 16 Erlang to the link 2-16 as the initial load, which is much higher than the actual value (around 8.1 Erlang). The direct traffic from the node 2 to the node 16, which takes up 85% of the traffic on the link 2-16, won't be effected by the over estimated traffic intensity value. However, it makes the link much more costly to be used in a 2-hop route. The 2-hop rate is then much less than it should be. After the first update cycle, the measured traffic intensity is 6.8 Erlang (averaged over 3 traffic patterns), which is mostly direct traffic. In the course of the second cycle, the 2-hop traffic arrives at a higher rate than it should, due to the low measured traffic intensity. Some direct traffic may be blocked because the 2-hop traffic is taking more resources than planed. In the third cycle, the measured intensity is a bit higher than the target value, because of the excess of 2-hop traffic in the second cycle. In later cycles, the measured traffic intensity converges to its real value quickly; and the blocking rate reaches and stays at its lowest level. Within 4 updates, the measured intensity matrix converges to the real offered load.

The converging process on the link 0-2 (Figure 5.2.3.2) is slightly different from that on the link 2-16. The link 0-2 is a link between two gateway nodes, which have links to and from every node in the network. Thus, it is seldom used in a 2-hop route. Therefore, it doesn't show clear oscillations in its 2^{nd} and 3^{rd} cycles.

These two examples demonstrate the 2 typical traffic intensity converging processes among the links. In both situations, links are shared by many concurrent connections and most of the traffic is direct traffic, which complies with the assumptions (made in section 2.3.3 to define SDR), i.e. each link is considered statistically independent and a n-hop connection can be seen as n independent connections.

Comparing DF with MC and smoothed MC in Figure 5.2.3.1 and 5.2.3.2, we can see that DF performs better than both MC and smoothed MC. Due to DF's ability to encourage direct traffic and at the same time control 2-hop traffic, DF curves reach their steady states faster and at a lower level than their MC counterparts. Overall, SDR-ADAPT fits in our network environment very well. Under DF, the monitored traffic intensity values converge to their real values within 4 updates.

5.2.4 The Performance under Fixed Traffic Pattern

During the first two or three updates, the measured traffic intensity values are far from the actual steady state values. Hence, at the beginning of these cycles, the blocking rate curves are quite unstable. Figure 5.2.1.1 is an example. The data collected during those unstable periods doesn't present the performance of SDR-ADAPT. Thus, it has to be removed from our study. For this reason, the blocking rates are calculated with the data collected during the last 40 seconds within each cycle, i.e. from the 41^{st} to the 81^{st} second. Hence, we can filter out the unstable period.

SDR-ADAPT is tested under three different traffic patterns, "AM", "Noon" and "PM". All the data in the following tables is in percentage with 95% of confidence interval. With confidence intervals, the curves look rather flat since most of the small fluctuations are filtered out.

Figure 5.2.4.1 presents the average blocking curves. The small bars that spread out from the blocking curves show the confidence intervals of the blocking rate values. We can see that, under DF, SDR-ADAPT converges very fast (within 4 updates), and shows a very stable performance after. The blocking rates are getting lower as the traffic intensity matrices converge to their real values. Smoothed MC, on the other hand, takes more updates to get stable and generates considerably higher blocking rates.

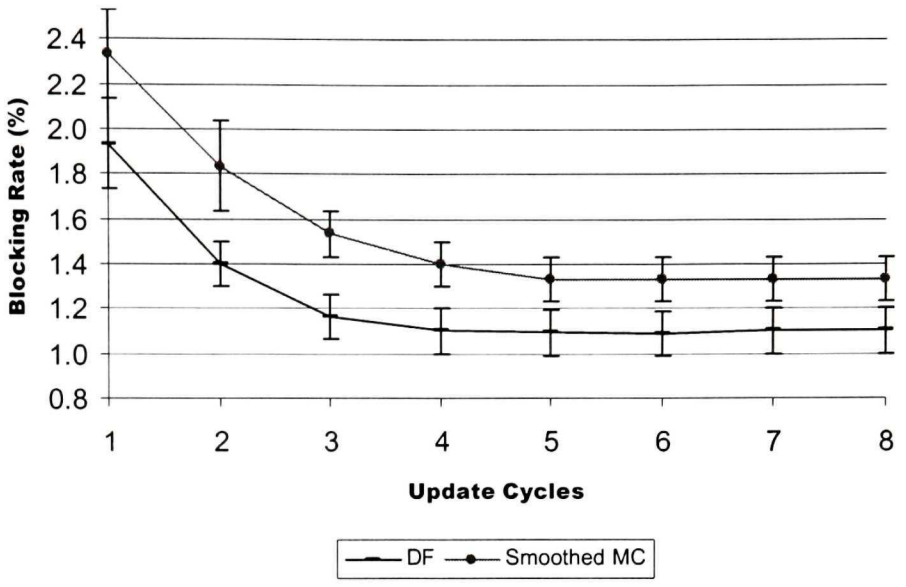


Figure 5.2.4.1: Averaged blocking rate of SDR-ADAPT

Table 5.2.4.1 and 5.2.4.2 presents the more detailed simulation results.

Note:
Direct R. The percentage of the number of connections on direct paths
2-Hop R. The percentage of the number of connections on two-hop paths
B. Rate The percentage of the number of connections that were blocked

	(%)	1st	2nd	3rd	4th	5th	6th	7th	8th
Direct R.	A M	80.2 ± 0.5	77.9 ± 0.5	76.3 ± 0.5	75.7 ± 0.5	74.8 ± 0.5	75.3 ± 0.5	74.6 ± 0.5	75.1 ± 0.5
	Noon	80.6 ± 0.5	78.0 ± 0.5	76.9 ± 0.5	75.8 ± 0.5	75.3 ± 0.5	76.0 ± 0.5	75.4 ± 0.5	75.9 ± 0.5
	P M	81.4 ± 0.5	78.7 ± 0.5	77.3 ± 0.5	76.5 ± 0.5	75.9 ± 0.5	76.3 ± 0.5	75.8 ± 0.5	76.1 ± 0.5
2-hop R.	A M	17.6 ± 0.5	20.3 ± 0.5	22.2 ± 0.5	22.9 ± 0.5	23.9 ± 0.5	23.4 ± 0.5	24.1 ± 0.5	23.6 ± 0.5
	Noon	17.4 ± 0.4	20.3 ± 0.5	21.7 ± 0.5	22.9 ± 0.5	23.4 ± 0.5	22.7 ± 0.5	23.3 ± 0.5	22.8 ± 0.5
	P M	15.8 ± 0.4	19.3 ± 0.5	21.0 ± 0.5	22.0 ± 0.5	22.7 ± 0.5	22.3 ± 0.5	22.8 ± 0.5	22.5 ± 0.5
B. Rate	A M	2.2 ± 0.2	1.8 ± 0.2	1.5 ± 0.1	1.4 ± 0.1	1.3 ± 0.1	1.3 ± 0.1	1.3 ± 0.1	1.3 ± 0.1
	Noon	2.0 ± 0.2	1.7 ± 0.2	1.4 ± 0.1	1.3 ± 0.1	1.3 ± 0.1	1.3 ± 0.1	1.3 ± 0.1	1.3 ± 0.1
	P M	2.8 ± 0.2	2.0 ± 0.2	1.7 ± 0.2	1.5 ± 0.1	1.4 ± 0.1	1.4 ± 0.1	1.4 ± 0.1	1.4 ± 0.1

Table 5.2.4.1: Performance of SDR-ADAPT under smoothed MC

Since, under MC/smoothed MC, the 2-hop traffic has the same priority in using resources, the direct rate can never be as high as under DF. In addition, as the recorded traffic intensity, $\lfloor A_{i\times j} \rfloor$, goes down to the actual load, the 2-hop rate

continues to grow. The more than 20% of the 2-hop rate adds a considerable amount of load to the network, resulting in a higher blocking rate. Figure 2.4.3.1 shows how traffic intensity can affect the blocking rates.

B. Rate		(%)	1st	2nd	3rd	4th	5th	6th	7th	8th
Direct R.	A	M	90.8 ± 0.3	89.1 ± 0.4	90.5 ± 0.4	90.1 ± 0.4	90.0 ± 0.4	90.3 ± 0.4	90.4 ± 0.4	90.2 ± 0.4
	Noon		90.8 ± 0.3	89.2 ± 0.4	90.6 ± 0.3	90.2 ± 0.4	90.4 ± 0.3	90.1 ± 0.4	90.0 ± 0.4	90.3 ± 0.4
	P	M	90.9 ± 0.3	88.9 ± 0.4	90.4 ± 0.3	90.0 ± 0.3	90.1 ± 0.3	89.8 ± 0.4	90.2 ± 0.3	90.0 ± 0.3
2-hop R.	A	M	7.5 ± 0.3	9.6 ± 0.4	8.4 ± 0.3	8.9 ± 0.3	9.0 ± 0.3	8.7 ± 0.3	8.6 ± 0.3	8.8 ± 0.3
	Noon		7.6 ± 0.2	9.6 ± 0.3	8.3 ± 0.3	8.8 ± 0.3	8.6 ± 0.3	8.9 ± 0.3	9.0 ± 0.3	8.7 ± 0.3
	P	M	6.6 ± 0.2	9.4 ± 0.3	8.3 ± 0.3	8.9 ± 0.3	8.8 ± 0.3	9.1 ± 0.3	8.7 ± 0.3	8.9 ± 0.3
	A	M	1.7 ± 0.2	1.3 ± 0.1	1.1 ± 0.1	1.0 ± 0.1	1.0 ± 0.1	1.0 ± 0.1	1.0 ± 0.1	1.0 ± 0.1
	Noon		1.6 ± 0.1	1.2 ± 0.1	1.1 ± 0.1	1.0 ± 0.1	1.0 ± 0.1	1.0 ± 0.1	1.0 ± 0.1	1.0 ± 0.1
	P	M	2.5 ± 0.2	1.7 ± 0.2	1.3 ± 0.1	1.1 ± 0.1	1.1 ± 0.1	1.1 ± 0.1	1.1 ± 0.1	1.1 ± 0.1

Table 5.2.4.2: Performance of SDR-ADAPT under DF

Since the traffic intensity, $\lfloor A_{i\times j} \rfloor$, is initialized with full link capacities, which is much higher than the actual offered load, the two-hop rate of the first cycle is much lower than for the other cycles. Many of the two-hop connections are blocked because of over priced two-hop routes, which leads to the highest blocking rate in all cycles and the low measured traffic intensity, $\lfloor A_{i\times j} \rfloor$. However the direct traffic is not impacted by the over estimated $\lfloor A_{i\times j} \rfloor$. Since many two-hop connections are blocked, more resources are available for direct traffic.

In the second cycle, $\lfloor A_{i\times j} \rfloor$ gets a lower than actual value (see Figures and explanations in section 5.2.1), which means that links are cheaper to be used in 2-hop routes. 2-hop rates, hence, reach their peak values. The direct traffic is cut off a bit because two-hop traffic is taking too much resource. Although the blocking rate decreases dramatically due to the admission of more two-hop connections, the overall link utilization is not so efficient.

In the third cycle, $\lfloor A_{i\times j} \rfloor$ gets very close to its destination, just a bit higher. The direct traffic gains a bit of share, while two-hop traffic gives back some resources. The blocking rate continues to go down.

In later cycles, the blocking rate stays at its lowest level, around 1%. The direct traffic and two-hop traffic are stable at around 90% and 8% respectively.

During the whole simulations, DF gives very stable direct rates, since $\lfloor A_{i\times j} \rfloor$ is never significantly lower than its actual value.

Comparing the two tables, we can find that the reason for which DF outperforms smoothed MC is that DF leads to a more preferable traffic distribution. With smoothed MC, the direct traffic is always less than 80% and stabilizes around 75%. The 2-hop traffic stabilizes around 22%, which is about 13% more than that in DF. The extra amount of 2-hop traffic causes the higher traffic intensity in Figure 5.2.3.1 and 5.2.3.2, and a higher blocking rate in Figure 5.2.4.1. From the above comparison, we can draw the following conclusions:

1. Traffic should be routed on its direct link whenever possible, even when some 2-hop routes have lower costs;
2. The more accurate the estimated traffic intensity, $\lfloor A_{i\times j} \rfloor$, is, the lower the blocking rate is;
3. Under estimated $\lfloor A_{i\times j} \rfloor$ leads to high 2-hop rate;
4. DF shows low a blocking rate even when the estimated $\lfloor A_{i\times j} \rfloor$ is far from the actual load. DF is more adaptive to a bad estimation of the traffic intensity;
5. MC/smoothed MC are more vulnerable than DF to under estimated $\lfloor A_{i\times j} \rfloor$;
6. In general, SDR-ADAPT performs better under the DF policy than under the MC/smoothed MC policy.

5.2.5 The Performance under Changing Traffic Pattern

SDR-ADAPT performs very well under the DF policy with a fixed traffic pattern. In a telecom network, the traffic pattern may vary from time to time. It would be useful for SDR-ADAPT to adapt to an ever-changing traffic intensity matrix automatically.

In the following, still using the DF policy, we expand the update period to 200 seconds to give SDR-ADAPT a more practical update interval; and simulate the 3 traffic patterns consecutively (from the AM pattern to the noon pattern, then to the PM pattern), to examine the performance of SDR-ADAPT under a shifting pattern. The simulation starts with full link capacities and stays for 6 cycles in each pattern.

Figure 5.2.5.1 shows the simulation result. With a shifting traffic pattern, SDR-ADAPT shows a very similar performance to that with a fixed traffic pattern. The blocking rates when the traffic pattern changes, at the 7^{th} and 13^{th} cycle, are rather low, since DF can handle a wrongly estimated traffic intensity quite well. In addition, the blocking curve gets stable very fast after each change of pattern.

This simulation proves that SDR-ADAPT can also track a changing traffic pattern with 200-seconds update cycles.

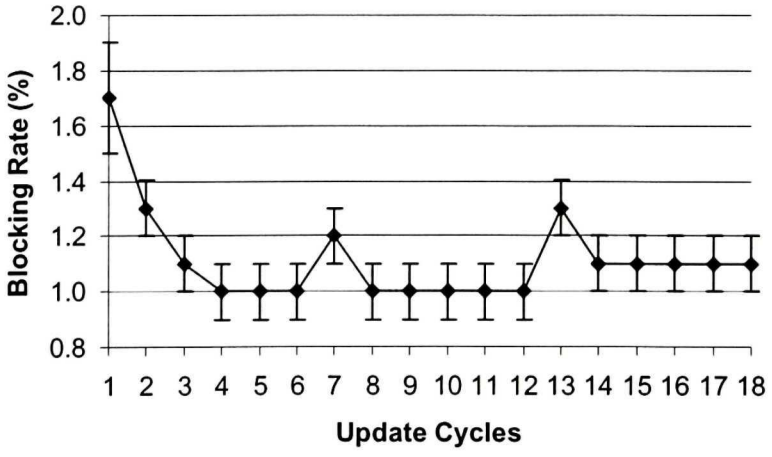


Figure 5.2.5.1: Blocking rate under traffic pattern shifting

5.3 Centralized Routing Controls (CRC)

With CRC, the real time link usage information is supposed to be available; the simulations are to find the best threshold value for the TR. As defined in section 2.5.2, a threshold value, x, is decided by 3 parameters, C B and δ, in which C and B are fixed values for a specific link. To use TR, one needs to specify a x for every link in the network. Clearly, this isn't necessary and could be a scalability problem. Since δ reflects the traffic intensity on a link, if several links have similar loads (blocking rates), the same δ can be used on all of the links. When we assume that, in a well-engineered network, the traffic will be evenly distributed among the links, we can actually use the same δ for the whole network. This assumption simplifies the threshold values determination process. We can get different threshold values for the network by changing δ. To be clear, we use δ directly to identify the simulation results, instead of the actual threshold value x.

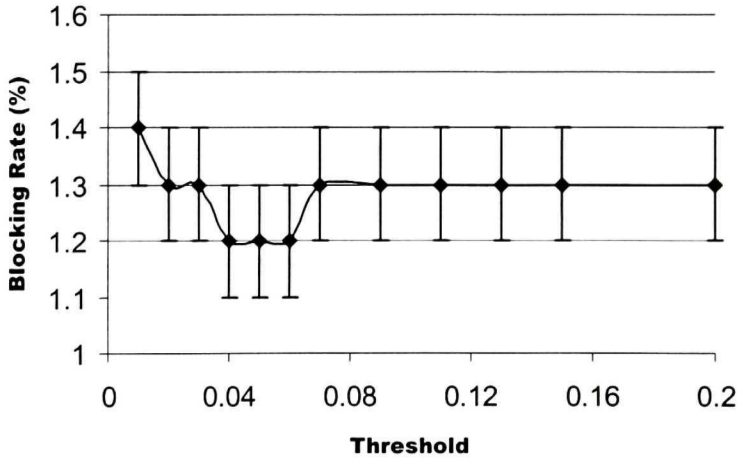


Figure 5.3.1: Average blocking rate curve under different δ s

δ	Blocking Rate (%)	Direct Rate (%)	2-hop Rate (%)
0.01	1.4 ± 0.1	88.5 ± 0.4	10.1 ± 0.4
0.02	1.3 ± 0.1	89.3 ± 0.4	9.4 ± 0.3
0.03	1.3 ± 0.1	89.9 ± 0.4	8.8 ± 0.3
0.04	1.2 ± 0.1	90.1 ± 0.4	8.7 ± 0.3
0.05	1.2 ± 0.1	90.3 ± 0.4	8.5 ± 0.3
0.06	1.2 ± 0.1	90.4 ± 0.3	8.4 ± 0.3
0.07	1.3 ± 0.1	90.5 ± 0.3	8.2 ± 0.3
0.09	1.3 ± 0.1	90.6 ± 0.3	8.1 ± 0.3
0.11	1.3 ± 0.1	90.6 ± 0.3	8.1 ± 0.3
0.13	1.3 ± 0.1	90.7 ± 0.3	8.0 ± 0.3
0.15	1.3 ± 0.1	90.7 ± 0.3	8.0 ± 0.3
0.20	1.3 ± 0.1	90.9 ± 0.3	7.7 ± 0.3

Table 5.3.1: Average blocking rates under different δ s

The discussion in section 2.5.2 has shown that the threshold should be set at the link utilization state that has a RUC cost around 0.5; let's call it 0.5-cost-state. From Figure 2.5.2.2, we can see that this threshold should be rather close to the link capacity, which means a very small δ. Therefore, we start the test from $\delta = 0.01$.

Figure 5.3.1 shows the blocking rate curve under different δ s. The vertical bars on the curve are the 95% confidence intervals of the blocking rates. The blocking curve reaches its lowest level when δ is around 0.05. When δ moves from 0.01 to 0.05, blocking rates get lower. The curve moves up again when $\delta > 0.06$.

Since CRC employs the DF policy, the changing of δ won't have a direct impact on direct traffic. The different blocking rates are actually caused by different costs for 2-hop traffic. Figure 2.5.2.2 shows us the reason behind the curve.

Now, let's move the threshold from the right side to the left. At the beginning, when $\delta < 0.04$, the threshold value is too low; links have 0 cost even when very little free capacity is available, which leads to high 2-hop rates (see Table 5.3.1). For example, a 2-hop route has a RUC cost $C = C_{A,N}(p)_1 + C_{A,N}(p)_2 = 0.7 + 1 = 1.7$, and should be blocked. However, if the threshold is at the 0.8-cost-state, the cost

will be 1, and the 2-hop route is admissible. The cost curve is very steep at this stage. Every step to the left can cause rather big changes on simulation results, including blocking rate, direct rate and 2-hop rate. In addition, it shows that the closer the threshold is to the 0.5-cost-state, the lower the blocking rate is. Then, continuing to move the threshold to the left, $0.04 \le \delta \le 0.06$, the simulations show the best performance.

After, when $\delta > 0.06$, the cost curve is getting increasingly flat. Hence, as the threshold moves further left, the changes in the simulation results are becoming negligible. The blocking rate gets higher and 2-hop rate gets lower.

Overall, it shows the trend that the bigger the δ is, the lower the 2-hop rate is. The reason is that δ specifies the amount of bandwidth that is reserved for direct traffic.

In addition, we can also see from figure 2.5.2.2 that if the traffic intensity offered to the network is higher than that is planed (causing 1% blocking rate), the lowest blocking rate δ should be higher than 0.05. For example, comparing the curves $A = 90Erl.$ and $A = 80Erl.$, and their threshold curves, we can see that the 0.5-cost-state for the $A = 90Erl.$ curve is less than that for the $A = 80Erl.$ curve. For the same reason, when the offered load is lower than planed, δ should be lower than 0.05; and the threshold should be moved right. This finding could be used as a guide to find the correct threshold quickly, as the offered load in a network changes.

The CRC, with $\delta = 0.05$, shows a bit higher blocking rate than that in SDR-ADAPT. There could be 3 reasons:

1) The TR can not compare the costs of two 2-hop routes; hence, it can only find an admissible 2-hop path, not the optimal one.
2) It can often be found that, for some links, there is no link utilization state that has a 0.5 cost. Therefore, the TR can not always be accurate in admission control.
3) To use a single δ for the whole network, we assume that the traffic is evenly distributed (i.e. 1% blockings) on every link. If in some traffic

pattern, the traffic demands are not balanced on the network, TR can't make optimal routing decisions.

5.4 Decentralized Routing Control (DRC)

Based on the best threshold (when $\delta = 0.05$) obtained from CRC, DRC study the performance of TR utilizing near real time link utilization information. Since δ won't change, the traffic distribution (direct rate and 2-hop rate) is almost fixed. Therefore, the direct rate and the 2-hop rate are not presented in the simulation results.

As discussed in section 2.6.1, the captured energy is growing as the update frequency increases, but not linearly (Figure 2.6.1.1). One of our objectives is to find the optimal balance between the update frequency and the blocking rate.

Figure 2.6.1.1 shows that once 99% of energy has been captured; increasing the update frequency to seize more energy becomes more and more inefficient. 99% of captured energy is also proven to be a good balancing point by simulation results.

Figure 5.4.1 shows the blocking rate under different captured energy levels. It is clear that between 90% and 99%, the blocking rate decreases almost linearly as the captured energy grows. However, when the captured energy reaches 99%, the blocking rate has a major drop down. Then, as the captured energy is approaching 100%, the blocking rate stays almost the same. From this figure, we can also reach the conclusion that 99% of captured energy is the threshold point that we've been looking for.

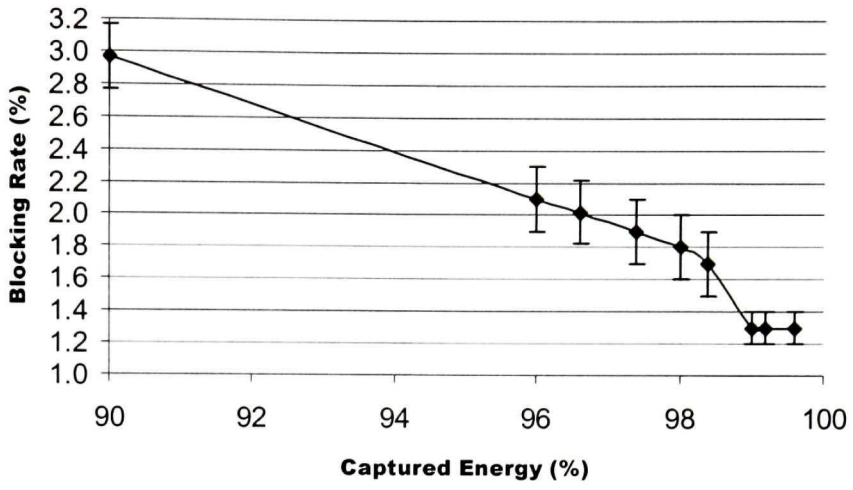


Figure 5.4.1: Blocking rate curve under different captured energy levels

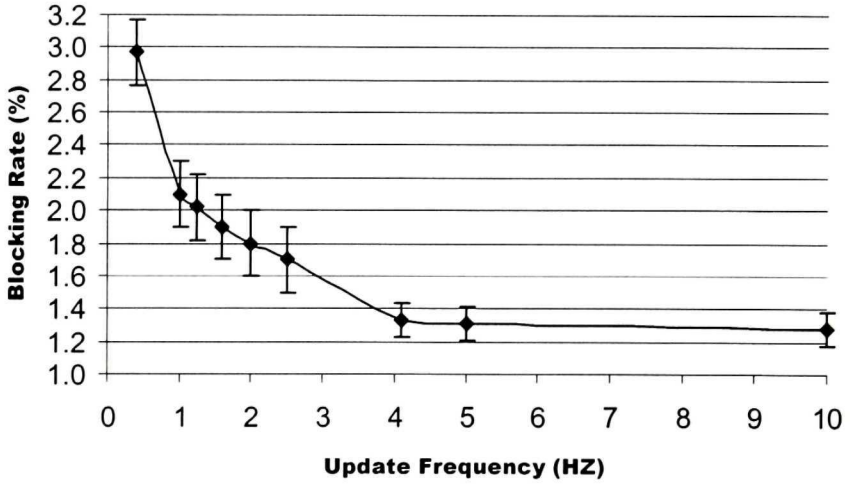


Figure 5.4.2: Blocking rate curve under different update frequencies

According to above discussion, we can estimate that the optimal update frequency should be the one that can capture 99% of energy. Figure 5.4.2 proves this estimation. It shows that after the update frequency increases to 4 HZ, the blocking rate remains almost unchanged. Therefore 4 HZ is the optimal update frequency. The table 5.4.1 shows that with 4HZ, we can capture 99% of energy.

Update Frequency (HZ)	Captured Energy (%)	Blocking Rate (%)
0.4	90.0	3.0 ± 0.2
1.0	96.0	2.1 ± 0.2
1.3	96.6	2.0 ± 0.2
1.6	97.4	1.9 ± 0.2
2.0	98.0	1.8 ± 0.2
2.5	98.4	1.7 ± 0.2
4.1	99.0	1.3 ± 0.1
5.0	99.2	1.3 ± 0.1
10.0	99.6	1.3 ± 0.1

Table 5.4.1: Blocking rate with corresponding update frequency and energy level

The update frequency is decided by the mean holding time of the routed connections, which is 5 seconds in our simulations. Although 4 HZ (0.25 second) is not a realistic update frequency for a telecom network, it is valid as a proof of the estimation theory.

With 99% of the energy captured, DRC shows only slightly higher blocking rate than that of CRC. Hence, we can have the conclusion that:

1. Without any requirement for real time information (traffic intensity or link utilization) and complex routing cost computation, DRC can achieve comparable performance to SDR-ADAPT and CRC.
2. Based on CRC, DRC also has the limitations (discussed in section 5.3) induced by TR.
3. DRC is a practical (implementable) routing scheme for IP QoS routing.

Chapter 6

Conclusions and Further Studies

6.1 Conclusions

This thesis studies the admission control and routing problems of single class IP QoS service. Based on SDR-ADAPT and TR, the CRC and DRC routing schemes are proposed and examined.

SDR-ADAPT is simulated under both the MC and the DF policies as an analytical model of SDR. We found that: (1) the full link capacities matrix is an adequate initial value for SDR-ADAPT; (2) in most of the cases, the 2-hop traffic is the reason for unstable traffic distributions, which makes blocking rate hard to control; (3) the reason for which DF outperforms smoothed MC is that DF encourages direct traffic and at the same time controls 2-hop traffic; (4) traffic should be routed on direct link whenever possible, even when some 2-hop routes have lower costs; (5) the more accurate the estimated traffic intensity, $\lfloor A_{i\times j} \rfloor$, is, the lower the blocking rate is; (6) under estimated $\lfloor A_{i\times j} \rfloor$ leads to high 2-hop rate; (7) DF is more adaptive to a bad estimation of the traffic intensity; (8) under DF, the blocking rate curve reaches its steady state faster and at a lower level than that under MC/smoothed MC.

TR is used to replace the RUC function under CRC; hence, only DF can be implemented. The simulations show that: (1) δ can be used to reflect the traffic intensity on a link and has a similar impact on traffic distribution as $\lfloor A_{i\times j} \rfloor$ in SDR-ADAPT; (2) the bigger the δ is, the lower the 2-hop rate is; (3) δ should be set to match the 0.5-cost-state; e.g. when a network is designed to carry its offered load with 1% of blocking rate, δ should be around 0.05; (4) when the blocking

rate is higher than 1%, δ should also be higher than 0.05, and vice versa; (5) the closer the threshold is to the 0.5-cost-state, the lower the blocking rate is; (6) when the blocking rate is higher than 1%, the TR will be less efficient; (7) hence, one of the network design objectives should be to limit the blocking rate on each link to 1% or less; (8) when the blocking rates on some links are similar, a single δ can be used to control the threshold values on all of them.

CRC shows slightly higher blocking rates than SDR-ADAPT. The reasons are: (1) TR can only find an admissible 2-hop path, not the optimal one, since it can not compare the costs of 2-hop routes; (2) for some links, the exact 0.5-cost-state may not exist, therefore, TR can not always be very accurate in admission control; (3) for some traffic patterns, the traffic rates on some links may not comply with their planned values; hence, using a single δ for the whole network may lead to some non-optimal routing decisions.

The study of DRC is based on the simulation results of CRC. Under DRC, the last unrealistic assumption, the real time link utilization information, is removed. It is found from the simulations that: (1) 99% of captured energy is proved to be a good balancing point, i.e. the optimal update frequency should be the one that can capture 99% of the energy; (2) the update frequency is decided by the mean holding time of the routed connections; (3) the captured energy grows as the update frequency increases, but not linearly; (4) the blocking rate decreases as the update frequency increases, but not linearly; (5) DRC can achieve a comparable performance to SDR-ADAPT and CRC; (6) DRC gives slightly higher blocking rates than those in SDR-ADAPT and CRC, since it has the same limitations from CRC and it doesn't have access to any real time information; (7) without any requirement for real time information (traffic intensity or link utilization) and complex routing cost computation, DRC is a practical (implementable) routing scheme for IP QoS routing.

6.2 Further Studies

Further studies could be conducted to investigate the following areas:

1) improve the performance of SDR routing schemes;
2) implement routing schemes for multi-service class case;
3) investigate routing schemes for networks, in which traffic is not distributed evenly;
4) compare the performance of SDOR with other advanced routing schemes.

Here are some sample ideas:

- To further improve the performance of CRC and DRC. We can customize the threshold values for each link in a network, i.e. specify a δ for every link. In this way, we do not need to assume that the offered load is evenly distributed. We can monitor the traffic intensity on each link, just like we did in SDR-ADAPT, then, find the 0.5-cost-state for each link according to the recorded traffic intensity and apply a proper δ for each link based on its load. This will lead to a δ matrix. In addition, to simplify the process, we can divide the network into different parts that have similar load, and use one δ for each of them. With this approach, no matter how the traffic pattern is going to change, we will always be using the best threshold values. However, depending on the offered traffic pattern, the improvement offered by this modification may be quite limited.
- Based on this research, one can step forward to study premium class non-homogenous IP traffic and multi-service class non-homogenous IP traffic, e.g. connections with different mean holding times. For premium class non-homogenous IP traffic, the same PHB may be used to describe the treatment of the aggregated flows. However the CAC process becomes much more complicated (assuming that different types of connections do not generate the same amount of revenue), since the resources competition among different types of premium connections has to be taken into account. For multi-service class non-homogenous IP traffic, non-

premium classes can be treated as degraded premium classes. They can be forwarded in a similar manner as the premium class but with lower priority and less resources, such as bandwidth. In this way, the same CAC and routing algorithm can be used in a multi-service class DiffServ network.

- The performance of MPLS based static multipath routing could be simulated and compared with SDOR. All traffic is sent through pre-established routes, with no real time route computation. Traffic between the same pair of source and destination can be balanced among all available routes. However, without considering the global link utilization, static multipath routing cannot be the optimal routing. For example, when multiple routes share the same link, static routing does not differentiate them. Some routes may be the last path available; some others may have several alternatives. Assigning administrative weights to some routes may help to ease the problem, but it will be hard to adapt to the changes of traffic pattern.

Appendix A

Traffic Distribution and Topology

In this section, we will present the algorithms that we used to build the network.

- **Originating traffic**

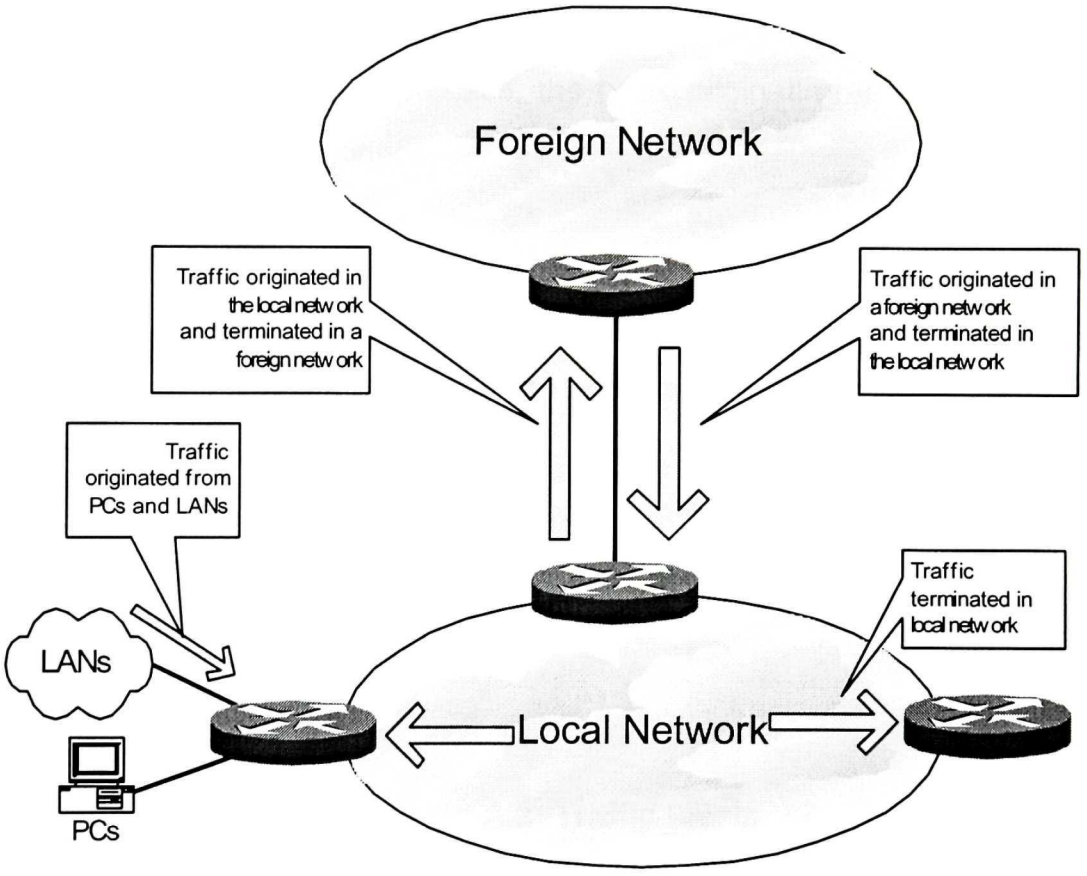


Figure A.1: Originating and terminating traffic

The originating traffic of a node is the traffic that enters a network through the node. It is generated by customers (or networks).

Some nodes are originating more traffic than others, because they serve more customers. In this implementation, we use the following scale:

Nodes size	Originating capability	Prob. of occurrence
Small	0.5	0.3
Medium	1	0.4
Large	1.5	0.2
Very large	2	0.1

Table A.1: Node size and distribution

The size of each node is determined by drawing randomly according to the probability of occurrence in Table A.1. For instance, node 1 is drawn to be a small node. Let TO be the total originating traffic corresponding to the normalized peak traffic of a medium node. The total originating traffic of node 1 would be $TO_1 = 0.5\,TO$. Following this process, the peak originating traffic and the size of each node can be determined.

- **Non-coincidence of peak hours**

A node may not keep generating traffic at its peak rate. In general, we consider that the originating traffic of a node can be in one of several levels defined in the table below.

Traffic level	Fraction of peak traffic
Peak hour	1.0
Side peak hour	0.85
Off peak hour	0.70

Table A.2: Traffic levels

A traffic level can be associated with a node based on its time zone. As an example, for a continental U.S. network, the different traffic levels can be associated with the nodes in the eastern, the central and the western regions in three particular hours.

Hour	Region of the US where the node is located		
	East	Central	West
10:00 EST	Peak hour	Side peak hour	Off peak hour
12:00 EST	Side peak hour	Peak hour	Side peak hour
13:00 EST	Side peak hour	Side peak hour	Peak hour

Table A.3: Peak hours

For example, if node 1 is in its side peak hour, its originating traffic becomes $TO_1 \Leftarrow 0.85\,TO_1$. Repeating this process, we can generate the originating traffic demands for all nodes for a certain hour.

- **Allocating the traffic demand**

Traffic may be destined to a node within the same network or to a node in a foreign network. In the former case, we call the traffic the intra-network traffic. In the later case, the traffic is delivered through 3 gateway nodes, and we call it the inter-network traffic.

We assume that there is no special limitation for traffic to remain in one network. Thus, we assume that the destinations for originating traffic are evenly distributed across the networks. Considering that there are X such networks; we are studying one of the networks; and the remaining $X-1$ are the foreign networks.

Let the probability that the originating traffic belongs to the intra-network traffic be P. The proportion of inter-network traffic is $1-P$.

We assume that the amount of traffic that is generated by a node and that is terminated at the node are in proportion. Thus, let TO_i be the total originating demand of node i. The demand allocated to node j from node i in the network is:

$$TO_{ij} = TO_i \frac{TO_j}{\sum_{k \neq i} TO_k}$$

Let the foreign networks be $F_1,\ldots,F_{X-1}$. The originating demand from node i to each of these networks is:

$$TO_{iF_j} = TO_i \frac{1-P}{X-1}, \qquad j = 1,\ldots,X-1$$

A set of gateway nodes can be used to reach a foreign network F_j. Let the gateway nodes be $G_{F_j} = \left\{g_{F_j,1}, g_{F_j,N_{F_j}}\right\}$, where $g_{F_j,k}$ identifies the k^{th} node in the set and N_{F_j} denotes the number of nodes in the set.

Assuming that the originating traffic from node i to the foreign network F_j is uniformly distributed among its gateway nodes; for each $g_{F_j,k}$ in G_{F_j} the originating demand from node i to $g_{F_j,k}$ is:

$$TO_{i,g_{F_j,k}} = TO_{iF_j} \frac{1}{N_{F_j}}$$

To determine the incoming traffic from a foreign network F_j and terminating in the network, we assume that the incoming traffic is equal to the total traffic originating in the network and terminating in the foreign network F_j. Let TO_{F_j} be the total traffic originating in the network and terminating in the foreign network F_j.

$$TO_{F_j} = \sum_{\text{all } i} TO_{iF_j}$$

Let TT_{F_j} be the total traffic incoming from the foreign network F_j and terminating in the network. We have:

$$TO_{F_j} = TT_{F_j}$$

The terminating traffic incoming from foreign network F_j is uniformly distributed among its gateway nodes. We can express the terminating traffic of foreign network F_j incoming from gateway node $g_{F_j,k}$ as:

$$TT_{F_j,g_{F_j,k}} = TT_{F_j} \frac{1}{N_{F_j}}$$

In addition, we consider that the terminating traffic incoming from $g_{F_j,k}$ is distributed to each node in the network proportionately to its originating traffic.

$$TT_{F_j,g_{F_j,k},j} = TT_{F_j,g_{F_j,k}} \frac{TO_j}{\sum_k TO_k}$$

- **Building The Capacity Matrix**

The network topology is built according to the traffic distribution. We use the node-to-node traffic demands as the input of our sizing algorithm. This algorithm is designed to achieve our network modeling objectives:

- About 50% Connectivity
- About 90% Traffic on Direct Link
- One or Two Hop Paths

First, there must be enough outgoing capacity from each node. For this purpose, we fist determine the total originating traffic from each node for each of the traffic periods. Then, we determine, $TO_{i,\max}$, the maximum total originating traffic demand from node i over all of the traffic periods. We assume that the peak efficiency of the link is η. The total outgoing capacity from node i can then be set as $TO_{i,\max}/\eta$ (η = 0.7).

Second, there must be enough incoming capacity at each node. For this purpose, we determine the total terminating traffic at each node for each of the traffic periods. Then, we determine, $TT_{i,\max}$, the maximum total terminating traffic demand at node i over all of the traffic periods. For the same reason as above, the total incoming capacity from node i is set as $TT_{i,\max}/\eta$.

Third, we have to ensure the consistency of the total originating and terminating traffic capacity. The total originating capacity in the network $TO_{\max}$ can be expressed as:

$$TO_{\max} = \sum_{\text{all } i} TO_{i,\max}$$

The total terminating capacity in the network $TT_{\max}$ can be expressed as:

$$TT_{\max} = \sum_{\text{all } i} TT_{i,\max}$$

$TO_{i,\max}$ is the sum of the elements in row i of the capacity matrix; $TO_{\max}$ is the sum of every row sum of the capacity matrix; and the $TT_{\max}$ is the sum of every column sum of the capacity matrix. Hence, we should have:

$$TO_{\max} \equiv TT_{\max}$$

To ensure this condition, we adjust this $TO_{i,\max}$ and $TT_{i,\max}$ as below.

If $TO_{\max} > TT_{\max}$, scale up $TT_{\max}$, i.e.:

$$TT_{i,\max} \Leftarrow TT_{i,\max} \frac{TO_{\max}}{TT_{\max}}$$

If $TO_{\max} < TT_{\max}$, scale up $TO_{\max}$, i.e.:

$$TO_{i,\max} \Leftarrow TO_{i,\max} \frac{TT_{\max}}{TO_{\max}}$$

Fourth, we find a capacity matrix that matches all of the $TO_{i,\max}$ and $TT_{i,\max}$. This can be achieved by estimating a set of initial values for the capacity matrix and by updating the rows and columns of the capacity matrix repeatedly until the procedure converges. Let $C = [c_{i,j}]$ be the capacity matrix, initialized with the average traffic demand (i.e. for a 30-node network, the total originating traffic demands of node i are evenly distributed among the other 29 nodes, $[c_{i,j}] = C_{i,\max}/29$). We use the following iteration procedure to update $[C_{i,j}]$, based on a method originally proposed by Fruithof.

Row iteration: for all i, set

$$c_{i,j} \Leftarrow c_{i,j} \frac{TO_{i,\max}}{\sum_k c_{i,k}}, \qquad \text{all } j$$

Column iteration: for all j, set

$$c_{i,j} \Leftarrow c_{i,j} \frac{TT_{j,\max}}{\sum_k c_{k,j}}, \qquad \text{all } i$$

The row and column updates continue until the capacity matrix $C=\left[c_{i,j}\right]$ stabilizes. In this implementation, the row and column iterations are stopped when the modification of $C=\left[c_{i,j}\right]$ between two iterations becomes less than a threshold (e.g. 10^{-4}).

- **Building The Topology**

Although the capacity matrix normally cannot be used directly to build the topology, it provides two pieces of important information: where and how much the traffic volume is. There are still 3 questions that need to be figured out. First, how many links spread out from every single node to satisfy the fifty percent connectivity? Second, which nodes are on the other ends of the links? Third, how much capacity is on each link?

To solve the first question, we use the following equation:

$$\frac{TO_{i,\max}}{TO_{\max}} = \frac{x_i}{n*(n-1)*0.5}$$

where n is the number of the nodes in the network, x_i is the number of links spreading out from node i. $n*(n-1)*0.5$ gives the total number of links that complies with 50% connectivity.

For the second question, we first have to check the i^{th} row of the capacity matrix $C=\left[c_{i,j}\right]$ to find out the nodes that have direct connections from node i. Then, pick the biggest x_i items. For example, when $x_i=6$, the biggest 6 items in row i will be selected. After sorting by size, the capacity vs. destination node-ID graph has a shape like in Figure A.2.

The 6 trunks have to be able to carry all of the traffic originating from node i, not only the traffic that goes directly to these 6 destination nodes, but also that toward the other nodes. To answer the third question, one of the simplest ways is

to add all of the other 23 elements in row i of $C=\left[c_{i,j}\right]$ together and divide evenly into 6 parts, then add each part to one of the 6 links.

After running above process for each node, the topology matrix is ready.

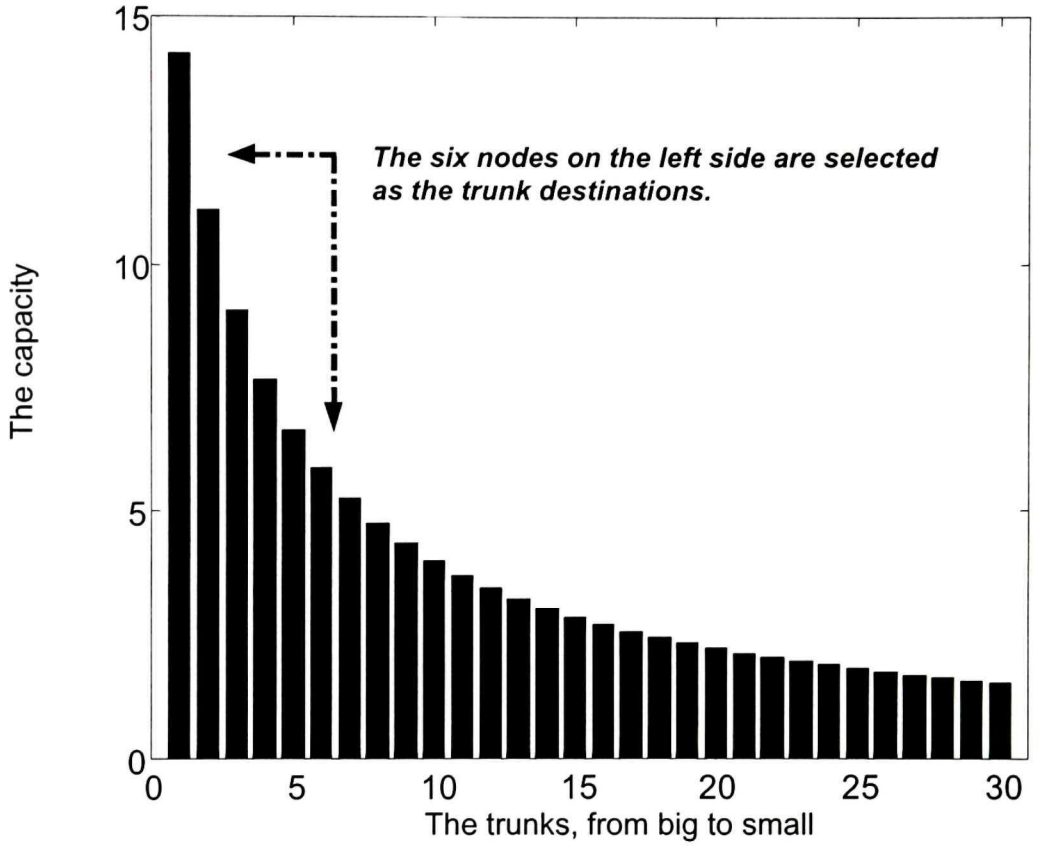


Figure A.2: Capacity vs. Destination Node-ID

References

[1]. G. R. Ash, J. S. Chen, A. E. Frey, and B. D. Huang, "*Real-time Network Routing in a Dynamic Class-of-Service Network*", Proc., 13th Int'l Teletraffic Cong., Copenhagen, June 1991

[2]. S. Blake et. al., "*An Architecture for differentiated Services*", draft-ietf-diffserv-arch-02.txt.

[3]. E. Rosen et al., "*Multiprotocol Label Switching Architecture*", draft-ietf-mpls-arch-02.txt.

[4]. L. Andersson et. al., "*LDP Specification*", draft-ietf-mpls-ldp-02.txt., November 1998.

[5]. B. Davie et. Al., "*Use of Label Switching With RSVP*", draft-ietf-mpls-rsvp-00.txt. *AND* "*Explicit Route Support in MPLS*", draft-daive-mpls-explicite-routes-00.txt

[6]. R. Kawahara, "*Traffic Control for Weighted Bandwidth Allocation per Flow in Diffserv Architecture*", Networks 2000 Conference, Toronto, Canada, September 10-15, 2000.

[7]. S. Bosch, "*Multi-Objective Traffic Engineering of IP Networks Using Label Switched Paths*", Networks 2000 Conference, Toronto, Canada, September 10-15, 2000.

[8]. F. Kelly, "*Notes on Effective Bandwidth*", 1997.

[9]. UC Berkeley, LBL, USC/ISI and Xerox PARC, "*The NS Manual*", December 9, 2000.

[10]. Dziong Z., Liao K-Q., and Mason L. G. 1993. "*Effective Bandwidth Allocation and Buffer Dimensioning in ATM Based Networks With Priorities*". Computer Networks and ISDN-Systems, 25:1065-1078.

[11]. M. Chatzaki, and S. Sartzetakis. "*QoS-Policy based Routing in Public Heterogeneous Broadband Networks*". In Proceedings of Interneting'98 Conference, Ottawa, Canada, 6-10 July 1998.

[12]. K. Nichols, V. Jacobson, L. Zhang. "*draft-nichols-diff-svc-arch-00.txt*". IETF INTERNET-DRAFT, Nov. 1997.

[13]. McDysan, D.E. and D. L. Spohn. "*ATM: Theory and Application*", McGraw-Hill, New York, 1995.

[14]. H. Tode, Y. Sakai, M. Yamamoto, H. Okada, "*A Study on the Support of Multicast Traffic in ATM Networks*", June 1993.

[15]. T. J. Ott and K. R. Krishnan. "State-dependent routing of telephone traffic and the use of separable routing schemes", Proc. 11th International Teletraffic Congress, Kyoto, September 1985.

[16]. Krishnan, K. R. and Hibner-Szabo de Buts, F., "*Admission Control and State-Dependent Routing for Multirate Circuit-Switched Traffic*", Proceedings of the 15th International T&traffic Congress, Washington D.C., USA, 1997.

[17]. A. Girard, "*Routing and Dimensioning in Circuit-switched Networks*", Addision-Wesley, 1990.

[18]. Z. Dziong et al., "*On Adaptive Call Routing Strategy for Circuit Switched Networks – Maximum Reward Approach*", in Twelfth International Teletraffic Congress, Torino, June 1988.

[19]. Z. Dziong, B. Shukhman and L. G. Mason, "*Estimation of Aggregate Effective Bandwidth for Traffic Admission in ATM Networks*", INRS-Telecommunications, IEEE 1995

[20]. G. Kesidis, "*Modeling to Obtain the effective Bandwidth of a Traffic Source in an ATM Network*", E&CE Dept., Davis Centre, U. Waterloo, IEEE 1994

[21]. F. P. Kelly, "*Routing in Circuit Switched Networks: Optimization, Shadow Prices and Decentralization*", Advanced Applied Probability, vol. 20, pp. 112-144, 1988

[22]. Ren-Hung Hwang, James F. Kurose and Don Towsley, "*MDP Routing for Multirate Loss Networks*", 2000

[23]. A. F. Atlasis, E. D. Baltatzis, G. I. Stassinopoulos, I. Venieris, "*A Linear-Based Trunk Reservation Routing Algorithm for ATM Networks*", LCN '98. Proceedings, 23rd Annual Conference on 11-14 Oct. 1998

[24]. V. Anantharam, M. Benchekroun, "*Trunk Reservation Based Control of Circuit Switched Networks with Dynamic Routing*", Proceedings of the 29th Conference on Decision and Control, Dec. 1990

[25]. A. A. Puhalskii, M. I. Reiman, "*A Critically Loaded multirate link with trunk reservation*", Queueing System 28, 1998, 157-190

[26]. I. Andrikopoulos and G. Pavlou, "*Supporting Differentiated Services in MPLS Networks*", Proc. 7th Int'l. Wksp. QoS, London, U.K., May 1999.

[27]. F. Le Faucheur et al., "*MPLS Support of Differentiated Services*", Internet draft, draft-ieft-mpls-diff-ext-07.txt, IETF. Aug. 2000, work in progess.

[28]. J. Heinanen, F. baker, W. Weiss, and J. Wroclawski, "*Assured Forwarding PHB Group*", IETF RFC 2597, June 1999.

[29]. V. Jacobson, K. Nichols, and K. Poduri, "*An Expedited forwarding PHB*", IEFT RFC 2598, June 1999.

[30]. A. Sridharan, S. Bhattacharyya, C.Diot, R. Guerin, J. Jetcheva, and N. Taft, "*On the Impact of Aggregation on the Performance of Traffic Aware Routing*", Technical Report, Department of Electrical Engineering, University of Pennsylvania, June 2000.

[31]. R. Guerin, and V. Pla, "*Aggregation and Conformance in Differentiated Services Networks: A Case Study*", ACM Computer Communications Review, vol. 31, no.1, January 2001, pp.21-32.

[32]. Y. Xu, and R. Guerin, "*Individual QoS versus Aggregaed QoS: A Loss Performance Study*", Technical Report, Department of Electrical Engineering, University of Pennsylvania, July 2001.

[33]. H. Fu and E. Knightly, "*Aggregation and Scalable QoS: A Performance Study*", Technical Report, Department of Electrical Engineering, University of Pennsylvania, February 2001.

[34]. W. H. Cameron, J. Regnier, P. Galloy, and A.M. Savoie, "*Dynamic Routing for intercity Telephone Networks*", Proc. 10th Inte'l Teletraffic Cong. Montreal, Canada, June 1983

[35]. G. R. Ash, "*Use of a Trunk Status Map for Real-Time DNHR*", Proc. 11th Inte'l Teletraffic Cong., Kyoto, Japan, Sept. 1985

[36]. P. Chemouil, J. Filipiak, and P. Gauthier, "*Analysis and Control of Traffic Routing in Circuit-Switched Networks*", Comp. Networks and ISDN Syst., vol. 11, no. 3, Mar. 1986

[37]. K. R. Krishnan and T. J. Ott, "*State-Dependent Routing for Telephone Traffic: Theory and Results*", Proc. 25th IEEE Control and Decision Conf., Athens, Greece, Dec. 1986

[38]. P. Gauthier, P. Chemouil, and M. Klein, "*STAR: A System to Test Adaptive Routing in France*", Proc, Globecom '87, Tokyo, Japan, 1987

[39]. A. Weinrib and G. Gopal, "*Decentralized Resource Allocation for Distributed Systems*", Proc. IEEE Infocom, San Francisco, CA, 1987

[40]. R. J. Gibbens, F. P. Kelly, and P. B. Key, "*Dynamic Alternative Routing-Modeling and Behavior*", Proc. 12th Int'l Teletraffic Cong., Torino, Italy, Jne 1988

[41]. K. R. Krishnan and T. J. Ott, "*Forward-Looking Routing: A New State-Dependent Routing Scheme*", Proc. 12th Int'l Teletraffic Cong., Torino, Italy, June 1988

[42]. K. Mase and H. Uose, "*Consideration on Advanced Routing Schemes for Telecommunication Networks*", Proc. 12th Int'l Teletraffic Cong., Torino, Italy, June 1988

[43]. G. Gopal and A. Weinrib, "*Limited Waiting for Valuable Resources: An Adaptive Control Strategy for Circuit-Swithced Networks*", Proc. 26th Allerton Conf. on Commun. Control and Comp., Monticello, IL, Sept. 1998

[44]. R. J. Gibbens and F. P. Kelly, "*Dynamic Routing in Fully Connected Networks*", IMA J. of Math. Control and Info., vol. 7, 1990

[45]. G. R. Ash, R. H. Cardwell, and R. Murray, "*Design and Optimization of Networks with Dynamic Routing*", Bell System Technical Journal, vol. 60, pp. 1787-1820, 1981

[46]. J. Regnier, P. Blondeau, and W. H. Cameron, "*Grade of Service of a Dynamic Call Routing System*", in Tenth International Teletraffic Cong., June 1983

[47]. Y. Watanabe and T. Oda, "*Dynamic Routing Schemes for International Networks*", IEEE Communications Magazine, vol. 28, pp. 66-69, Oct. 1990

[48]. W. G. Lazarev and S. M. Starobinets, "*The use of dynamic programming for optimization of control in networks of communications of channels*", Engineering Cybernetics (Academy of Sciences, USSR), No.3, 1977

[49]. R. A. Howard, "*Dynamic Programming and Markov Processes*", Cambridge, MA. M.I.T. Press, 1960

[50]. D. P. Heyman and M. J. Sobel, "*Stochastic Models in Operations Research*", Volume II (Stochastic Optimization), New York: McGraw-Hill, 1984

[51]. V. E. Benes, "*Programming and Control Problems Arising from Optimal Routing in Telephone Networks*", Bell System Tech. J., vol. 45, No. 9, pp 1373-1438, Nov. 1966

[52]. J. Regnier, Notes on "*Resource Utilization Costs and State-Dependent Optimal Routing*", 2000

[53]. Z. Dziong and L. Mason, "*Control of Multi-Service Loss Networks*", in Proc. Of the 28th Conference on Decision and Control, Tampa, FL. Dec. 1989

[54]. K. R. Krishnan, "*Adaptive State-Dependent Traffic Routing Using On-Line Trunk-Group Measurements*", in 13th International Teleraffic Cong. Denmark, June 1991.

[55]. K. R. Krishnan, "*Network Control with State-Dependent Routing*," Proc. Int'l Teletraffic Cong. Specialists' Sem., Adelaide, Australia, Sept. 1989