THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

The Department of Computing

Mobile Agent-based Routing for the Next-Generation Internet

by Liu Wai Tung

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Philosophy

September 2007

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____

Liu Wai Tung

ABSTRACT

Abstract of thesis entitled “Mobile Agent-based Routing for the Next-Generation Internet” submitted by Liu Wai Tung for degree of Master of Philosophy at The Hong Kong Polytechnic University in September, 2007.

One of the fundamental issues of the Internet is routing. In general, its purpose is to forward packets over the Internet as efficiently as possible. Currently the Internet is largely based on the well-known TCP/IP protocol suite, in which the transport layer is responsible for the flow control of data communication, while the network layer is responsible for carrying out the routing process. In general, these traditional routing methods, together with the Internet Protocol, have three disadvantages. First, they only offer best-effort delivery service. In other words, no guarantee of support for Quality of Service (QoS). Second, the processing overhead for routing packets is very high since basically each router needs to perform the time-consuming longest prefix match operation. Third, it cannot provide customized routing services.

To address the first problem, a number of research projects are being conducted to support QoS in the Internet. These include Differentiated Services (DiffServ), Integrated Services (IntServ) and Resource reSerVation Protocol (RSVP). In addition, a number of QoS routing protocols are emerging to route packets based on QoS requirements. The second problem has been addressed by the introduction of an evolutionary switching technique, known as Multi-Protocol Switching Protocol (MPLS). This technique binds network

layer routing to data link layer switching. The third issue, the provision of customized routing services, is being tackled by the proposal of a novel network architecture known as active networks. In this new architecture, packets are active and programmable so as to facilitate the introduction of new or customized services. The active packets function like mobile agents and can thus be run by the network nodes to perform various functions.

In recent years, there has been considerable interest in employing mobile agents for various purposes. Initially, mobile agents were mainly used at the application layer but more recently mobile agents have also been employed to address network layer problems like routing. It can be foreseen that mobile agents may be incorporated into the emerging active network framework as a way of providing many innovative services. The aim of this thesis is to explore some of these opportunities.

As an extension to previous work, we have studied an active routing service which uses active packets (packlets) to configure customized communication paths based on the network information. With the aim of minimizing network costs, both a finite horizon and an infinite horizon Markov decision model have been formulated to support active routing.

We have also studied an active MPLS service with bandwidth reservation using a dynamic pricing mechanism. With the aim of maximizing the profits of the network service provider, an optimal bandwidth selling policy has been obtained for both the finite and infinite horizon models based on the Markov decision theory. In both models the optimal policy is threshold-based with the

threshold depending mainly on the amount of remaining bandwidth. It is expected that the Markov decision model will also be adaptable for the analysis of similar bandwidth selling problems in other networking scenarios.

Lastly, we have studied a load balancing problem. We have applied a modified network simplex algorithm. This algorithm can employ the traditional network simplex method not only to find multiple MPLS paths for a user application but also balance the load to obtain aggregate minimum network delay for the user application. Active packets (or mobile agents) can be used to implement our modified network simplex algorithm and to set up the relevant label switched paths.

ACKNOWLEDGEMENTS

Many people have contributed to this thesis. Without their help, I can never complete this thesis. I would like to take this opportunity to say thanks to them. First and foremost, I am very thankful to my supervisor Dr. Henry C.B. Chan of the Department of Computing at the Hong Kong Polytechnic University. I would like to give my sincere thanks for his guidance, comments and effort in regard to every aspect of this thesis.

The project has been supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China under account number PolyU 5085/00E and by the Department of Computing, The Hong Kong Polytechnic University.

I would like to thank all my colleagues at the Hong Kong Polytechnic University. The countless discussion with them has been fruitful and inspiring.

Last but not least, I have been fortunate to have a family that is always supportive for my studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1.
INTRODUCTION

Routing is a major network function that involves the forwarding of packets across a network as efficiently as possible [1]. In connection-oriented networks (e.g., ATMs [2]), communication paths are usually fixed or reserved during call set-up in accordance with the user requirements based on a routing scheme. In connectionless networks (e.g., the Internet), packets are forwarded by routers on a hop-by-hop basis [3] using a routing protocol. The routing table in each router, and thus the communication paths, can be changed dynamically based on traffic conditions. In the context of the Internet, in recent years, we have seen the development of cost-effective distributed router architectures as well as very high-speed routers [4,5]. Furthermore, there has been considerable interest in employing label switching to support routing, which has led to the standardization of Multi-Protocol Label Switching (MPLS) [6,7].

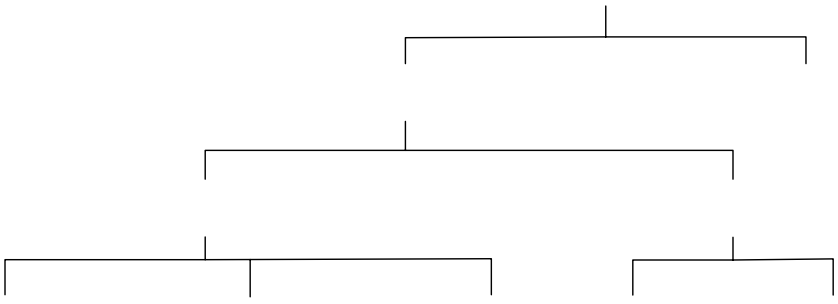


Fig. 1.1 Different types of routing schemes

Fig. 1.1 shows different types of routing protocols in the Internet. Two types of routing protocols are commonly used to support intra-domain routing [8]: the distance vector (e.g., RIP [9]) and link state (e.g., OSPF [10]) routing protocols. For inter-domain routing, policy-based routing protocols, such as BGP [11], and distance vector method, such as EGP, are often used [8]. In recent years, a new type of routing protocol, known as quality of service routing [12], has emerged, which forwards packets based on a particular quality of service requirement. From the end users' point of view, the above routing protocols are "passive" because the end users have very little control over the routing process, other than providing the essential information (e.g., destination address).

With the advent of active networks for providing advanced Internet services [13-16], a new routing paradigm, known as active routing [17-19], has been proposed. Basically this novel routing mechanism enables an end-user application to instruct a network node on how to forward its packets across the network. This is analogous in daily life to the driver of a car who determines how to drive a car through a road network according to the traffic conditions and his/her requirements. In particular, software agents or mobile agents [20, 21] are well-suited to enable active routing, as shown in [22-27]. In [22] and [23] mobile agents are employed to gather link delay information to support QoS routing. As an extension to [22], Oida and Sekido considered not just link delay but also link bandwidth and hop-counts in [24]. In [25], Gonzalez-Valenzuela and Leung investigated a mobile agent-based system called Waves to facilitate QoS routing over an MPLS network. In [26], Migas *et al.* employed both stationary and mobile agents to collect network

information in an ad-hoc network so as to make the routing decision. In [27], Selamat and Omatu applied genetic algorithms in mobile agents to determine the best routes in a network.

Contributing to this relatively new research area, this project proposes a Markov-decision-based Active Routing (MARS) scheme. It is well known that the Markov decision theory can be used to determine the optimal policy for a Markov-based system or model. However to the best of our knowledge, there has been little work to date applying this useful technique to support active routing. Hence this project provides new contributions to this emerging research area.

The use of active routing allows the provision of highly customized and intelligent routing services. This is a paradigm shift from the existing routing services. For instance, while some users prefer to choose high-speed links, others may be more concerned with reliability. Furthermore, in making the routing decision, some users may want to take into account the session duration. It is very difficult if not impossible to address these requirements in the existing passive routing paradigm but active routing provides an effective solution. As complements to each other, passive and active routing services can work with other networking technologies such as MPLS to provide innovative and intelligent services in the next-generation network. In general, a user can use his/her own active routing algorithm through the active packets. The active packets can also be used to configure customized forwarding tables at the network nodes for packet forwarding purposes. To support active routing, this project presents a new scheme called MARS in Chapter 3. The analysis

can provide valuable insights into the design of active routing services for the next-generation network.

Recent years have seen the development of Multiprotocol Label Switching (MPLS) [6] for forwarding Internet traffic efficiently using label switched paths (LSPs). Essentially, label edge routers (LER) and label switch routers (LSR) are employed for attaching MPLS headers/labels and transferring MPLS packets, respectively. Integrating with a bandwidth reservation protocol, MPLS can also be used to facilitate Quality of Service (QoS) routing [12]. Moreover, based on active network technologies [14,28] (especially active routing and agent-based routing [17,24,25]), it would be useful to develop an active MPLS service that can reserve bandwidth according to an adaptive pricing method. In particular, it is envisioned that both active and non-active (passive) network services can be integrated in an advanced network called ISDN3 [19,29]. Chapter 4 presents an adaptive bandwidth selling method. In particular, the bandwidth selling policy is determined based on a Markov decision model. For the purposes of evaluation, we will compare the selling policy with two other schemes.

The distribution of MPLS labels will set up paths known as label switch paths (LSPs) along the network, and the setting up of these LSPs can facilitate traffic engineering which involves finding suitable paths (QoS routing) and assigning suitable load in each of these paths (load balancing) if there is more than one LSP between the same source-destination pair. In Chapter 5, we apply a modified network simplex algorithm in the ISDN3 [29] network which can employ the traditional network simplex method to not only find multiple

MPLS paths for a user application, but also balance the load to obtain aggregate minimum network delay for the user application. In ISDN3, active packets can be used to implement our modified network simplex algorithm and to set up the concerned LSPs as in [30]. Preliminary results comparing the proposed scheme with other simplex schemes are also presented.

Chapter 2.
RELATED WORK

In this chapter, we will review some related literature, including conventional routing protocols, active networks, mobile agents, MPLS and ISDN3. We will also give a survey of some of the works which employ mobile agents in network routing.

2.1. Conventional Routing Protocols

In general, the Internet is divided into a number of administrative domains. Some routers know how to forward packets only to destinations within the same administrative domain. Others can interface between the administrative domain they belong to and other administrative domains. The former type of router conforms to intra-domain routing protocols while the latter type of router has to support inter-domain routing protocols. Before discussing the novel mobile agent-based routing approach, it is of interest to first discuss some of these routing protocols.

2.1.1. Intra-domain Routing Protocols

The three most commonly used intra-domain routing protocols are distributed (there is no central server that computes the routes), hop-by-hop (the routing decision is local) and dynamic (routes can be changed at different times). There are two main types of intra-domain routing protocols: distance vector and link state routing protocols. We will describe each of these in the

following.

2.1.1.1. Distance Vector Routing Protocols

Routers that conform to distance vector routing protocols do not have complete topological information. Each network node learns the costs of its directly connected links, and each network node will construct a one-dimensional array containing the costs to all other network nodes. Each node will transmit its cost vector only to its neighboring nodes. On receiving a cost vector from its neighboring node, the network node will, on reaching a particular destination, update its routing table whenever there is a change in the costs. Eventually, each network node will have a converged routing table. The update processes can be illustrated as in the example network in Fig 2.1.

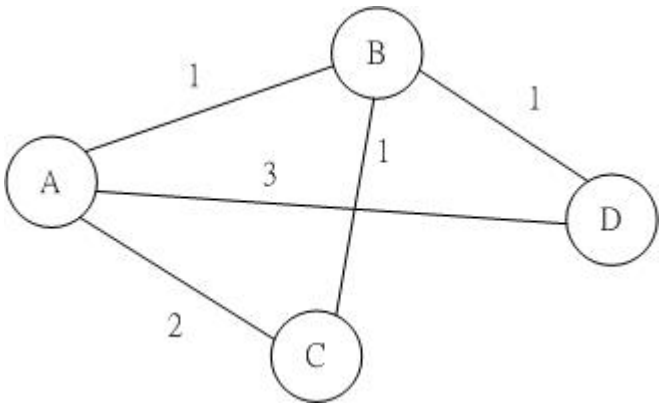


Fig. 2.1 An example network

We denote the cost vector of each network node by (*w*, *x*, *y*, *z*), meaning that the cost of reaching node A, B, C and D from this node are *w*, *x*, *y* and *z* respectively. The initial routing table of node A is given in Table 2.1. When it receives the cost vector from node B, which is (1, 0, 1, 1), it learns that node B can reach node D with cost 1. Since it can reach node B with cost 1, its cost on

reaching node D will be 2 if it routes the data packets through node B. So it will update its routing cost to node D to 2 accordingly. The updated routing table of node A is given in Table 2.

Destination	Cost	Next-hop
B	1	B
C	2	C
D	3	D

Table 2.1 Initial routing table of node A

Destination	Cost	Next-hop
B	1	B
C	2	C
D	2	B

Table 2.2 Updated routing table of node A

On routing a packet across the network, the network node will examine the packet's header to identify its destination. The network node will then route the packet to the next-hop as indicated in the routing table. As a result, packets with the same destination will follow exactly the same path. Examples of distance vector routing protocols are RIP and RIP-2.

2.1.1.2. Link State Routing Protocols

Routers that conform to link state routing protocols have complete topological information. Similar to distance vector routing protocols, each network node also learns the costs of its directly connected links. Yet, the cost vector of each network node will be flooded to all the other network nodes in the same domain. Thus in calculating the routing table, each router will make use of all the cost vectors of the network nodes within the same network. All the routers have global knowledge. Since each router knows exactly the network topology, the construction of routing table, which is equivalent to finding a shortest path

to all other nodes, can be found by using Dijkstra's algorithm. Examples of link state routing protocols are OSPF and OSPF-2.

2.1.2. Inter-domain Routing Protocols

Since different domains will have different routing policies, it is necessary when computing inter-domain paths to consider policy related constraints. As a result, inter-domain routing adopts policy routing.

2.1.2.1. Policy Routing Protocols

Policy routing protocols take into account policy related constraints when computing the required paths. The whole network is divided into several administrative domains. Each administrative domain is governed by an autonomous administration. Thus different administrative domains will have different routing policies, such as the QoS it is able to deliver, the cost associated to provide such a QoS, etc. There are policy gateways in each administrative domain. They will advertise the policy related constraints information explicitly to neighboring administrative domains such that when a packet reaches the policy gateway of an administrative domain, the policy gateway can choose the appropriate administrative domain the packet should traverse next. In the context of the Internet, the most commonly used policy routing protocol is BGP4.

2.2. Active Networks

Traditionally, the introduction of new network services like routing protocols is a very long process and, because the present network services are hardware

specific, it also requires the cooperation of the hardware vendors to increase the pace of deployment of a new network service. Moreover, it is extremely difficult if not impossible to provide customized network services. To address these issues, researchers propose the active networks paradigm which allows network users to "program" the network. Under this new paradigm, network nodes are no longer passive. Rather, they are active and programmable. By providing a program that can be executed in the network nodes, network users can have customized control of how their packets are processed by the network nodes. This can greatly increase the pace of the introduction of new network services because one only needs to provide the program that carries out the new service. And on top of that, more effective solutions to existing networking problems can be provided.

Generally speaking, there are two approaches on which active networks are based: the programmable switch approach and the programmable packet approach. In the programmable switch approach, a user of active network can send a customized program to the network node and subsequent packets can be processed according to the program. In the programmable packet approach, each packet carries a program with it to specify what and how the processing is carried out on it.

As described in [13], all the network nodes in an active network are known as active nodes. An active node consists of different execution environments (EEs), and there is a node operating system (NOS) responsible for managing the operations and resources allocation of these EEs. Different active applications can be run or executed in the most suitable EE. In particular,

mobile agents can be used to realize this active network paradigm. The next section will give an overview of relevant mobile agent technologies.

2.3. Mobile Agents

Agents are commonly defined as software programs that work on behalf of the user or another entity [31]. Mobile agents are agents that can move across different systems within the network to perform tasks like searching for information and facilitating decision making. Mobile agents can function independent of each other or cooperate to solve problems in telecommunications and network management.

There are quite a few mobile agent models being developed currently. Aglet closely follows the applet model of Java to allow mobile Java objects to be executed in Aglet-enabled hosts in a computer network. Concordia is another mobile agent model that is built on Java. A Concordia agent is regarded as a collection of Java objects and can have states information stored internally and externally. Mole is an agent modeled as a cluster of Java objects, which can communicate with other agents through a defined communication mechanism.

In existing server-side and client-side programming paradigms, the client processes usually run on remote machines and communicate with server processes to perform work. This can generate a high level of network traffic and may result in congestion delay. Mobile agents can be used to complement the existing server-side and client-side programming paradigms, and the use of mobile agents in data networks will create a paradigm shift from

client/server-side programming to network-side programming. Besides bringing the requesting client closer to the server, and hence reducing network traffic, users can also have more control of the network. Before migrating into the active network paradigm, the co-existence of passive network functions and the ability to program the network by using mobile agents can create many innovative services. In particular, MPLS is a label switching technology that is suitable for use in conjunction with mobile agents to support active MPLS services (which will be discussed in Chapter 3). The next section will give an overview of the MPLS technology.

2.4. MPLS

When a packet enters a network, the routers will analyze the packet's header and determine the next hop for the packet according to the routing protocol. In general, choosing the next hop can be considered as classifying the packet into a particular group, and different groups will be mapped to different next hops. Each of this group is known as a forwarding equivalence class (FEC).

In a conventional IP network, a router will perform the longest prefix match operation when a packet traverses it. If two packets are matched as having the same prefix in the router's routing table, they will be regarded as having the same FEC. As a result, they will be forwarded to the same next hop. This process repeats as packets traverse the next router. Thus two packets with same destination will be forwarded identically and follow the same path. In order to provide different classes of service, packets must be marked with special labels as they enters the network. Subsequently, the packets will be

treated according to the labels. This is the rationale of the MPLS technology.

In MPLS, the mapping of FEC with a particular packet is done as the packet enters the network. The assignment of FEC is done just once. In a MPLS network, routers at the edges are known as label edge routers (LERs). On the ingress side, an LER is responsible for mapping a FEC to an entering packet. Unlike a conventional IP network, the analysis of the packet's header can focus on not only the packet's destination address, but also other information such as the source address, application port number, etc. Once an FEC is assigned, the packet is attached with a short fixed length header known as label. The packet will then be forwarded to the next router which is known as a label switch router (LSR). It is not necessary for the LSR to examine the packet's header again. Instead, the LSR only has to examine the label and the label is used as an index to find the next hop in a switching table. It then performs a label swapping function and forwards the packet to the next LSR. The examination and swapping of labels will continue until the packet reaches the egress side of the MPLS network. When the packet reaches the LER on the egress side of the MPLS network, the LER will remove the label and the packet will be forwarded according to the traditional IP protocol. As a result, the sequence of labels determines the path the packet traverses. The path established between the LERs and the LSRs is known as the label switch path (LSP). Fig. 2.2 shows how an LSP is setup in an MPLS network. By setting up different LSPs, different packets can traverse different paths even if they have the same destination. Moreover, we can take into account different traffic characteristics like path traffic load and dropped packet percentage during the path setup.

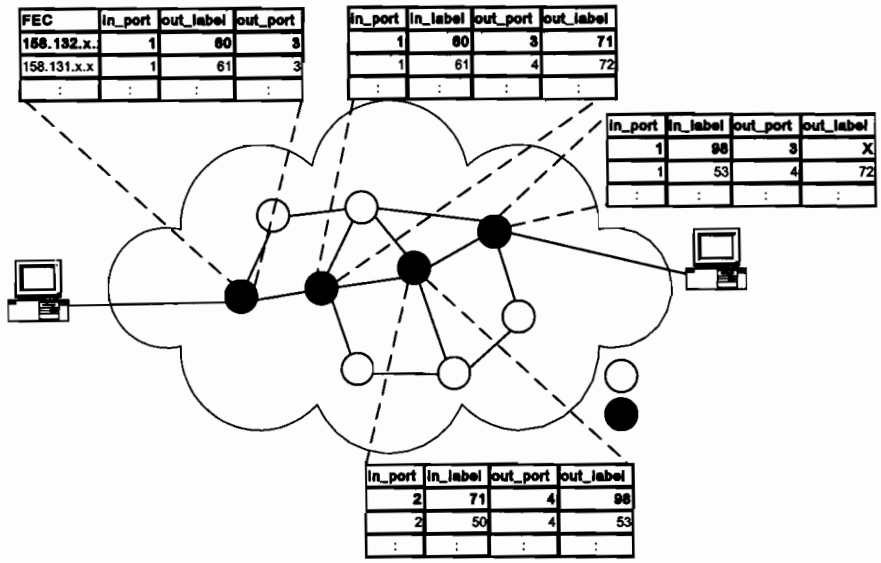


Fig. 2.2 An example LSP in an MPLS network

In a MPLS network, all the forwarding decisions are driven by labels. There are a number of advantages in this forwarding paradigm. First, the time-consuming analysis of the packet's header is only done once at the ingress LER. The subsequent forwarding is only driven by the fast label lookup and swapping. Second, the assignment of FEC is no longer confined to the destination address and it is possible to make more complicated assignments. Third, customized LSPs can be set up before a packet enters the network. The traversed path of the packet can be assured and QoS routing is much more easily achieved in MPLS.

2.5. ISDN3

In the previous century, two generations of Integrated Services Digital Network (ISDN) were developed to build the telecommunication infrastructure. The first generation of ISDN (ISDN1) was developed in the 1980's to integrate narrowband voice/data/video traffic over digital telephone

systems [32]. The second generation of ISDN (ISDN2) known as Asynchronous Transfer Mode (ATM) was introduced in the 1990's to support broadband services by means of cell switching [33]. Recently, there has been a growing interest in developing the third generation of ISDN (ISDN3) [34]. The view presented in [34] is that the convergence of ATM, the Internet and active networks will form the basis for this next-generation network. In ISDN3, not only network traffic, but also network functions are integrated, thus creating a novel next-generation network. In essence, the network is partitioned to support a spectrum of traditional forwarding functions (e.g., switching) as well as emerging intelligent network services (e.g., active network services). This is known as horizontal partitioning, which complements the conventional vertical layering networking model. To achieve this, an integrated traffic-forwarding device, called the Forwarding EnginE (FEE) is employed. Essentially, various processing modules (e.g., switching module, active network module) can be installed in a FEE under a flexible and modular architecture.

Active routing can be implemented in ISDN3 by making use of its active network module, which is presented in [19]. One type of packet, called the packlet, is active and programmable and can be used to support the active routing service in ISDN3. A packlet functions like a software agent in general or a mobile agent in particular. A general framework of active routing which supports switching over new paths is discussed in [19], and we extend the work presented in [19] in Chapter 3 to employ MPLS to investigate how active routing can actually be implemented.

2.6. Related Literature: Mobile Agent-based Routing

As mentioned above, mobile agents can move across different systems within the network to search for network information. As a result, different types of information can be collected when making routing decisions. In addition, network data are collected only when needed. This is different from the traditional routing protocols which will trigger an update of routing data whenever there are changes. This can reduce the number of update messages in the network so that the bandwidth can be better utilized.

There is a considerable literature describing the employment of mobile agents to assist network routing. In [22] mobile agents are employed to gather link delay information to support QoS routing. In [24], Oida and Sekido extended the work in [22] so that suitable paths can be found based not just on link delay but also link bandwidth and hop-counts. In [35], mobile agents are used to discover multiple optimal or near optimal paths in a network. In [25], Gonzalez-Valenzuela and Leung investigated a mobile agent-based system called Waves to facilitate QoS routing over a MPLS network. In [36] mobile agents are used as representatives of Autonomous Systems to exchange and communicate their autonomous systems' routing policy. Here we describe in detail the work presented in [22], [24], [35] and [25] and discuss the results.

2.6.1. AntNet

The mobile agent-based routing algorithm presented in [22] is AntNet. AntNet is inspired by the behavior of ant colonies. In AntNet, each mobile agent

behaves like an artificial ant that will build a path from its source to its destination. It collects information about the links it traverses such as the time required to traverse the links and the load status of the links. Link information will then be used to modify the routing tables of all the visited nodes when another ant is moving in the opposite direction from the destination to the source, with an attempt to setup routes with the least average packet delay and maximum throughput.

The AntNet algorithm is executed by mainly two types of mobile agents: forward-traveling ants and backwards-traveling ants. The details of the algorithm are described as follows:

1. At regular time intervals, each network node will launch a forward ant with randomly chosen destination.
2. The forward-traveling ant will choose the next hop according to the information stored in the current routing table of each node. The routing table contains probabilities of choosing a node as the next-hop for a particular destination, the average delay times when an agent is sent from this node to all the other nodes, and the standard deviation of the delay times. The probabilities represent the goodness of choosing a node as the next-hop, and are determined by the mean and standard deviation of the delay times. The route is selected randomly according to the goodness probabilities and together with an exploration probability indicated by the user.
3. As the forward-traveling ant traverses towards the destination, it will keep track of the nodes visited and the traversal time.
4. When the destination is reached, a backward-traveling ant is launched.

The backward-traveling ant will traverse exactly the same path which is traversed by the forward-traveling ant, but in the reverse direction.

5. As the backward-traveling ant visits a node, it will update the goodness probabilities of the next hops, mean and the standard deviation of the delay time from this node to all the nodes along the forward ant's path using the traversal time recorded by the forward-traveling ant.

The update of the routing table is triggered by the travel time experienced by the forward-traveling ant. This is crucial since the data packets will share the same path.

AntNet is evaluated by comparing it with other Internet standard routing algorithms like OSPF and Bellman-Ford. Even in networks with delays that involve great statistical fluctuations have little effect on the performance of AntNet.

2.6.2.ARS

The work presented in [24] is an extension of the work in [22]. The system is named ARS. It is also a mobile agent-based routing system. It considers not only delay time, but also bandwidth and hop-count in constructing the goodness probability of choosing each network node as the next hop. In order to consider the newly added constraints, ARS restricts the actions of the forward-traveling ants by:

1. only using links whose unreserved amount of resource is not less than a predefined bottleneck bandwidth, and
2. terminating the execution of any ants which exceed a maximum

hop-count.

If a forward-traveling ant reaches the destination, the path that it traversed will satisfy both the bandwidth and hop-count constraints.

Another contribution of the ARS system is the consideration of global admission controls. A global admission control is essential in increasing the efficiency of network resources allocation and attempts to maintain a high resource utilization rate even when the request arrival rate is high. It implements efficient admission control by utilizing the link state information gathered by the backward-traveling ants. A path can be considered to be a collection of links. If all the links of the path possess resources of an amount greater than that required by the path, the path can be regarded as a feasible path. Each node in the ARS caches a set of feasible paths from itself to all the other nodes.

In [24], $P(s, d, m)$ denotes a cached path from node s to node d with a bandwidth not less than m units and $s(n1, n2, m)$ denotes a state indicating whether link $(n1, n2)$ has at least m units of bandwidth. ARS can determine whether a cached path is feasible or not by the following methods:

1. When a backward-traveling ant leaves a node k, it examines all the outgoing links of that node and records the number of unreserved resources of all the links. For example, if the outgoing links (k, l) has r units of unreserved resources, the backward ant will record the tuple (k, l, r).
2. When a backward-traveling ant enters a node s, it will examine all the

cached paths that contain the outgoing links recorded in 1. For example, we consider a cached path $P(s, d, m)$ that contains the link (k, l) which is recorded in 1. If r is greater than or equal to m, s(k, l, m) will be set to 1 and 0 otherwise.

3. When the backward-traveling ant returns to the source, it will update the cached path $P(s, d, m)$. It will also update the link states as described in 2.

ARS can provide QoS quarantees to user applications as it updates the set of feasible paths on a regular basis. It decreases the possibility of suffering congestion in the midway of the path traveling towards the destination.

2.6.3. Multiple Ant-Colony Optimization for Network Routing

In [35], Sim and Sun investigated the use of mobile agents to support network routing in a scheme that was analogous with the behavior of ants in colonies. The paper modeled the behavior of ant in which, as they travel, ants lay down a pheromone as a chemical marker which other ants of their colony can then use to detect changes in the environment.

In the context of network routing, each mobile agent can be regarded as an ant. The destination can be regarded as a food source and there are a number of paths to the destination. A number of ants will be sent to the destination at regular intervals. Initially, each ant will randomly choose a path towards the destination. It will lay down a pheromone signal when it travels on the path. When the ant reaches the destination, it will choose a path that has the greatest

amount of pheromone to travel back to the source. Since a better path should have a smaller value in trip delay, the fastest ant will follow the same path back to the source. It will also add its pheromone marker to the path on its traversal back to the source. As a result, all other ants will travel back to the source along the path that has the smallest trip delay value. As the path becomes more and more popular, its trip delay will increase. The amount of pheromone on the other paths will then increase and packets will be motivated to choose other paths.

This ant colony algorithm identifies only one optimal path but Sim and Sun's work tried consider the issue of load balancing as well. Consider that ants from different colonies will lay different types of pheromone. The presence of a pheromone will affect only the behavior of other ants within the same colony. So [35] considered more than one ant colony in finding optimal paths. The processes in finding the optimal path for each ant colony are the same as that in using one ant colony. Since initially each of the ants will choose a path at random, there is a high probability that the resulting optimal path for each ant colony will be different. Data packets can thus be sent evenly between the optimal paths.

2.6.4. QoS Routing for MPLS Networks Employing Mobile Agents

Gonzalez-Valenzuela and Leung tried to integrate MPLS, DiffServ and a special type of mobile agent named waves to provide QoS routing in the Internet in [25]. They recognized that the use of FECs in MPLS networks can be used to support aggregation of data streams in DiffServ networks, which

supports prioritized service for different data streams. Some of the data streams may be partitioned into the same FEC, thus following the same route. Inside the MPLS network, data streams of several FECs may be further aggregated into a single FEC if they share the same labels in some intermediate LSRs. This reveals that an efficient use of FECs to provide QoS routing requires the setup of multipoint-to-point (mp2p) connections. It is the mobile agent that is responsible for gathering network information and setup the required mp2p paths by the means of setting up the label distribution inside the MPLS network.

The wave agents have two missions: they are responsible for monitoring the available network resources and discovering the mp2p paths that satisfy the QoS requirements. For the mission of monitoring, there is an agent residing at every network node. It gathers QoS information from the MPLS switch and it will assign a weighted scalar value to the network links attaching to the node. The value can be a representation of the magnitude of a certain QoS metric or a combination of a group of QoS metrics. Each agent keeps the values of each link. Each agent can be regarded as a virtual node of a knowledge network with weighting of each of the links recorded. The advantage of this approach is that the information is recorded in a distributed manner and this can reduce management of network traffic.

Given the knowledge network is established, the wave agents can then configure the mp2p paths. Since mp2p paths can be regarded as a tree connecting all the ingress nodes and the egress nodes, the problem of finding the mp2p paths is the same as the constrained Steiner Minimal Tree (SMT) in

graph theory. The basic operations of the agents are outlined as follows:

1. Several ingress nodes request a route towards a common egress node. The requests can be combined and the objective is to create a mp2p tree connecting the ingress nodes to the egress node.
2. Each network node clones the original agent and launches as many copies of it as the number of QoS compliant outgoing links the node has.
3. On reaching an intermediate node, the agent will record the distance traveled by it so far in a local variable in the node if no agent from the same origin has visited the node before.
4. If an agent from the same origin has visited the node before, the agent can only continue to travel towards the destination if its traveling distance is shorter than the value recorded in 3. The agent will also update the traveling distance to this lower value. Otherwise, the agent will be discarded.
5. A second set of agents is similarly launched, and an agent can continue its traversal towards the destination only if its traveling distance equals that recorded in 4. This step is necessary in finding all, not just one, of the shortest paths.
6. A third set of agents is launched but this time only from the ingress nodes. Upon reaching an intermediate node, the agent will first check if other agents from the same origin have visited it before. If the agent is the first to visit this node, it will increment a variable indicating the number of visits by agents from different origins. Otherwise, it continues to traverse along the path. This step tries to identify possible nodes where data merges.

7. When all the agents reach the destination, they are sent back to the origin nodes over the shortest paths they have traveled. The agents will credit the shortest paths they traveled by some weight as they traverse along the path.
8. On reaching an intermediate node, an agent checks the number of visits by agents from different origins. More credit will be given to the path if the number of visits is greater. Thus the credit of a path is directly proportional to the degree of overlapping of that path.
9. All the agents will record their traversed path and the associated weight at the destination node. A traversed path is updated if its associated weight is greater than the previous recorded one.

As a result, the egress node will have a record of mp2p tree that satisfies the QoS constraint specified with the highest degree of aggregation and thus minimal cost.

As these works show, mobile agents can be implemented in various ways to support network routing, like collecting up-to-date network information and configuring present network protocols. However, none of the works presented above provide a mechanism for users to choose their own path towards the destination. All the works concentrate on the search for an optimal path which is defined by the network operators. Yet, in a scenario where active routing is possible, a user may consider other factors in choosing his own optimal path. To implement active routing in an MPLS network, mobile agents are well-suited to configure customized LSP. To prevent wastage of network resources, the bandwidth along the configured LSP should be reserved through

a pricing mechanism. In the next chapter we will present our approach to employing mobile agents in the ISDN3 to support a novel active MPLS service with bandwidth reservation.

Chapter 3.
MARKOV- DECISION-BASED ACTIVE ROUTING SCHEME (MARS) FOR THE NEXT GENERATION NETWORK

3.1. Introduction

Active routing is an emerging routing mechanism that enables an end-user application to direct the network on how to forward its traffic. This is accomplished by using active packets to establish a customized communication path dynamically. This intelligent routing method can also be used in conjunction with existing networking technologies (e.g., Multi-Protocol Label Switching) to provide effective and flexible forwarding services. Based on both an infinite horizon Markov decision model, this chapter presents a Markov-decision-based Active Routing Scheme (MARS) for active networks in general and the next-generation network called ISDN3, in particular. Our aim is to determine the active routing policy so as to minimize the network cost for a user's session. Theoretical analysis is presented to show the advantages of MARS as compared with several other schemes. The analysis provides valuable insights into the design of active routing services for the next-generation intelligent network.

3.2. Overview of ISDN3 and Active Routing

In the previous century, two generations of Integrated Services Digital Network (ISDN) were developed for building the telecommunication infrastructure. The first generation of ISDN (ISDN1) was developed in the 1980's to integrate narrowband voice/data/video traffic over digital telephone systems [32]. The second generation of ISDN (ISDN2) known as Asynchronous Transfer Mode (ATM) was introduced in the 1990's to support broadband services by means of cell switching [2,33]. As we enter the 21st century, it is an appropriate time to initiate research the development of the third generation of ISDN (ISDN3) [34].

Table 3.1 compares ISDN1, ISDN2, ISDN3 and the Internet. In our view, the convergence of ATM, the Internet and active networks will form the basis for this next-generation network. In ISDN3, not only network traffic, but also network functions are integrated, thus creating a novel next-generation network. In essence, the network is partitioned to support a spectrum of traditional forwarding functions (e.g., switching) as well as emerging intelligent network services (e.g., active network services). We call this horizontal partitioning, which complements the conventional vertical layering networking model. To achieve this, an integrated traffic-forwarding device, called the Forwarding EnginE (FEE) is employed. Essentially, various processing modules (e.g., switching module, active network module) can be installed in an FEE under a flexible and modular architecture.

	ISDN1 (Circuit-switched ISDN and Frame Relay)	**ISDN2 (ATM)**	**ISDN3 (Next-generation network)**	**Internet**
Base network	Connection-oriented	Connection-oriented	Connectionless*	Connectionless
Transmission units	Frame	Cell	Combination of passive packet and active packet (packlet)	Datagram
Traffic forwarding device	Switch	Switch	FEE	Router
Network nature	Non-active with little intelligence	Non-active with more intelligence	Active with high intelligence	Non-active with little intelligence
Basic design philosophy	Integrate narrowband traffic over the digital telephone system	Integrate narrowband and broadband traffic over a cell-switching network	Integrate both network traffic and network functions over the same network	Provide best-effort transport service

Table 3.1 Comparison of ISDN1, ISDN2, ISDN3 and the Internet

Fig. 3.1 shows the basic network architecture of ISDN3 for supporting active routing. We propose that a one-byte label be attached to each packet to identify the packet type (e.g., active packets, MPLS packets). One type of packet, called the packlet[1], is active and programmable. In particular, it can be used to support the active routing service in ISDN3. To a certain extent, a packlet functions like a software agent in general or a mobile agent in particular.

[1] Packlets are Java-based to facilitate portability and interoperability. The proposed name is in line with the Java terminology where servlets, applets and packlets are for server-side, client-side and packet-oriented applications, respectively.

Hence, for example, the widely used Java-based Aglet development tool [20] can be extended and enhanced to develop packlets. When an FEE receives a packet, it will be passed to the respective module for processing based on the packet type specified in the label. For instance, a packlet and an MPLS-based packet are transferred to the active network module and MPLS module for self-execution and switching, respectively. Furthermore, a hybrid service can also be realized. For example, a communication path can be established by using a packlet. Subsequent data packets are sent along the path by means of label switching. As shown in Fig. 3.1, each FEE keeps both a private and a public label switching table to support active routing and MPLS services, respectively. In both tables, an incoming label of an incoming port is mapped into an outgoing label of an outgoing port. This mapping thus defines a path for packet transfer. A packlet can manipulate the corresponding entry in the private label switching table through the active module. In each table entry, a secret is used for authentication purposes (i.e., a packlet can only change the information if the secret is provided). In summary, we envision that as the next-generation network becomes more powerful, the current server-side and client-side programming paradigms will be extended to the network-side, thus allowing users to control part of the network. The co-existence of active and passive network functions will create many innovative and intelligent network services and active routing is just one of them.

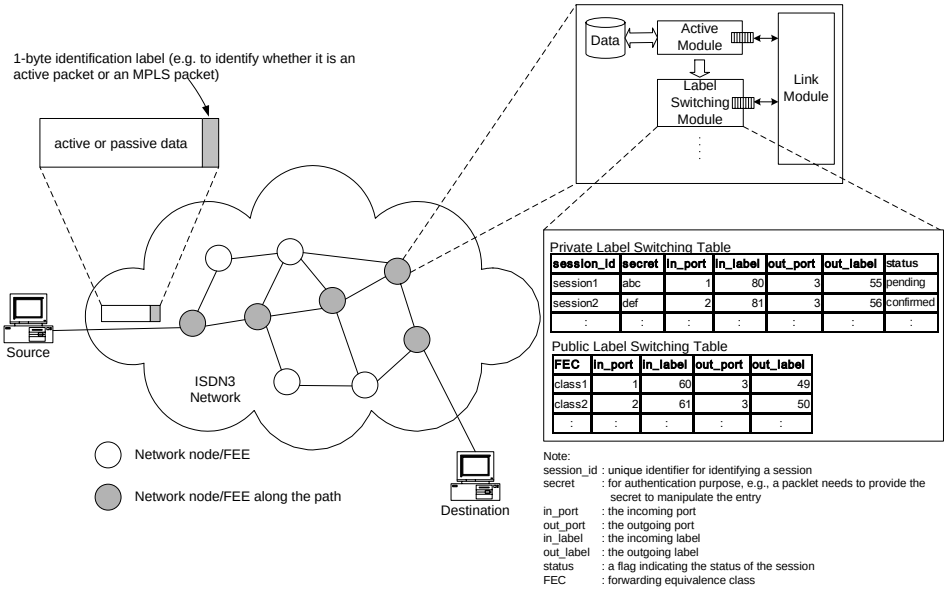


Fig. 3.1 Overview of ISDN3 and active routing

In active routing, an end-user application tells the network, or more specifically the FEEs, how to forward its packets. To do this, the network provides a Network Object Model (NOM) to allow the packlets to manipulate the network information. This is similar to how JavaScript manipulates a Web browser using the well-known Document Object Model [37]. Basically, the NOM provides a tree-like interface for packlets to access the network information (e.g., bandwidth, reliability, connectivity, etc.) used to make the routing decision. Here, we assume that the network cost, and the associated path for sending packets from one node to any of the other nodes, can be retrieved from the NOM. Such information is updated regularly, at the beginning of each routing cycle. The basic operations of the active routing services are as shown in Fig. 3.2.

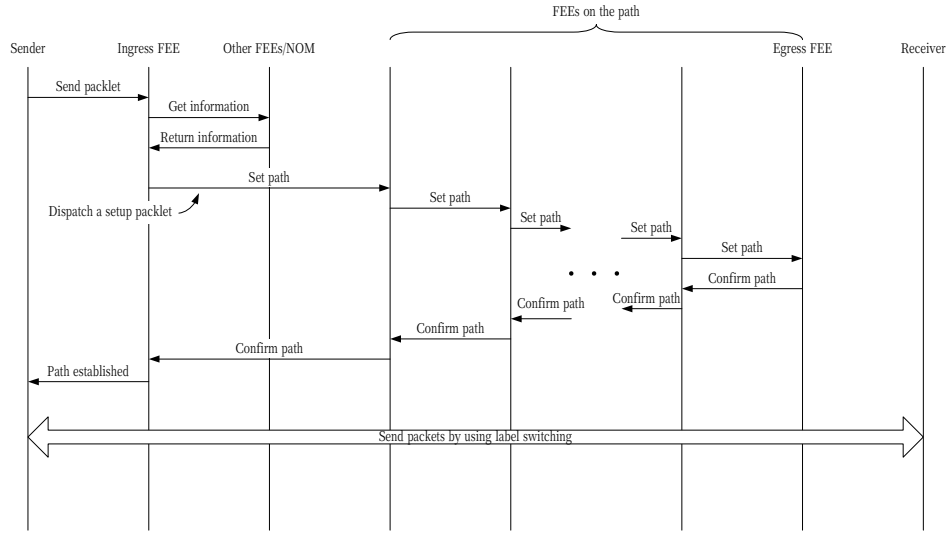


Fig. 3.2 Basic operations of MARS in setting up a communication path

At the beginning of a communication session, the sender's application sends a master packlet to the ingress FEE to get the network information (e.g., possible paths, cost information, etc.) from the NOM. Note that this master packlet will reside in the ingress FEE to manage the path on behalf of the user. If necessary, the master packlet may dispatch further packlets to other FEEs to obtain the required network information (e.g., like the knowledge network described in [25]). Based on the network information and possibly working in conjunction with the sender's applications, the master packlet determines the best possible path to the destination. Having determined the target path, a setup packlet is sent to propagate through the respective FEEs to update the forwarding tables. If the path can be set up successfully, the packlet returns to the previous FEEs to confirm the path establishment. Once the path is established, subsequent data packets can be forwarded by means of label switching based on the forwarding tables (i.e., by mapping an incoming label to an outgoing label as mentioned above). At the beginning of each routing

cycle (or whenever necessary), the master packlet determines whether there is a better communication path according to the specified routing policy. Should a better path be available, it can reconfigure the communication path and remove the old path. Similar to the set up procedures, the basic operations for reconfiguring a path is shown in Fig. 3.3.

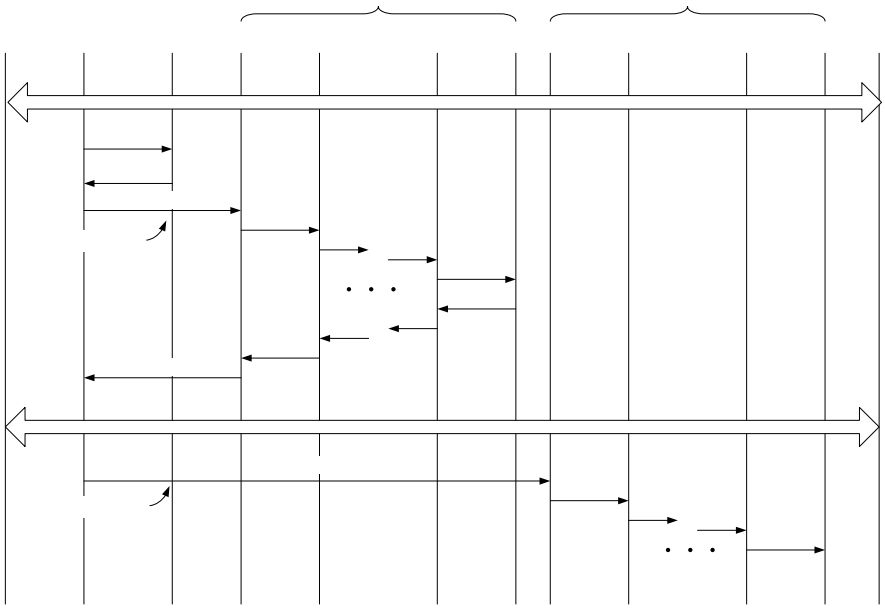


Fig. 3.3 Basic operations of MARS in switching to a new path

In essence, once a path is established, the data packets will be forwarded by the high speed label switching and thus the overheads incurred by the routers are relatively low. Each application has a master packlet which acts as the coordinator of the application, mainly making the rerouting decision on behalf of the application. The update of network information can also utilize mobile agents like the knowledge network described in [25] in a distributed manner with minimized overheads. The self-explanatory pseudo codes for implementing these operations are shown in Fig. 3.4. In the next section, we formulate two Markov decision models to determine the routing policy so as to minimize the average network cost for a communication session.

Sender Ingress FEE Other

```
// Master packlet (stationary) for managing the session
when the packlet arrives at the ingress FEE {
      obtain relevant information from the NOM/other FEEs
      choose a suitable path that can best satisfy the user's requirements
      create a setup packlet with necessary information to set up the path
      set the status of the setup packlet to be "Setup" and dispatch it
   wait for the setup packlet to return
   carry out necessary actions based on the result
}

for every routing cycle or whenever necessary as defined by the user {
      obtain relevant information from the NOM/other FEEs
      run the routing algorithm (e.g., MARS) to determine if there is a better path
      if a better path is found {
            create a setup packlet with necessary information to set up the path
            set the status of the setup packlet to be "Setup" and dispatch it
            wait for the setup packlet to return
            carry out necessary actions based on the result
            if the status of the setup packlet is "Success" {
                  dispatch a deletion packlet to remove the old path
            }
      }
}

// Setup packlet (mobile) for establishing the required path
when the packlet arrives at a FEE {
      if the status of the packlet is "Setup" {
            if the packlet is at the egress FEE {
                  setup the forwarding table with relevant information such as suitable labels, session number, secret, etc
                    if successful {
                        set the status of the table entry to be "confirmed"
                        set the status of the packlet to be "Success"
                  } else {
                        set the status of the packlet to be "Failure"
                  }
                  dispatch the packlet to the previous FEE along the path
            } else {
                  setup the forwarding table with relevant information such as suitable labels, session number, secret, etc
                  if successful {
                        set the status of the table entry to be "pending"
                        dispatch the packlet to the next FEE along the path
                  } else {
                        if the packlet is at the ingress FEE {
                              inform the master packlet of the failure of the path setup
                        } else {
                              remove the corresponding entry in the forwarding table
                              set the packlet status to be "Failure"
                              dispatch the packlet to the previous FEE along the path
                        }
                  }
            }
      } else if the status of the packlet is "Failure" {
            remove the corresponding entry in the forwarding table
            if the packlet is at the ingress FEE {
                  inform the master packlet of the failure of the path setup
            } else {
                  dispatch the packlet to the previous FEE along the path
            }
      } else if the status of the packlet is "Success" {
            update the status of the corresponding table entry to be "confirmed"
            if the packlet is at the ingress FEE {
                  inform the master packlet that the path setup is successful
            else {
                  dispatch the packlet to the previous FEE along the path
            }
      }
}

// Deletion packlet (mobile) for removing the label information in the original path
when the packlet arrives at a FEE {
            remove the corresponding entry in the forwarding table by providing the required secret
            if the packlet is at the egress FEE of the original path {
                  confirm the deletion and dispose the packlet
            } else {
                  dispatch the packlet to the next FEE along the original path
            }
      }
}
```

Fig. 3.4 Pseudo codes showing the key operations of the packlets

3.3. Markov Decision Models For Active Routing

This section presents two Markov decision models: finite horizon (previous work) and infinite horizon (new contribution) to support the active routing service. For ease of reference, Table 3.2 summarizes the symbols and notations used in the models.

Table 3.2 List of symbols used in the models

t	The t-th decision point
T	The set of decision points
C_{switch}	The switching cost
c_j	The j-th network cost
N	The probability distribution of the network cost. $N(c_j)$ gives the probability when the network cost is c_j
a	The action
μ	The mean of the network costs
σ	The standard deviation of the network costs
δ	A value determining the sampling width of the network costs
Δ	Half of the width of the interval between consecutive network costs
C	The expected cost
Finite horizon model	
d	The duration of a communication session
S_t	The set of system states at decision point t
(c, s)	The system state, where c is the reserved cost, s is the cost saving

$A_{(c, s),t}$	The set of available actions at decision point t with system state being (c, s)
$A^*_{(c, s),t}$	The preferred action for state (c, s) at t
$K_t(c, s, a)$	The cost for the t^{th} routing cycle if the system state is (c, s) and the chosen action is a
$T((k, l)\|(i, j), a)$	The state transition probability from the state (i, j) to state (k, l) if the chosen action is a
$v_t(c, s)$	The minimum expected accumulated cost for routing cycles $t, t + 1, \ldots, d + 1$
$\phi_t(c, s, a)$	The expected cost from decision point t to the end of the communication session if the system state and the chosen action are (c, s) and a, respectively at decision point t

Infinite horizon model

m	The expected duration of a communication session
ψ	The set of system states
η	The system state
τ	The state representing the end of a session
$\kappa(\eta)$	The reserved cost if the state is η
$\varsigma(\eta)$	The cost saving if the state is η
α_η	The set of available actions when the system state is η
$\Omega(\eta, a)$	The cost of the routing cycle if the system state is η and the chosen action is a
$\Gamma(\eta'\|\eta, a)$	The state transition probability from the current state η to the next state η' when the chosen action is a
π	The stationary policy in the infinite horizon model. $\pi(\eta)$

	denotes the action performed when the system state is η
π_{DCR}	The stationary policy of routing scheme DCR
$H_t^{\pi}(\eta'\|\eta)$	The probability that if the current system state is η and the stationary policy is π, then the system state becomes η' after t decision points
λ	The discount factor
$\gamma_{\lambda}^{\pi}(\eta)$	The average discounted cost when the stationary policy is π, the current state is η and the discount factor is λ
$\mathbf{\Gamma}_{\pi}$	The matrix containing the state transition probabilities with stationary policy π
$\mathbf{\Omega}_{\pi}$	The matrix containing the costs for each routing cycle of different states if the stationary policy is π
$\boldsymbol{\gamma}_{\lambda}^{\pi}$	The matrix containing the average discounted cost of different initial states if the stationary policy is π and the discount factor is λ

A discrete time system is used and the network cost is updated at time 1, 2, 3, and so on. We denote the time interval between t and $t + 1$ as the t^{th} routing cycle. In the finite horizon model, a communication session lasts for a duration of d time units (i.e., the session starts at time 1 and ends at time $d + 1$). For the infinite horizon model, a communication session may end with a probability $1/m$ in each routing cycle, thus giving an expected duration of m time units. At the beginning of a session, the sender's packlet chooses the minimum cost path and then reserves it for the subsequent routing cycles. Once the path is reserved, the cost is fixed until changes occur in the path. This means that the sender can at least use the existing path at the current reserved cost for the

subsequent cycles. Later, a lower cost path may become available. A sender's packlet may switch to the better path by paying a one-time switching cost, C_{switch}. After switching, the newly reserved cost applies to the current and subsequent cycles until switching occurs again. The cost for each cycle is determined by adding the current reserved cost and any switching cost. The total cost for the whole session is the sum of the costs for all the routing cycles. The objective of our active routing policy is to minimize the total cost for a session.

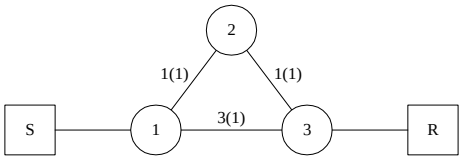


Fig. 3.5 A network with two paths where the link weights are the link costs in the 1st cycle and the 5th cycle (inside the brackets)

Fig. 3.5 shows a simple example to illustrate the problem. Suppose that a sender S and a receiver R are connected by two paths: 1→2→3 and 1→3. At the 1st cycle (session set-up), the path 1→2→3 is reserved because it has a smaller cost of 1 + 1 = 2. At the 5th cycle, the cost of the path 1→3 becomes 1. Since a lower cost path is now available, the session needs to decide whether to keep the path 1→2→3 or switch to the path 1→3 by paying a switching cost, say 10. If the session will terminate after two further cycles, and the subsequent network costs remain unchanged, then the cost of the remaining cycles will be 10 + 1 + 1 = 12 if switching is performed, or 2 + 2 = 4 otherwise. In this case, it is preferable not to switch, even though a lower-cost path is

available. Note that, the actual problem is more complicated because the network cost is probabilistic.

Following the notation in [38], we formulate the Markov decision models as follows. In both the finite and infinite horizon models, we assume that the network cost c_j is governed by a probability distribution $N(c_j)$, where $c_j > 0$, j = 1, 2, …, n, and $c_j < c_{j+1}$ for all j. Decisions are made at the beginning of each routing cycle. The set of decision points is denoted by T = {1, 2, …, d} and T = {1, 2, …} for the finite and infinite horizon models, respectively. In both models, we define (c, s) as the system state, where c is the current reserved cost and s is the available cost saving. If $s > 0$, the current network cost is $c - s$, whereas if s = 0 it means that the current network cost is larger than or equal to the current reserved cost. The specific details for each model are given below.

3.3.1. Finite Horizon Model

We denote S_t as the set of system states at stage t in the finite horizon model as follows:

$$S_t = \left\{c_i, \left(c_i - c_j\right)^+\right\} \quad \text{.....(3.1)}$$

where i = 1, 2, …, n and j = 1, 2, …, i, for t = 1, 2, …, d + 1. In other words, S_t = {(c_1, 0), (c_2, 0), (c_2, $c_2 - c_1$), (c_3, 0), …, (c_n, $c_n - c_2$), (c_n, $c_n - c_1$)}. There are $n(n + 1)/2$ possible states at each decision point in the finite horizon model. Note that, S_{d+1} is the set of terminal states (i.e., when the session is terminated).

Let $A_{(c,s),t}$ denotes a set of actions that are to be chosen at decision point t with state (c, s) $\in$ S_t. It is defined as follows:

$$A_{(c,s),t} = \begin{cases} \{1\,(\text{Switch}), 0\,(\text{Do not switch})\}, & \text{if } s > 0 \\ \{0\}, & \text{if } s = 0 \end{cases} \quad \text{.....(3.2)}$$

where t = 1, 2, …, d. Obviously, it is not cost-effective to switch if s = 0 (i.e., there is no cost saving).

Let $K_t(c, s, a)$ denote the cost for the t^{th} routing cycle if the system state is $(c, s) \in S_t$ and the chosen action is $a \in A_{(c,s),t}$ at t. It is defined by the following function:

$$K_t(c, s, a) = \begin{cases} c, & \text{if } a = 0 \\ c - s + C_{switch}, & \text{if } a = 1 \end{cases} \quad \text{.....(3.3)}$$

where t = 1, 2, …, d. It means that if the chosen action is to "Switch", the cost of the current cycle is the new reserved cost plus the switching cost. If the choice is "Do not switch", the cost is equal to the existing reserved cost.

Let $T((k, l)|(i, j), a)$ denote the state transition probability from the current state, $(i, j) \in S_t$ at t, to the next state, $(k, l) \in S_{t+1}$ at $t + 1$ when the chosen action is $a \in A_{(i,j),t}$. It is given by expression (4) for t = 1, 2, …, d, as follows:

$$T((k, l)|(i, j), a) = \begin{cases} \sum_{c \geq k, c \in \{c_1, c_2, \ldots c_n\}} N(c), & \text{if } k = i, l = 0 \text{ and } a = 0 \text{ or} \\ & \text{if } k = i - j, l = 0 \text{ and } a = 1 \\ N(k - l), & \text{if } k = i, l > 0 \text{ and } a = 0 \text{ or} \\ & \text{if } k = i - j, l > 0 \text{ and } a = 1 \\ 0, & \text{otherwise} \end{cases} \quad \text{.....(3.4)}$$

If the action is to "Switch", the reserved cost for the next state (i.e., k) becomes $i - j$; otherwise, it remains unchanged. The first term in expression (3.4) represents the transitions with no cost savings (i.e., l = 0). The corresponding probability is the sum of $N(c)$ for all $c \geq k$, $c \in \{c_1, c_2, ..., c_n\}$. The second term represents the cases with cost savings.

Let $v_t(c, s)$ denote the minimum expected accumulated cost for routing cycles

$t, t + 1, \ldots, d + 1$, when the system state is $(c, s) \in S_t$ at t. According to [38], $v_t(c, s)$ and $v_{t+1}(i, j)$ are related by a recursive equation for $t = 1, 2, \ldots, d$, as follows:

$$v_t(c, s) = \min_{a \in A_{(c,s),t}} \left\{ K_t(c, s, a) + \sum_{(i, j) \in S_{t+1}} T((i, j) \mid (c, s), a) v_{t+1}(i, j) \right\} \quad \text{.....(3.5)}$$

To solve the recursive equation, the expected cost for the terminal state $(c, s) \in S_{d+1}$ is initialized as

$$v_{d+1}(c, s) = 0 \quad \text{.....(3.6)}$$

Starting from $t = d + 1$, we can calculate recursively $v_t(c, s)$ for each state $(c, s) \in S_t$ at $t = 1, 2, \ldots, d$.

At each decision point t, the sender's packlet needs to choose an action $a \in A_{(c,s),t}$ to make $v_t(c, s)$ as small as possible. Let $A^*_{(c,s),t}$ be the preferred action for state (c, s) at t. This means that it is the action with the smallest $v_t(c, s)$. By using the backward induction algorithm, we can obtain the minimal expected cost for the whole session (i.e., $v_1(c, s)$) for each initial state $(c, s) \in S_1$. Furthermore, the set of preferred actions for different states at each decision point (i.e., the routing policy) can be found. Appendix I gives a simple example to illustrate the backward induction algorithm.

3.3.2. Infinite Horizon Model

For the infinite horizon model, we denote ψ the set of states at each decision point as follows:

$$\psi = \left\{ c_i, (c_i - c_j)^+ \right\} \cup \{\tau\} \quad \text{.....(3.7)}$$

where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, i$. The states are similar to those of the

finite horizon model except that a state τ is included to represent the end of a session. The total number of states is $n(n + 1)/2 + 1$. If the current state is $\eta \in \psi$, where $\eta \neq \tau$, we denote $\kappa(\eta)$ and $\varsigma(\eta)$ as the corresponding reserved cost and available cost saving, respectively, i.e., if $\eta = (i, j)$, we have $\kappa(\eta) = i$ and $\varsigma(\eta) = j$.

Denote α_η as a set of actions that can be selected when the state is $\eta \in \psi$. It is defined as follows:

$$\alpha_\eta = \begin{cases} \{1\,(\text{Switch}), 0\,(\text{Do not switch})\}, & \text{if } \eta \neq \tau \text{ and } \varsigma(\eta) > 0 \\ \{0\}, & \text{if } \eta \neq \tau \text{ and } \varsigma(\eta) = 0 \\ \{\text{-}1\,(\text{No available action})\}, & \text{if } \eta = \tau \end{cases} \quad \text{.....(3.8)}$$

The available actions at each state are similar to those in the finite horizon model except that no action is needed if $\eta = \tau$ (i.e., the session has ended).

Let $\Omega(\eta, a)$ be the cost for each routing cycle if the system state is $\eta \in \psi$ and the chosen action is $a \in \alpha_\eta$. Similar to (3), the function is defined as follows:

$$\Omega(\eta, a) = \begin{cases} \kappa(\eta), & \text{if } \eta \neq \tau \text{ and } a = 0 \\ \kappa(\eta) - \varsigma(\eta) + C_{switch}, & \text{if } \eta \neq \tau \text{ and } a = 1 \\ 0, & \text{if } \eta = \tau \end{cases} \quad \text{.....(3.9)}$$

Obviously, if the session has ended, the cost should be zero (i.e., the third case).

We denote $\Gamma(\eta'|\eta, a)$ as the state transition probability from the current state $\eta \in \psi$ to the next state $\eta' \in \psi$ when the chosen action is $a \in \alpha_\eta$. Recall that m is the expected duration of a session. $\Gamma(\eta'|\eta, a)$ is given by the following expression:

$$\Gamma(\eta' \mid \eta, a) = \begin{cases} (1-1/m) \sum_{c \geq \kappa(\eta'), c \in \{c_1, c_2, \ldots, c_n\}} N(c), & \text{if } \eta' \neq \tau, \eta \neq \tau, \kappa(\eta') = \kappa(\eta), \varsigma(\eta') = 0 \text{ and } a = 0 \text{ or} \\ & \text{if } \eta' \neq \tau, \eta \neq \tau, \kappa(\eta') = \kappa(\eta) - \varsigma(\eta), \varsigma(\eta') = 0 \text{ and } a = 1 \\ (1-1/m) N(\kappa(\eta') - \varsigma(\eta')), & \text{if } \eta' \neq \tau, \eta \neq \tau, \kappa(\eta') = \kappa(\eta), \varsigma(\eta') > 0 \text{ and } a = 0 \text{ or} \\ & \text{if } \eta' \neq \tau, \eta \neq \tau, \kappa(\eta') = \kappa(\eta) - \varsigma(\eta), \varsigma(\eta') > 0 \text{ and } a = 1 \\ 1/m, & \text{if } \eta' = \tau \text{ and } \eta \neq \tau \\ 1, & \text{if } \eta' = \tau \text{ and } \eta = \tau \\ 0, & \text{otherwise} \end{cases} \quad \text{.....(3.10)}$$

The first two terms in (3.10) are similar to those in (3.4) except that both terms are multiplied by $1 - 1/m$. This is because an on-going session will end and continue with a probability of $1/m$ and $1 - 1/m$, respectively. The third term represents the situation where the session will end at the next stage. The fourth term denotes the termination of a session.

For the infinite horizon model, the actions are determined by a stationary policy (i.e., the actions depend on the system state only). Mathematically, if the system state is $\eta \in \psi$ and a stationary policy called π is used, the corresponding action is given by the function $\pi(\eta)$. For instance, if $n = 2$ such that the number of possible states are $(c_1, 0)$, $(c_2, 0)$, $(c_2, c_2 - c_1)$ and τ, a possible stationary policy is $\pi((c_1, 0)) = 0$, $\pi((c_2, 0)) = 0$, $\pi((c_2, c_2 - c_1)) = 1$ and $\pi(\tau) = -1$. In this example, switching to the new path is performed only if the system state is $(c_2, c_2 - c_1)$.

Suppose that the stationary policy is π and at the current decision point, the system state is η. The probability that the system state becomes η' after t decision points is denoted as $H_t^{\pi}(\eta' \mid \eta)$, which can be found by the recursive equation: $H_t^{\pi}(\eta' \mid \eta) = \sum_{i \in \psi} \Gamma(\eta' \mid i, \pi(i)) H_{t-1}^{\pi}(i \mid \eta)$. Starting from a current state η and using a policy π, the average discounted network cost $\gamma_{\lambda}^{\pi}(\eta)$ can be found as follows:

$$\begin{aligned}\gamma_\lambda^\pi(\eta) &= \Omega(\eta,\pi(\eta)) + \sum_{\eta'\in\psi}\lambda H_1^\pi(\eta'|\eta)\Omega(\eta',\pi(\eta')) + \sum_{\eta'\in\psi}\lambda^2 H_2^\pi(\eta'|\eta)\Omega(\eta',\pi(\eta')) + ... \\ &= \Omega(\eta,\pi(\eta)) + \sum_{\eta'\in\psi}\lambda\Gamma(\eta'|\eta,\pi(\eta))\Omega(\eta',\pi(\eta')) + \sum_{\eta'\in\psi}\lambda^2(\sum_{i\in\psi}\Gamma(\eta'|i,\pi(i))H_1^\pi(i|\eta))\Omega(\eta',\pi(\eta')) + ... \\ &= \Omega(\eta,\pi(\eta)) + \sum_{\eta'\in\psi}\lambda\Gamma(\eta'|\eta,\pi(\eta))\gamma_\lambda^\pi(\eta')\end{aligned} \quad ..(3.11)$$

where λ ($0 \leq \lambda < 1$) is the discount factor. The discount factor is used to reflect the importance of the future costs. Note that if λ is close to one, the future costs are as important as the current ones. Essentially (3.11) is obtained by following the standard approach used in formulating a Markov decision infinite horizon model (see [38] for details). By rearranging the terms in (3.11), we obtain

$$\Omega(\eta,\pi(\eta)) = \sum_{\eta'\in\psi}[\delta(\eta',\eta) - \lambda\Gamma(\eta'|\eta,\pi(\eta)]\gamma_\lambda^\pi(\eta') \quad .....(3.12)$$

where $\delta(\eta', \eta) = 1$ if $\eta' = \eta$ and $\delta(\eta', \eta) = 0$ otherwise. In matrix terms, (12) can be expressed as follows:

$$(\mathbf{I} - \lambda\boldsymbol{\Gamma}_\pi)\boldsymbol{\gamma}_\lambda^\pi = \boldsymbol{\Omega}_\pi \quad .....(3.13)$$

where $\mathbf{I}$ is the identity matrix, $\boldsymbol{\Gamma}_\pi$ =

$$\begin{bmatrix} \Gamma((c_1,0)|(c_1,0),\pi((c_1,0))) & \Gamma((c_2,0)|(c_1,0),\pi((c_1,0))) & \Gamma((c_2,c_2-c_1)|(c_1,0),\pi((c_1,0))) & \dots & \Gamma((c_n,c_n-c_1)|(c_1,0),\pi((c_1,0))) & \Gamma(\tau|(c_1,0),\pi((c_1,0))) \\ \Gamma((c_1,0)|(c_2,0),\pi((c_2,0))) & \Gamma((c_2,0)|(c_2,0),\pi((c_2,0))) & \Gamma((c_2,c_2-c_1)|(c_2,0),\pi((c_2,0))) & \dots & \Gamma((c_n,c_n-c_1)|(c_2,0),\pi((c_2,0))) & \Gamma(\tau|(c_2,0),\pi((c_2,0))) \\ \Gamma((c_1,0)|(c_2,c_2-c_1),\pi((c_2,c_2-c_1))) & \Gamma((c_2,0)|(c_2,c_2-c_1),\pi((c_2,c_2-c_1))) & \Gamma((c_2,c_2-c_1)|(c_2,c_2-c_1),\pi((c_2,c_2-c_1))) & \dots & \Gamma((c_n,c_n-c_1)|(c_2,c_2-c_1),\pi((c_2,c_2-c_1))) & \Gamma(\tau|(c_2,c_2-c_1),\pi((c_2,c_2-c_1))) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \Gamma((c_1,0)|(c_n,c_n-c_1),\pi((c_n,c_n-c_1))) & \Gamma((c_2,0)|(c_n,c_n-c_1),\pi((c_n,c_n-c_1))) & \Gamma((c_2,c_2-c_1)|(c_n,c_n-c_1),\pi((c_n,c_n-c_1))) & \dots & \Gamma((c_n,c_n-c_1)|(c_n,c_n-c_1),\pi((c_n,c_n-c_1))) & \Gamma(\tau|(c_n,c_n-c_1),\pi((c_n,c_n-c_1))) \\ \Gamma((c_1,0)|\tau,\pi(\tau)) & \Gamma((c_2,0)|\tau,\pi(\tau)) & \Gamma((c_2,c_2-c_1)|\tau,\pi(\tau)) & \dots & \Gamma((c_n,c_n-c_1)|\tau,\pi(\tau)) & \Gamma(\tau|\tau,\pi(\tau)) \end{bmatrix},$$

$$\boldsymbol{\Omega}_\pi = \begin{bmatrix} \Omega((c_1,0),\pi((c_1,0))) \\ \Omega((c_2,0),\pi((c_2,0))) \\ \Omega((c_2,c_2-c_1),\pi((c_2,c_2-c_1))) \\ \vdots \\ \Omega((c_n,c_n-c_{n-1}),\pi((c_n,c_n-c_{n-1}))) \\ \Omega(\tau,\pi(\tau)) \end{bmatrix} \text{ and } \boldsymbol{\gamma}_\lambda^\pi = \begin{bmatrix} \gamma_\lambda^\pi((c_1,0)) \\ \gamma_\lambda^\pi((c_2,0)) \\ \gamma_\lambda^\pi((c_2,c_2-c_1)) \\ \vdots \\ \gamma_\lambda^\pi((c_n,c_n-c_{n-1})) \\ \gamma_\lambda^\pi(\tau) \end{bmatrix}.$$

By solving (3.13), we can determine the average discounted cost for each system state and hence obtain the optimal policy (i.e., the best action for each state). The policy iteration method (see [38]) is adopted to determine the optimal policy. To demonstrate the policy iteration method, a simple example is presented.

We assume ψ = {(1, 0), (2, 0), (2, 1), τ}, $N(1) = 1/2$, $N(2) = 1/2$, $m = 3$, $\lambda = 0.9$

and C_{switch} = 1/2. The policy iteration algorithm works as follows:

a. Initially, an arbitrary policy is chosen. For simplicity, we choose the policy as $\pi((1, 0)) = 0$, $\pi((2, 0)) = 0$, $\pi((2, 1)) = 0$ and $\pi(\tau) = -1$.

b. The equation $(\mathbf{I} - \lambda\mathbf{\Gamma}_\pi)\boldsymbol{\gamma}_\lambda^\pi = \mathbf{\Omega}_\pi$ is solved. In our example, the equation is $\left[\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} - 0.9\begin{bmatrix}2/3&0&0&1/3\\0&1/3&1/3&1/3\\0&1/3&1/3&1/3\\0&0&0&1\end{bmatrix}\right]\gamma_\lambda^\pi = \begin{bmatrix}1\\2\\2\\0\end{bmatrix}$ and thus $\gamma_\lambda^\pi = \begin{bmatrix}2.5\\5\\5\\0\end{bmatrix}$.

c. For each $\eta \in \psi$ and $a \in \alpha_\eta$, we compute $\Omega(\eta, a) + \sum_{\eta' \in \psi} \lambda\Gamma(\eta' | \eta, a)\gamma_\lambda^\pi(\eta')$. For example, for η = (2, 1), the feasible actions are 0 (not to switch) and 1 (switch). So for action 0,

$\Omega((2, 1), 0) + 0.9(\Gamma((1, 0)|(2, 1), 0)\,\gamma_\lambda^\pi((1,0)) + \Gamma((2, 0)|(2, 1), 0)\,\gamma_\lambda^\pi((2,0))$ $+ \Gamma((2, 1)|(2, 1), 0)\,\gamma_\lambda^\pi((2,1)) + \Gamma(\tau\,|(2, 1), 0)\,\gamma_\lambda^\pi(\tau)) = 2 + 0.9(0(2.5) +$ $1/3(5) + 1/3(5) + 1/3(0)) = 5$

and for action 1,

$\Omega((2, 1), 1) + 0.9(\Gamma((1, 0)|(2, 1), 1)\,\gamma_\lambda^\pi((1,0)) + \Gamma((2, 0)|(2, 1), 1)\,\gamma_\lambda^\pi((2,0))$ $+ \Gamma((2, 1)|(2, 1), 1)\,\gamma_\lambda^\pi((2,1)) + \Gamma(\tau\,|(2, 1), 1)\,\gamma_\lambda^\pi(\tau)) = 1.5 + 0.9(2/3(2.5)$ $+ 0(5) + 0(5) + 1/3(0)) = 3$

For each $\eta \in \psi$, we put $a \in \alpha_\eta$ into the set α_η* if a gives a lower cost. In our example, we have α_η* = {1}. Note that in the case of a tie, we put a = 0 into α_η* since it is generally preferrable to stay in the same path.

d. If for each $\eta \in \psi$, the coressponding action a in the set α_η* is equal to $\pi(\eta)$, the iterative process is stopped. The optimal policy π* is given by π. If not, we update the policy rule $\pi(\eta)$ based on the actions in α_η* and then return to step b. In our example, we update $\pi((2, 1)) = 1$ and then repeat step b.

Table 3.3 summarizes the results of the above policy iteration process for this example. As shown in the table, it is better to switch to a new path only if the

current reserved cost is 2 and the current network cost is 1.

Table 3.3 Summary of results for the policy iteration example

η	(1, 0)		(2, 0)		(2, 1)		τ	
Iteration	1	2	1	2	1	2	1	2
$\pi(\eta)$	0	0	0	0	0	1	-1	-1
$\gamma_\lambda^\pi(\eta)$	2.5	2.5	5	4.1429	5	3	0	0
$\Omega(\eta,0)+\sum_{\eta'\in\psi}\lambda\Gamma(\eta'\mid\eta,0)\gamma_\lambda^\pi(\eta')$	2.5	2.5	5	4.1429	5	4.1429	N/A	N/A
$\Omega(\eta,1)+\sum_{\eta'\in\psi}\lambda\Gamma(\eta'\mid\eta,1)\gamma_\lambda^\pi(\eta')$	N/A	N/A	N/A	N/A	3	3	N/A	N/A
α_η^*	0	0	0	0	1	1	-1	-1

3.4. Analysis and discussion

In this section, we analyze MARS in comparison with several other schemes. To carry out the later analysis, we assume that the network cost c_j (j=1,2,3,…,n) is normally distributed with mean μ and standard deviation σ. More specifically, we assume that

$$c_j = \mu - \frac{(n-1)\delta\sigma}{2} + (j-1)\delta\sigma \text{ , } j = 1, 2, \ldots, n. \quad \text{.....(3.14)}$$

$N(c_j)$ is given by the area under the respective normal curve, which is bounded by $c_j \pm\Delta$ where Δ equals a half interval (i.e., $\delta\sigma/2$). For the leftmost (c_1) and rightmost (c_n) cost levels, the areas are extended to $-\infty$ and ∞, respectively. It is assumed that $N(c_j)$ can be available from the NOM based on the past statistics. Note that the normal distribution is just an example for the later analysis. Other distributions can also be used in the Markov decision models.

3.4.1. Different Schemes

In the analysis, we assume that $n = 51$ and $\delta = 0.2$. Our aim is to evaluate the average cost C for a session under the following schemes:

Fixed (FIX): The network cost is reserved, at the time the session is set-up, and is then fixed for the whole session. This resembles the approach used by the connection-oriented services (e.g., ATM).

Dynamic without cost reservation (DYN): The cost of each cycle is changed according to the updated network cost. This resembles the approach used by the connectionless services as communication paths are changed dynamically.

Dynamic with cost reservation (DCR): The network cost is reserved in this case. Furthermore, if the current network cost is smaller than the reserved network cost, the latter will be updated to the former.

Markov-decision-based Active Routing Scheme (MARS): The decision policy, as described in Section III, is used to support active routing.

Note that in active routing, a user can employ any routing scheme including the traditional approaches (i.e., similar to FIX, DYN and DCR) and other new approaches (e.g., MARS and even customized or secret algorithms). Therefore it is highly flexible and dynamic. As shown later, MARS provides the best performance under a wide range of network conditions.

3.4.2. Equations for the Finite Horizon Model

In FIX, the total average cost can be easily found to be

$$C = d\mu \quad \text{.....(3.15)}$$

In DYN, the switching cost C_{switch} is added to the network cost in each routing cycle. Hence, the total average cost is

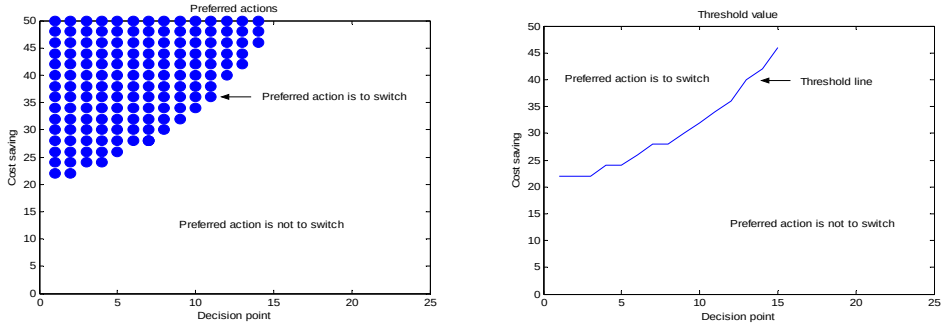
$$C = d\mu + dC_{switch} \quad \text{.....(3.16)}$$

In MARS, by using the discrete costs c_j and the associated probabilities $N(c_j)$, we can formulate the states and compute the transition probability matrix. In the finite horizon model, we can determine the preferred actions by using the backward induction algorithm [38] for a given d. Finally, the total cost is given by

$$C = \sum_{j=1}^{n} v_1(c_j, 0) N(c_j) \quad \text{.....(3.17)}$$

Referring to our earlier numerical example, the total average cost is 2(1/3) + 4(1/3) + 6(1/3) = 4. Besides the total cost, it is interesting to study how different parameters could affect the decision policy. Fig. 7(a) shows the preferred actions $A^*_{(c,s),t}$ for states (c, s) = (250, 0), (250, 2), …, (250, 50) at t = 1, 2, …, 25, when μ = 250, σ = 10, d = 25 and C_{switch} = 500. Each of the points in Fig. 3.6(a) shows that the preferred action is to “Switch”. It can be seen that the policy has the following form: at a particular t, switching is preferred if the cost saving reaches a certain threshold. For example, at t = 14, it is preferred to switch if the cost saving is greater than or equal to 42. Otherwise, no switching is preferred. Fig. 3.6(b) shows the corresponding threshold values for different decision points when the reserved cost is equal to

μ.



(a) **(b)**

Fig. 3.6 An example showing (a) the preferred actions and (b) the corresponding threshold values for different values of cost saving at different decision points in the finite horizon model (μ = 250, σ = 10, d = 25 and C_{switch} = 500)

For DCR, we use a slightly modified backward induction method to calculate the total average cost in the finite horizon model. In this case, when the cost saving is greater than 0, switching will be performed. Mathematically, $v_t(c, s)$ is expressed as follows:

$$v_t(c,s) = \begin{cases} K_t(c,s,1) + \sum_{(i,j)\in S_{t+1}} T((i,j)|(c,s),1) v_{t+1}(i,j), & \text{if } s > 0 \\ K_t(c,s,0) + \sum_{(i,j)\in S_{t+1}} T((i,j)|(c,s),0) v_{t+1}(i,j), & \text{if } s = 0 \end{cases} \text{, for } t=1, 2, \ldots, d \text{ .} \quad (3.18)$$

Based on (3.6) and (3.18), the total average cost can be obtained by using (3.17).

3.4.3. Equations for the Infinite Horizon Model

For FIX, the average cost is given by

$$C = \mu(1 - \lambda^m) / (1 - \lambda) \quad \text{.....(3.19)}$$

On average, a session has m stages. Taking into account the discount factor λ, the average cost is $\mu + \lambda\mu + \lambda^2\mu + \ldots + \lambda^{m-1}\mu = \mu(1 - \lambda^m) / (1 - \lambda)$, which is given in (3.19).

Similarly the average cost for DYN is:

$$C = \mu(1 - \lambda^m) / (1 - \lambda) + C_{switch}(1 - \lambda^m) / (1 - \lambda) \text{.....(3.20)}$$

In this case, we need to take into account the switching cost at each routing cycle.

For MARS, we can determine the optimal policy by solving (3.13) based on the policy iteration method. Let the optimal policy be π^*. The average cost can then be found as follows:

$$C = \sum_{j=(c,0),\ c=\{c_1,c_2,...,c_n\}} \gamma_\lambda^{\pi^*}(j) N(c) \text{.....(3.21)}$$

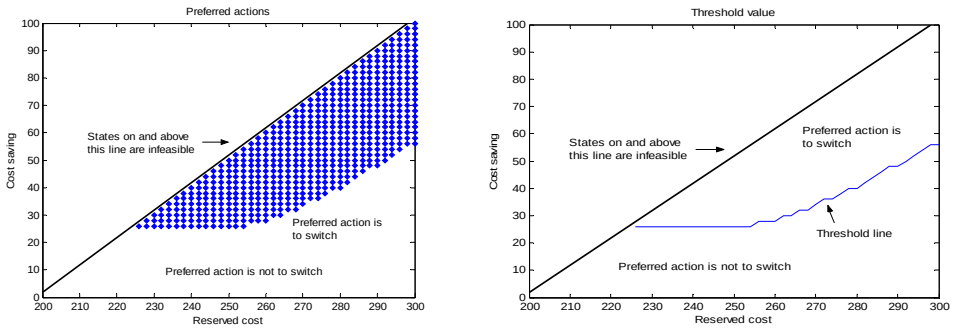
For DCR, denote $\pi_{DCR}(\eta)$ as the corresponding policy, i.e.,

$$\pi_{DCR}(\eta) = \begin{cases} 1, & \text{if } \eta \neq \tau \text{ and } \varsigma(\eta) > 0 \\ 0, & \text{if } \eta \neq \tau \text{ and } \varsigma(\eta) = 0 \\ -1, & \text{if } \eta = \tau \end{cases} \text{.....(3.22)}$$

This policy simply means that switching to the new path is performed whenever there is a cost saving (i.e., it ignores the switching cost and future effects). To calculate the average cost, we first determine the expected discounted cost $\gamma_\lambda^{\pi_{DCR}}(\eta)$ for each $\eta \in \psi$ where $\eta = (c, 0)$, $c \in \{c_1, c_2, ..., c_n\}$. By finding the transition probability matrix $\mathbf{\Gamma}_{\pi_{DCR}}$ and the cost vector $\mathbf{\Omega}_{\pi_{DCR}}$, $\gamma_\lambda^{\pi_{DCR}}(\eta)$ can be easily found based on (3.13). The average cost can then be obtained by (3.21).

It is also of interest to study how different parameters can affect the decision policy in the infinite horizon model. Fig. 3.7(a) shows the preferred actions α_η^* for all possible states, when $\lambda = 0.99$, $\mu = 250$, $\sigma = 10$, $m = 25$ and C_{switch} =

500. Each of the points in Fig. 3.7(a) shows that the preferred action is to "Switch" and all the points on and above the bolded line indicate the impossible states. It can be seen that the policy has the following form: at a particular value of $\kappa(\eta)$, switching is preferred if the cost saving reaches a certain threshold. For example, at $\kappa(\eta)$ = 250, it is preferred to switch if the cost saving is greater than or equal to 26. Otherwise, no switching is preferred. Fig. 3.7(b) shows the corresponding threshold values for different values of reserved cost. Note that when $\kappa(\eta)$ is smaller than or equal to 224, switching should not be performed no matter what the available cost saving is.



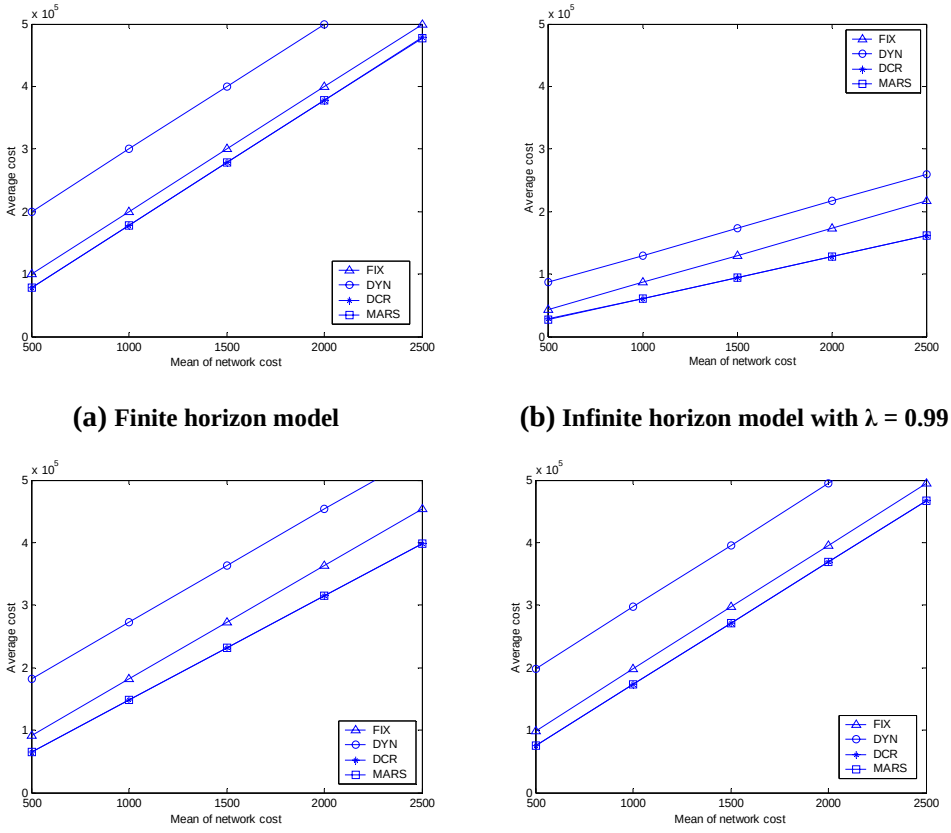
(a) **(b)**

Fig. 3.7 An example showing (a) the preferred actions and (b) the corresponding threshold values for different values of cost saving at different reserved costs in the infinite horizon model (λ = 0.99, μ = 250, σ = 10, m = 25 and C_{switch} = 500)

3.4.4. Analysis and Comparison

To carry out the analysis and comparison, we set our base parameters as follows: μ = 10,000, σ = 2,000, d and m = 200 and C_{switch} = 200,000. These parameters are varied in turn to study their effects on the average cost. For the infinite horizon model, we set λ = 0.99, 0.999 and 0.9999.

Figs. 3.8 – 3.10 show the average cost when μ is varied. Recall that the average cost at each routing cycle is composed of the reserved cost and the switching cost (if any).



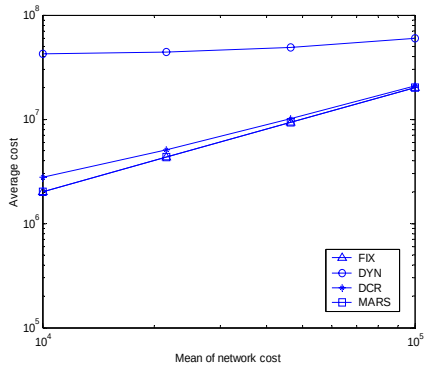
(a) Finite horizon model **(b) Infinite horizon model with λ = 0.99**

(c) Infinite horizon model with λ = 0.999 **(d) Infinite horizon model with λ = 0.9999**

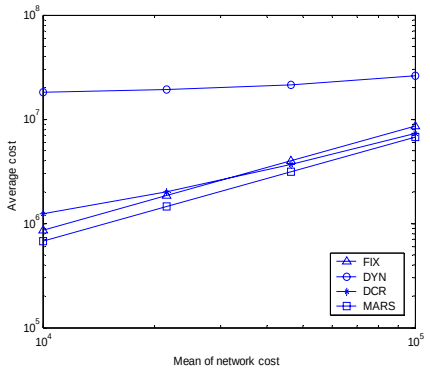
Fig. 3.8 Average cost when the mean of network cost is varied (σ = 50, d = 200 (m = 200) and C_{switch} = 500)

In Fig. 3.8, it can be seen that the average cost for all schemes increases linearly with respect to μ. In fact, in the finite horizon model, (3.15) and (3.16) indicate that the slopes for FIX and DYN are both equal to d, respectively. Similarly, in the infinite horizon model, the slopes for FIX and DYN are equal to $(1 - \lambda^m) / (1 - \lambda)$ based on (3.19) and (3.20), respectively. The figure shows that DCR and MARS give the lowest average cost. Because of more frequent

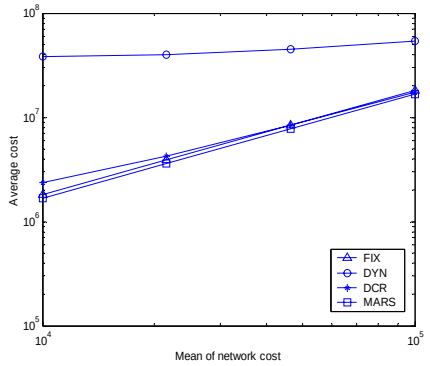
switching, the average cost of DYN is higher. Furthermore, Fig. 3.8(b) indicates that when λ is smaller, the average cost is relatively less sensitive to the change in μ.



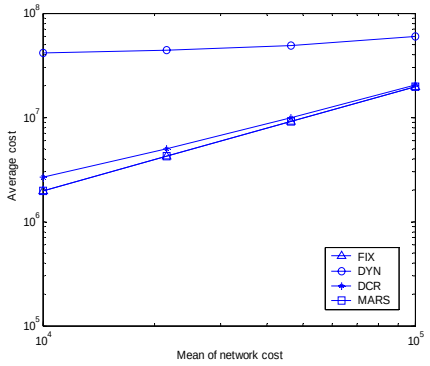
(a) Finite horizon model



(b) Infinite horizon model with λ = 0.99



(c) Infinite horizon model with λ = 0.999

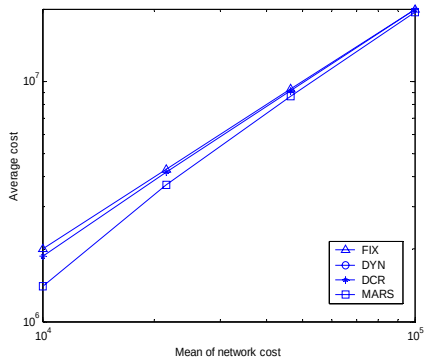


(d) Infinite horizon model with λ = 0.9999

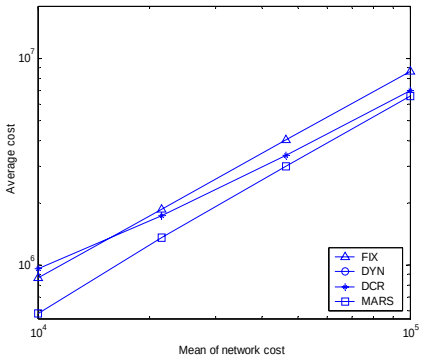
Fig. 3.9 Average cost when the mean of network cost is varied (σ = 50, d = 200 (m = 200) and C_{switch} = 200000)

Fig. 3.9 shows that when the switching cost is increased, DYN is most affected, making it giving the highest average cost. It can also be seen that the average cost of DCR becomes larger than that of MARS. In the new situation, FIX and MARS are more preferable. However, if the standard deviation of the network cost is increased, the difference in average cost between FIX and MARS becomes widen. This is because it is more likely for MARS to reserve a low

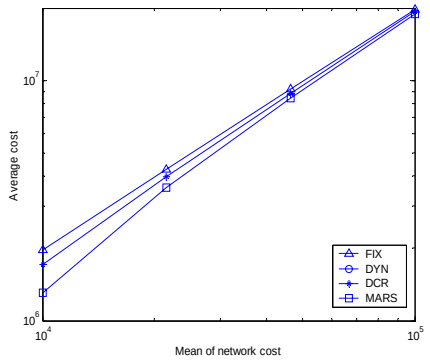
cost path.



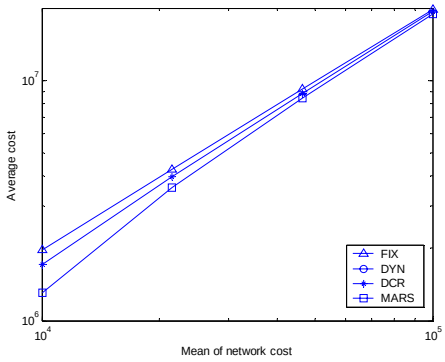
(a) Finite horizon model



(b) Infinite horizon model with λ = 0.99



(c) Infinite horizon model with λ = 0.999

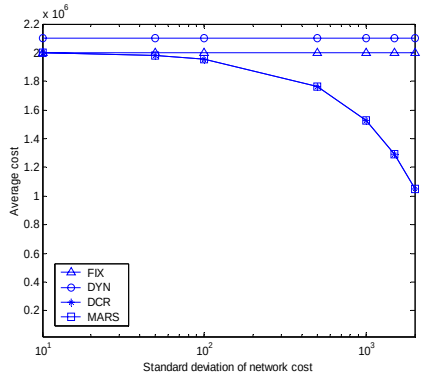


(d) Infinite horizon model with λ = 0.9999

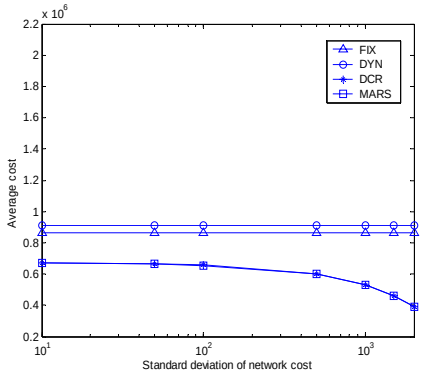
Fig. 3.10 Average cost when the mean of network cost is varied (σ = 2000, d = 200 (m = 200) and C_{switch} = 200000)

By comparing Fig. 3.8 with Figs. 3.9 and 3.10, it can be seen that DCR and MARS are more favorable when σ and C_{switch} are both small in the finite horizon model. As C_{switch} increases, the average cost for DCR will also increase, making it higher than that of FIX and MARS. MARS always gives the best performance, particularly when both σ and C_{switch} are large. The same observation can be found in the infinite horizon model, but only when μ is not large. With a large μ, DCR and MARS are more favorable irrespective of the value of σ and C_{switch}.

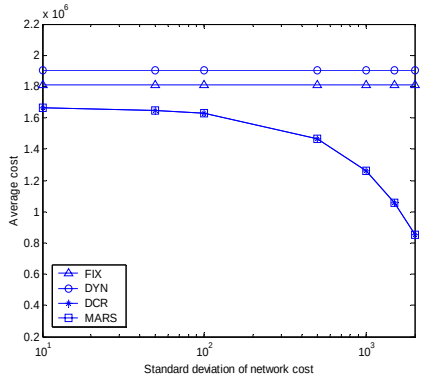
Figs. 3.11 and 3.12 show the effect on the average cost when σ is varied. In both models, the average cost of FIX and DYN is independent of σ.



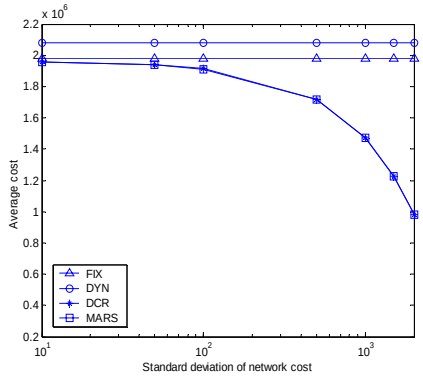
(a) Finite horizon model



(b) Infinite horizon model with λ = 0.99



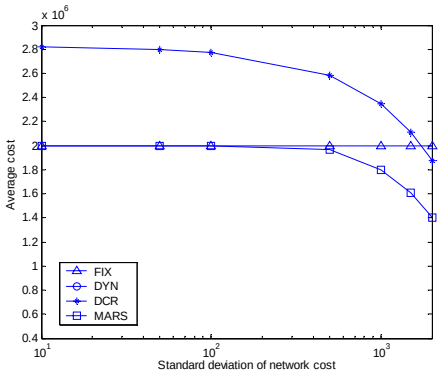
(c) Infinite horizon model with λ = 0.999



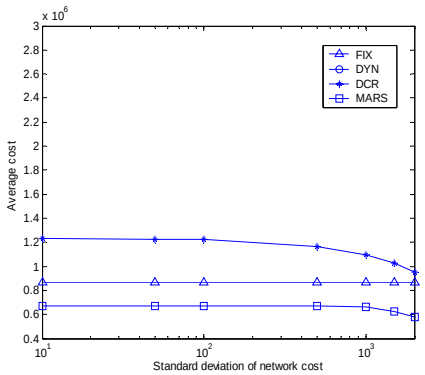
(d) Infinite horizon model with λ = 0.9999

Fig. 3.11 Average cost when the standard deviation of network cost is varied (μ = 10000, d = 200 (m = 200) and C_{switch} = 500)

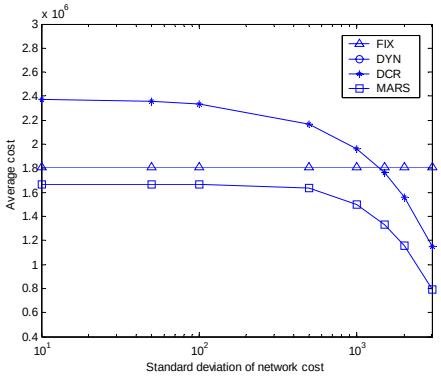
Fig. 3.11(a) shows that the average costs of FIX, DCR and MARS are comparable when σ and C_{switch} are both small in the finite horizon model. As σ increases beyond a certain threshold value, the average cost of DCR and MARS decreases dramatically. When C_{switch} is small, DCR and MARS have almost the same average cost. In Fig. 3.11(b), we can see that in the infinite horizon model, the average cost of FIX is greater than that of DCR and MARS even when σ is small. Furthermore the difference increases as σ is larger.



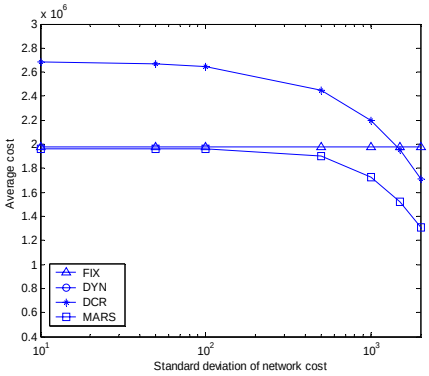
(a) Finite horizon model



(b) Infinite horizon model with λ = 0.99



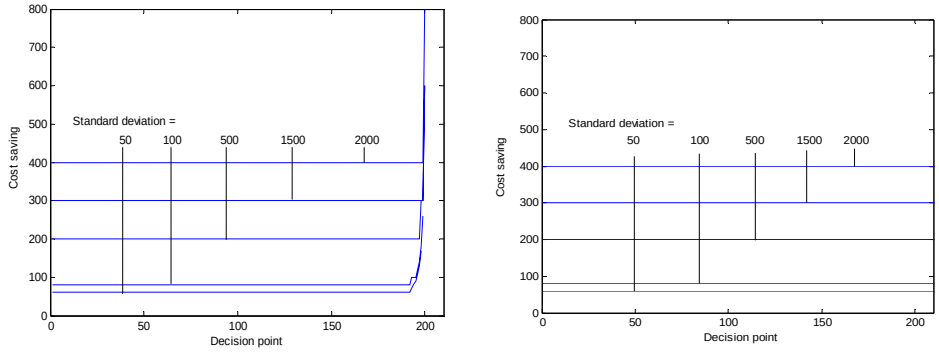
(c) Infinite horizon model with λ = 0.999



(d) Infinite horizon model with λ = 0.9999

Fig. 3.12 Average cost when the standard deviation of network cost is varied (μ = 10000, d = 200 (m = 200) and C_{switch} = 200000)

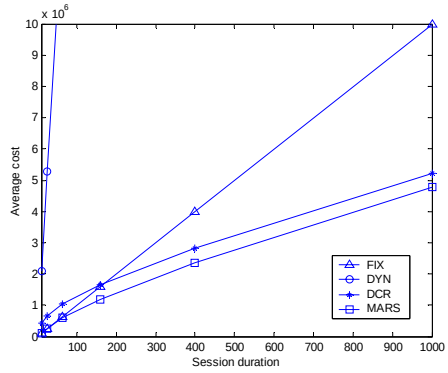
As C_{switch} increases, Fig. 3.12 shows that the average cost rises significantly in DYN and DCR. However, the average cost remains at a low value in MARS even when C_{switch} is large because of its adaptive function. Furthermore, DCR has a lower average cost than that of FIX when σ and C_{switch} are both large.



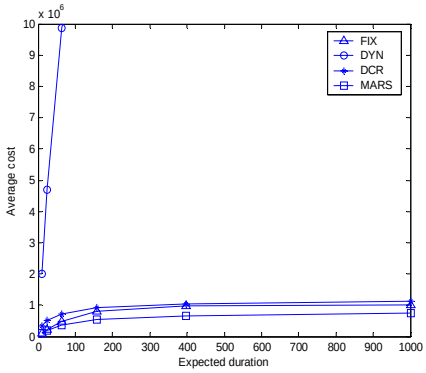
(a) Finite horizon model **(b) Infinite horizon model**

Fig. 3.13 Threshold values for different values of the standard deviation (μ = 10000, d = 200 (m = 200) and C_{switch} = 500)

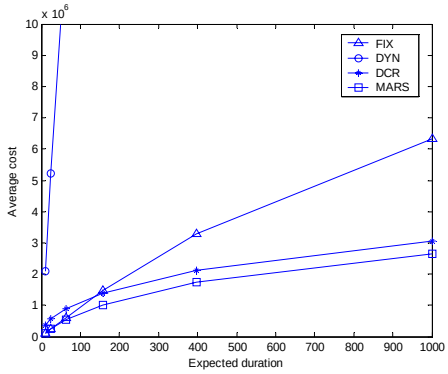
As mentioned before, the decision policy of MARS in the finite horizon model is threshold-based (i.e., it is better to switch to the new path if a certain cost saving is reached). Fig. 3.13(a) shows the threshold values for different values of σ when the reserved cost is μ in the finite horizon model. It can be seen that the nearer the session comes to an end, the larger are the values of the threshold. Towards the end of a session (about 10 routing cycles from the end), the threshold values for different values of σ are very close. Since switching is expensive, the active routing approach prefers not to switch unless the cost saving is very large. In comparison, the threshold values are dramatically lower and maintain stable values when the session is further from the end. The stable threshold values become larger as σ increases. Fig. 3.13(b) shows the threshold values for different values of σ when the reserved cost is μ in the infinite horizon model. Since the threshold values are not dependent on the decision points, the threshold values remain constant during the session. In fact, the threshold values are the same as the steady state threshold values in the finite horizon model. In other words, the discount factor λ does not have an effect on the threshold values.



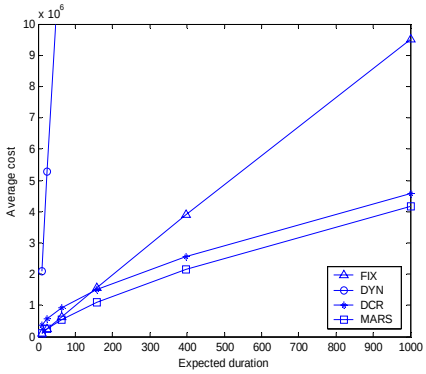
(a) Finite horizon model



(b) Infinite horizon model with λ = 0.99



(c) Infinite horizon model with λ = 0.999

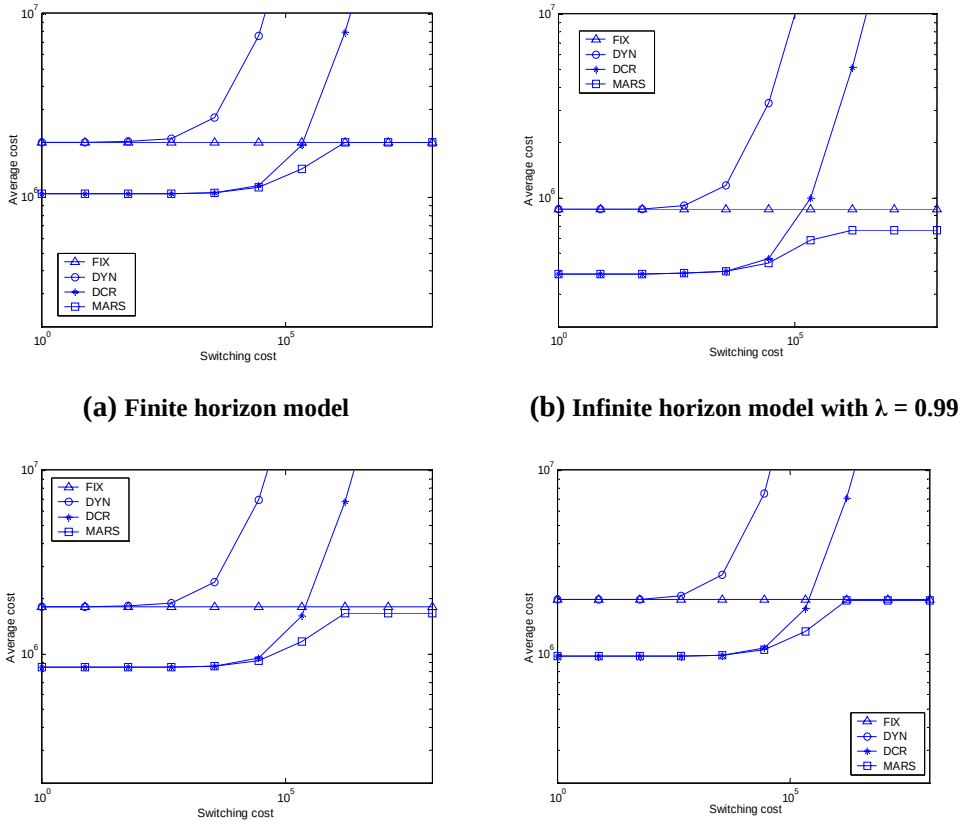


(d) Infinite horizon model with λ = 0.9999

Fig. 3.14 Average cost when the session duration/expected session duration is varied (μ = 10000, σ = 2000 and C_{switch} = 200000)

Fig. 3.14(a) and Fig. 3.14(b) show the average cost when the duration (d) and expected duration (m) of a session are varied in the finite and infinite horizon models, respectively. For the finite horizon model, when d is small, the average cost is close for all schemes. As d increases, the average cost increases at a different rate for each scheme. The average cost for DYN is the most sensitive to an increase in d, whereas DCR and MARS are the least sensitive. This is because, in DCR and MARS, the longer the duration of a session, the higher the chance that a lower cost path can be reserved. For the infinite horizon model, the average cost also increases at a different rate for each scheme when m is small. As m increases, the average cost of each scheme

approaches a constant value. For FIX and DYN, it can be seen from (3.19) and (3.20) that the average costs become $\mu / (1 - \lambda)$ and $\mu / (1 - \lambda) + C_{switch} / (1 - \lambda)$, respectively when m is very large. For DCR and MARS, the average cost is relatively insensitive to the change in m especially when the discount factor is small.



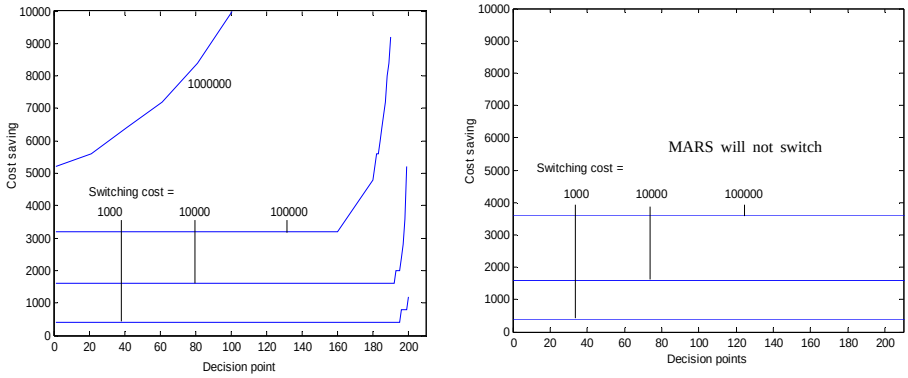
(a) Finite horizon model **(b) Infinite horizon model with λ = 0.99**

(c) Infinite horizon model with λ = 0.999 **(d) Infinite horizon model with λ = 0.9999**

Fig. 3.15 Average cost when the switching cost is varied (μ = 10000, σ = 2000 and d = 200 (m = 200))

Fig. 3.15 shows the average cost when the switching cost C_{switch} is varied. The results are similar in both models. When C_{switch} is small, the average costs of FIX and DYN are very close. Similarly, DCR and MARS give almost the same average cost. Since no switching is required, the average cost of FIX is

independent of C_{switch}. The average cost of DYN is the most sensitive to a change in C_{switch} because of frequent switching. In DCR, switching occurs when a lower cost path is available. Therefore, the average cost of DCR increases less dramatically than it does in DYN, as C_{switch} increases. For MARS, the average cost is even less sensitive to an increase in C_{switch}. When C_{switch} is small, switching occurs as frequently as it does in DCR. When C_{switch} is large, MARS prefers not to switch. Therefore, it performs no worse than FIX even when C_{switch} is very large. This contrasts with DYN and DCR in which the average cost rises far above that of FIX. From the above results, we can see clearly that MARS adapts well to different situations and always gives the lowest average cost.



(a) Finite horizon model **(b) Infinite horizon model**

Fig. 3.16 Threshold values for different values of the switching cost (μ = 10000, σ = 2000 and d = 200 (m = 200))

Referring to the finite horizon model, Fig. 3.16(a) shows the threshold values in MARS for different values of C_{switch}. It can be seen that as C_{switch} increases by an order of magnitude, the stable threshold values increase steadily. Towards the end of a session, MARS prefers not to switch, particularly when

C_{switch} is large, unless the cost saving is very large. Fig. 3.16(b) shows the threshold values in MARS for different values of C_{switch} when the reserved cost is μ in the infinite horizon model. When C_{switch} is small, the threshold values in the infinite horizon model are close to the corresponding stable threshold values in the finite horizon model. However when C_{switch} is large, the threshold values in the infinite horizon model are much higher than their counterparts in the finite horizon model. Note that in the infinite horizon model, when C_{switch} = 1,000,000, switching should not be performed no matter what the cost saving is.

Table 3.4 summarizes the effects of different parameters on the performance. A change in σ produces a more significant effect, than a change in C_{switch}, on both the average cost and the threshold values in MARS. Interestingly, when σ is increased tenfold, it is found that the decision policy is similar to that when C_{switch} is increased tenfold, except that the threshold values also increase tenfold. Due to its adaptive function, MARS provides the best performance in all situations. In particular, it can reduce the average cost significantly, as compared with other schemes, when σ, C_{switch} or d is large. For example, using the base parameters, it can decrease the average cost for a session by about 30% and 25% as compared with that in FIX and DCR, respectively. The cost saving as compared with DYN is even more significant due to the high switching cost.

Table 3.4 Summary of the analysis

Increase in	**Horizon**	**Effect on expected total cost C**				**Effect on threshold**	**Preferred scheme (s)**
		FIX	**DYN**	**DCR**	**MARS**		
μ	**Finite**	↑1	↑1	↑1	↑1	=	N/A
	Infinite	↑1	↑1	↑2	↑2	=	DCR/MARS
σ	**Finite**	=	=	↓1	↓1	↑	DCR/MARS
	Infinite	=	=	↓1	↓1	↑	DCR/MARS
d	**Finite**	↑2	↑1	↑3	↑3	=*	DCR/MARS
m	**Infinite**	↑3	↑1	↑3	↑3	=*	FIX/DCR/MARS
C_{switch}	**Finite**	=	↑1	↑2	↑3	↑	FIX/MARS
	Infinite	=	↑1	↑2	↑3	↑	FIX/MARS

Key:

N/A: Not applicable

↑: increase the value ↓: decrease the value =: no effect on the value

The number indicates the degree of effect: 1:most affected 3: least affected

*Threshold remains constant unless *d*/*m* is very small.

3.5. Conclusion

In conclusion, we have presented an active routing service for the next-generation network called ISDN3. This involves using active packets (packlets) to configure customized communication paths based on the network information. Hence, an end-user application can play an active role in forwarding its packets according to its requirements. The active routing

service can also work in conjunction with MPLS to provide effective and dynamic forwarding services. With the aim of minimizing network costs, both a finite horizon and an infinite horizon Markov decision model have been formulated to support active routing. According to the models, the communication paths should be reconfigured if a particular cost saving threshold is reached. The threshold values, which depend on the network parameters, can be found by solving the Markov decision models. Theoretical analysis confirms that by using the proposed routing policy, the average network cost can be minimized under various conditions as compared with other approaches. Under the proposed “horizontal partitioning” framework, active and traditional network services can co-exist to open many new innovative services in the next-generation network.

Chapter 4.
A Markov Decision-based Bandwidth Selling Policy for an Active MPLS Service

4.1. Introduction

Recent years have seen the development of Multiprotocol Label Switching (MPLS) [6] for forwarding Internet traffic efficiently using label switched paths (LSPs). Essentially, label edge routers (LER) and label switch routers (LSR) are employed for attaching MPLS headers/labels and transferring MPLS packets, respectively. Integrating with a bandwidth reservation protocol, MPLS can also be used to facilitate Quality of Service (QoS) routing [12]. Moreover, based on active network technologies [14,28] (especially active routing and agent-based routing [17,24,25]), it would be useful to develop an active MPLS service that can reserve bandwidth according to an adaptive pricing method. In particular, it is envisioned that both active and non-active (passive) network services can be integrated in an advanced network called ISDN3 [19,29].

To implement the proposed active MPLS service in ISDN3, we need to tackle

the following bandwidth selling problem. In essence, the network provider may receive an offer for reserving one or more unit(s) of bandwidth of an LSP at a certain price. To maximize the overall earnings, the network provider must then determine whether to accept that offer.

Adaptive pricing methods and related issues have also been studied in [17,39-42] for different purposes. Motivated by [17,39-42], this chapter presents an adaptive bandwidth selling method. In particular, the bandwidth selling policy is determined based on a Markov decision model. For the purposes of evaluation, we will compare the selling policy with two other schemes.

The remaining sections are as follows. Section 4.2 gives an overview of the active MPLS service for ISDN3. Section 4.3 outlines the Markov decision model for tackling the bandwidth selling problem. Section 4.4 presents and discusses some analytical and simulation results. Section 4.5 offers a Conclusion.

4.2. Active MPLS Service Over ISDN3

ISDN3 serves as a multi-functional network that can handle active and passive traffic using a traffic forwarding device called “Forwarding EnginE” (FEE) [19,29]. Motivated by ideas in [17,24,25], we investigate an active MPLS service for ISDN3 as outlined below. At the start of a session, an active network application can communicate with the associated FEE by using an active packet. The active packet specifies the destination, the bandwidth

needed, the price offered, etc. According to the information, the FEE needs to determine the LSP and whether the offer should be accepted. A possible decision policy will be introduced in the following section. Generally, the FEE may also make the decision based on other factors, such as the existing traffic condition. Following the acceptance of an offer, the specified bandwidth of the LSP will be reserved accordingly. The active packet then goes back to the sender to initiate the data transmission (i.e., MPLS packets can then be forwarded).

4.3. Markov Decision Model

We assume that the LSP has a limited amount of bandwidth and that each session (or offer) needs a fixed amount of it. Here, the bandwidth needed for a session is assumed to be one unit. Upon receiving an offer, the network provider must decide whether to accept it. If an offer is accepted, the network provider will give away one unit of bandwidth of the LSP but earns an income at the offered price. Note that as the LSP has a limited amount of bandwidth, the network provider may, after accepting an offer, give up the opportunity to get a later higher offer. Figure 4.1 shows a simple example to illustrate this situation where there is only one unit of bandwidth left in an LSP. If the network provider accepts the first offer at a price of 2 units, it will not be able to accept the second higher offer.

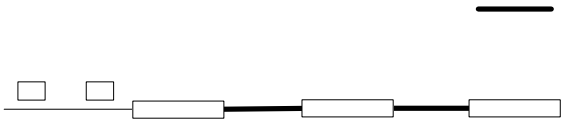


Fig. 4.1 A bandwidth selling problem

Following the approaches/notations in [38,43], the Markov decision model for the aforementioned bandwidth problem is outlined as follows. Note that to set up the model, we need to make some assumptions. A discrete time system is used. At each time point, with a probability of χ, no offer arrives (i.e., an offer arrives with a probability of 1-χ). For an offer, the offered price is φ with probability $\Omega(\varphi)$. A session may end at a time point with probability $1/\tau$. This means that the mean duration of a session is τ time units. We define (ω , φ) as the system state, where $\omega \in \{0, 1, 2, \ldots, W\}$ is the number of remaining units of bandwidth in the LSP, and $\varphi \in \{\eta_1, \eta_2, \ldots \eta_l\}$ is the offered price. For modeling purposes, we use $\varphi = \eta_0 = 0$ to indicate that no offer arrives. At a time point, we define that the allocated bandwidth is β (i.e., $\beta = 1$ or $\beta = 0$). Obviously, we have $\beta = 0$ if no offer arrives or if all the bandwidth has been reserved.

Define $\psi((\omega , \varphi), \beta)$ as the network provider's earnings if the system state is (ω , φ) and the allocated bandwidth is β. It is expressed as follows:

$$\psi((\omega,\varphi),\beta) = \begin{cases} \varphi, \text{if } \beta = 1 \\ 0, \text{if } \beta = 0 \end{cases} \quad \text{.....(4.1)}$$

If an incoming offer is accepted, the network provider will earn φ. If not, the network provider will earn nothing.

Denote $T((\omega', \varphi')|(\omega, \varphi), \beta)$ as the state transition probability from the current state, (ω, φ), to the next state, (ω', φ'), if the assigned bandwidth is β. It can be found by the following expression:

$$T((\omega',\varphi') \mid (\omega,\varphi),\beta) = \begin{cases} (1-\chi)\Omega(\varphi')\binom{W-\omega}{e}(1/\tau)^e(1-1/\tau)^{W-\omega-e} \text{ if } \varphi' > 0 \\ \chi\binom{W-\omega}{e}(1/\tau)^e(1-1/\tau)^{W-\omega-e} \text{ if } \varphi' = 0 \end{cases} \quad \text{.....(4.2)}$$

where $\omega' = \omega + e - \beta$, with e being the number of ending sessions (i.e., each of them release one unit of bandwidth). Note that the remaining bandwidth in the next time point is determined by summing the current amount of unused bandwidth with the released amount of bandwidth and then subtracting the resultant bandwidth from β. As there are $W - \omega$ sessions, the probability that e sessions release their reserved bandwidth is $\binom{W-\omega}{e}(1/\tau)^e(1-1/\tau)^{W-\omega-e}$. Note also that $(1 - \chi)\Omega(\varphi')$ is the probability that the price for the next offer is φ'.

To find the policy that can maximize the expected network provider's earnings, we analyze the bandwidth selling problem based on the Markov decision (infinite horizon) model (see [38,43] for details). According to (4.1) and (4.2), the policy iteration method given in [38,43] can be employed to determine the optimal policy and the expected earnings using a certain discount factor.

4.4. Discussion of results

In this section, the above Markov decision model is employed to analyze the active MPLS service's bandwidth selling problem. We assume that if there is an incoming offer, the offered price follows a discrete normal distribution with a mean of μ and a standard deviation of σ. We compare the earnings and the rejection ratio of the above Markov decision-based selling policy with that of the allocate-if-above-mean scheme (i.e., allocate if there is sufficient bandwidth and the offered price is above the mean price) and of the allocate-if-available scheme (i.e., always allocate if there is sufficient bandwidth). Moreover we study the factors that affect the earnings and the rejection ratio.

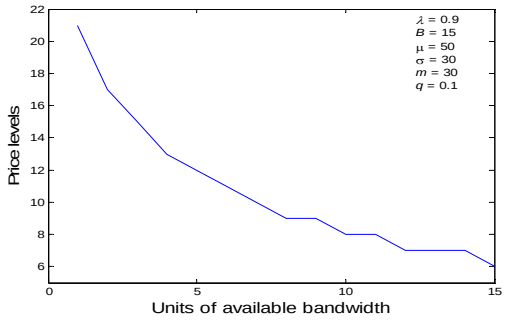


Fig. 4.2 Markov decision-based bandwidth selling policy

For given values of l, W, μ, σ, τ and χ, it is possible to find the Markov decision-based bandwidth selling policy. Figure 4.2 shows the Markov decision-based bandwidth selling policy where l = 50, W = 15, μ = 50, σ = 30, τ = 30 and χ = 0.1. The discount factor, f, is set to be 0.9. From that graph, the network provider can determine whether an offer should be accepted when the remaining amount of bandwidth and the offered price is known. For instance, if there are five units of bandwidth left, a new session should not be admitted unless the offered price is η_{12} or above. As expected, the threshold value decreases with the increase in the amount of remaining bandwidth.

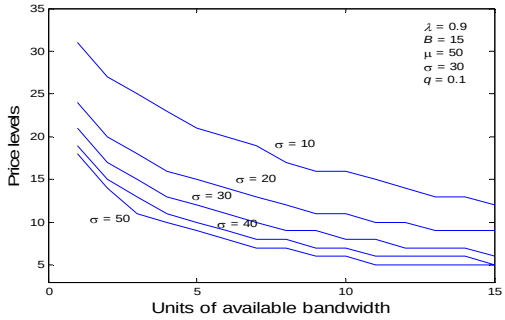
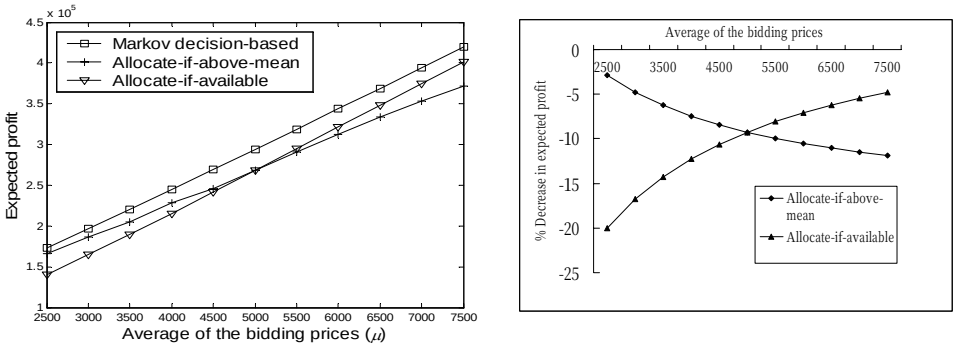


Fig. 4.3 Markov decision-based bandwidth selling policy for different σ

Figure 4.3 shows the threshold price value for accepting an offer when σ is varied. It can be seen that the threshold value decreases as σ increases. If the value of σ is large, the probability of receiving an offer with a relatively low

price will be larger than it would be when σ is smaller.

Figures 4.4 – 4.13 show the variation in the expected earnings and the rejection ratio of the Markov decision-based bandwidth selling policy, the allocate-if-above-mean scheme, and the allocate-if-available scheme. We set the base parameters as W = 10, μ = 5000, σ = 2000, τ = 20, χ= 0.1 and f = 0.99.



(a) (b)

Fig. 4.4 Expected earnings for different μ

Figure 4.4(a) shows the changes in the expected earnings of the different schemes with different values of μ. We can see that for all the schemes, the expected earnings rise when μ increases and that the Markov decision-based selling policy produces the highest earnings. However, the difference between the Markov decision-based selling policy and the allocate-if-available scheme decreases with an increase in μ. This is because when the value of μ is low, the offered prices are likely to be lower. Therefore it is preferable to impose a more selective policy, leaving room to accept a higher offer later. However, as indicated by the expected earnings of the allocate-if-above-mean scheme, if the threshold price is too high, the earnings will decrease significantly. Hence the allocate-if-above-mean scheme is not effective in general. Figure 4.4(b) shows the percentage decrease in expected profits of the allocate-if-above-mean and allocate-if-available schemes compared with the Markov-decision-based scheme for relative performance comparison.

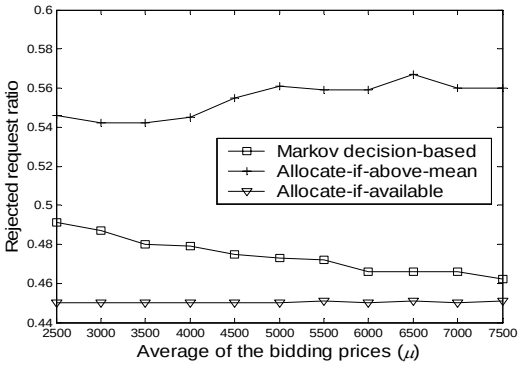
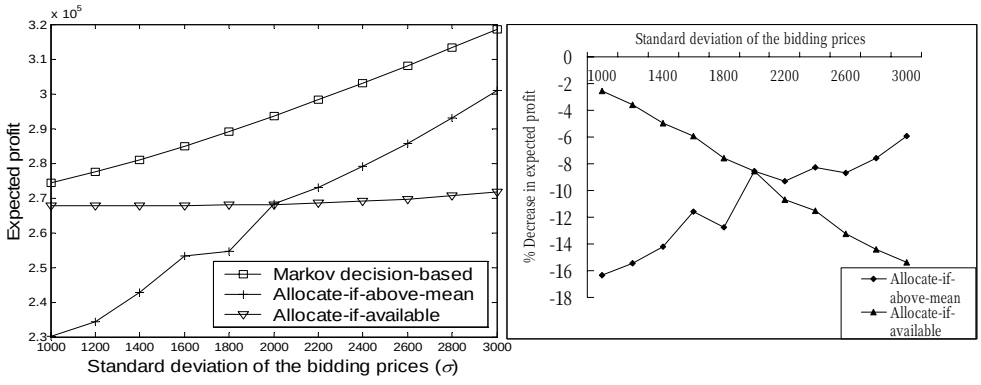


Fig. 4.5 Rejection ratio for different μ

Figure 4.5 shows the change in the rejection ratio when μ varies. We can see that the allocate-if-above-mean scheme has the largest rejection ratio of the three schemes. On average, the rejection ratio of the Markov decision-based selling policy is higher than that of the allocate-if-available scheme by 3 – 4%. However for small values of μ, the expected earnings of the Markov decision-based selling policy outperform those of the allocate-if-available scheme by 4 – 23%. Thus the small increase in the rejection ratio can produce a greater increase in earnings. For larger values of μ, the difference in the rejection ratio decreases.



(a) (b)

Fig. 4.6 Expected earnings for different σ

Figure 4.6(a) shows the change in the expected earnings of the different schemes with different values of σ. The expected earnings of the Markov

decision-based policy and the allocate-if-above-mean scheme increase with an increase in σ, but the expected earnings of the allocate-if-available scheme are independent of σ. For a selective policy, an increase in σ will result in greater earnings because the variation in the offered prices is much larger. For example, if there is only one unit of bandwidth left, it is better to reject a low offer. This is because it will leave room for a possible higher offer later. If the later offered price is higher, the increase in earnings will be larger due to the large price difference. Since there is no selection criterion in the allocate-if-available scheme, the expected earnings are dependent only on the average offered price. The result indicates that the expected earnings of the Markov decision-based selling policy are larger than those of the other two schemes. The difference in the expected earnings from the Markov decision-based policy and from the allocate-if-available scheme broadens as σ increases. As the expected earnings of the allocate-if-above-mean scheme increase as σ increases, and the expected earnings are larger than those of the allocate-if-available scheme for large σ, it can be seen that a selective policy is to be preferred if σ is large. Figure 4.6(b) shows the percentage decrease in expected profits of the allocate-if-above-mean and allocate-if-available schemes compared with the Markov-decision-based scheme for relative performance comparison.

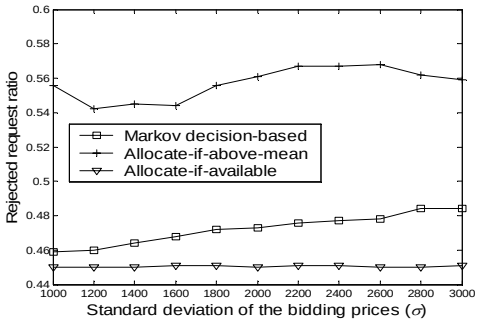
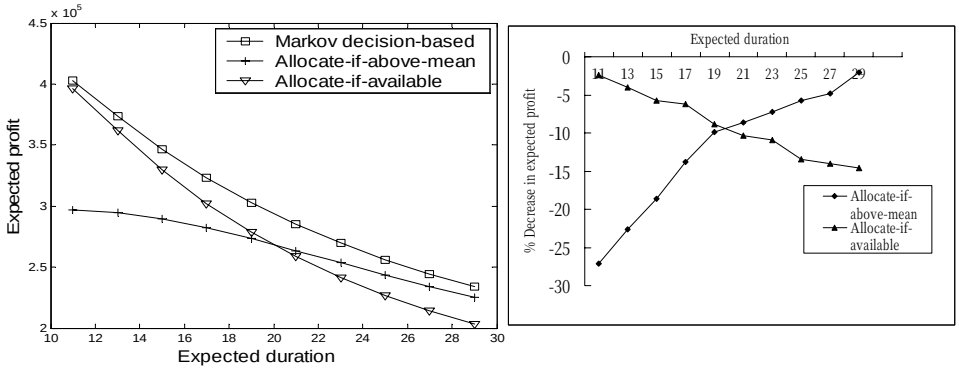


Fig. 4.7 Rejection ratio for different σ

Figure 4.7 shows the change in the rejection ratio when σ varies. The rejection ratio of the Markov decision-based selling policy increases slightly as σ increases (i.e., 1-3% higher than that of the allocate-if-available scheme). However, compared with the allocate-if-available scheme, the increase in the expected earnings of the Markov decision-based selling policy increases from 3% to 17%. Hence when the standard deviation of the offered price is high, it is much better to use a more selective policy.



(a) (b)

Fig. 4.8 Expected earnings for different τ

Figure 4.8(a) shows the variation of the expected earnings of the three schemes using different values for the mean duration of the session. For all the schemes, the expected earnings decrease as the expected duration increases. This is because a session is less likely to release its reserved bandwidth. As a result, only fewer sessions can be supported. Yet, the difference in the

expected earnings from the Markov decision-based selling policy and from the allocate-if-available scheme increase when the session duration is longer. One possible explanation for this is that when the session duration is longer, the bandwidth is more valuable so a more selective policy can ensure that the valuable bandwidth can be granted at a higher price. This explanation is supported by the observation that the expected earnings of the allocate-if-available scheme are greater than those of the allocate-if-above-mean scheme when τ is larger. Figure 4.8(b) shows the percentage decrease in expected profits of the allocate-if-above-mean and allocate-if-available schemes compared with the Markov-decision-based scheme for relative performance comparison.

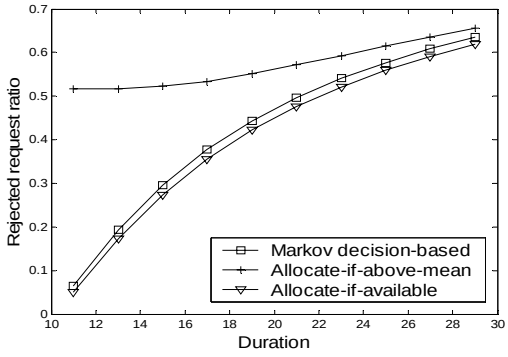
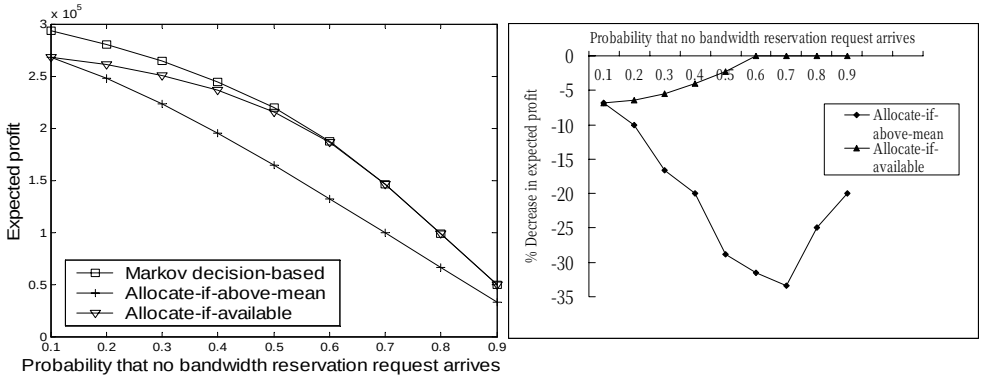


Fig. 4.9 Rejection ratio for different τ

Figure 4.9 shows the variation of the rejection ratio when the session duration changes. The rejection ratio of all schemes increases along with the session duration but they have almost the same values when the session duration is long.



(a) (b)

Fig. 4.10 Expected earnings for different χ

Figure 4.10(a) shows the changes in the expected earnings of the different schemes with different values of χ. The expected earnings of the three schemes decrease when χ increases since light traffic implies less choice in offered prices. Another observation is that the difference in the expected earnings of the three schemes decrease as χ increases. When χ is low, the Markov decision-based selling policy is more likely to assign bandwidth to a session even if the offered price is low. This is because the network provider may need to wait for a long time for the next offer. In this case, the Markov decision-based bandwidth selling policy is close to that of the allocate-if-available policy. Figure 4.10(b) shows the percentage decrease in expected profits of the allocate-if-above-mean and allocate-if-available schemes compared with the Markov-decision-based scheme for relative performance comparison.

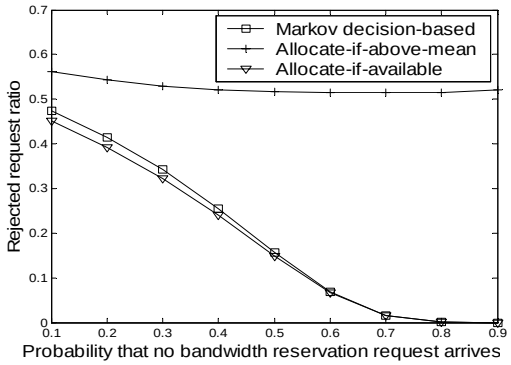
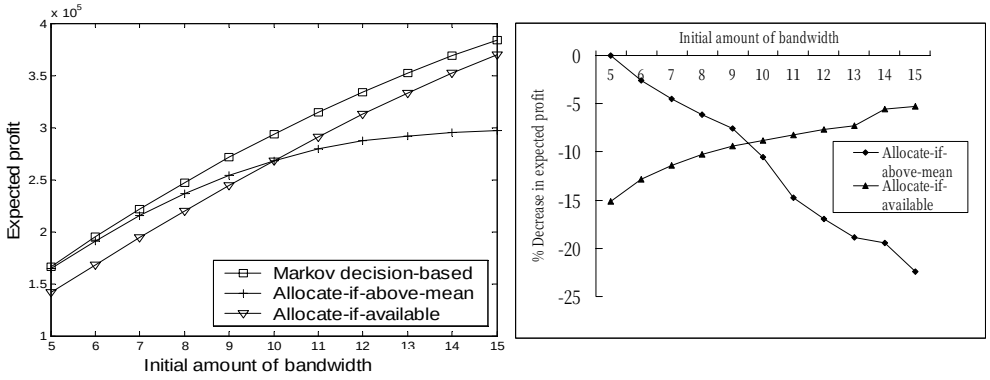


Fig. 4.11 Rejection ratio for different χ

Figure 4.11 shows the change in the rejection ratio with a change in χ. The rejection ratio of the Markov decision-based policy and of the allocate-if-available scheme decreases with an increase in χ. The two ratios become equal when the value of χ is large. This is reasonable because, as explained before, with a large value of χ, both schemes behave more or less the same. The allocate-if-above-mean scheme has the highest rejection ratio since it takes into account only the offered price.



(a) (b)

Fig. 4.12 Expected earnings for different *W*

Figure 4.12(a) shows the change in the expected earnings of the three schemes with different values of *W*. The expected earnings increase with an increase in *W* for all the three schemes. Yet, the difference in the expected earnings between the Markov decision-based policy and the allocate-if-available

scheme decreases with the value of *W*. This is because when *W* is large and sessions are relatively short, there is no need to reserve bandwidth for the possible higher offers. Thus the two schemes produce almost the same performance for large *W*. The rate of increase in the expected earnings of the allocate-if-above-mean scheme decreases with *W*. This indicates that there is no need to employ a highly selective policy when bandwidth is abundant. Since the expected earnings of the allocate-if-above-mean scheme are greater than those of the allocate-if-available scheme for small values of *W*, we can deduce that a selective policy would increase the expected earnings more significantly for small *W*. Figure 4.12(b) shows the percentage decrease in expected profits of the allocate-if-above-mean and allocate-if-available schemes compared with the Markov-decision-based scheme for relative performance comparison.

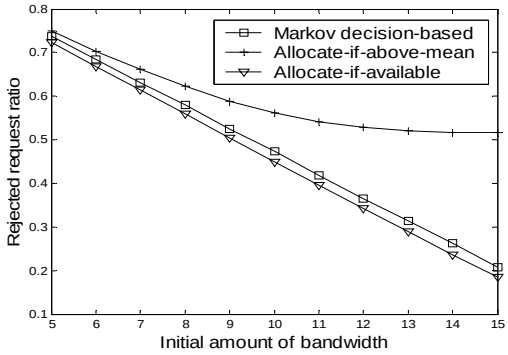


Fig. 4.13 Rejection ratio for different *W*

Figure 4.13 shows the change in the rejection ratio when *W* varies. As expected, the ratio for all the schemes decreases as the bandwidth increases. It can also be seen that the rejection ratio of the allocate-if-above-mean scheme is still the highest because of its highly selective policy.

4.5. Conclusion

We have presented an active MPLS service with bandwidth reservation based on an adaptive pricing method. To increase the earnings of the network provider, a bandwidth selling policy has been determined based on a Markov decision model. We have also presented analytical results comparing the selling policy with two other schemes. In accordance with simulations, we have also investigated the tradeoff in terms of the rejection ratio when employing the proposed policy. Generally, at the expense of a small increase in the rejection ratio, the selling policy produces increased earnings when the mean offered price is small, the standard deviation of the offered price is large, and the mean session duration is long.

Chapter 5. Load Balancing of MPLS Paths

5.1. Introduction

Recent years have seen the introduction of multi-protocol label switching (MPLS) [6] which employs labels in determining the next-hop of a packet. The essence of MPLS is to attach the label before the long and clumsy network layer header so that only a simple comparison between the label and the switching table is needed to switch the packet towards the destination. The distribution of labels will set up paths known as label switch paths (LSPs) along the network, and the setting up of these LSPs can facilitate traffic engineering which involves finding suitable paths (QoS routing) and assigning suitable load in each of these paths (load balancing) if there is more than one LSP between the same source-destination pair.

In the context of QoS routing in the MPLS network, [44] had surveyed several simple algorithms like dynamic routing with partial-information (DR-PI), dynamic restorable routing, minimum interference routing algorithm (MIRA) and profile-based routing (PBR). [45] modified MIRA by estimating the bandwidth requirement of future LSPs. [25] employed mobile agents in setting up multiple LSPs in the DiffServ network. All these works assumed there is only one LSP between the same source-destination pair.

In this chapter, we apply a modified network simplex algorithm in the ISDN3 [29] network which can employ the traditional network simplex method to not only find multiple MPLS paths for a user application, but also balance the load to obtain aggregate minimum network delay for the user application. In ISDN3, active packets can be used to implement our modified network simplex algorithm and to setup the concerned LSPs as in [30]. Preliminary results comparing the proposed scheme with other simplex schemes are also presented.

5.2. Modified Network Simplex Algorithm

5.2.1. Model

We consider a network represented by a graph $G = (V, E)$, where V is the set of MPLS switches and E is the set of network links. For each network link $(i, j) \in E$, we define μ_{ij} as the capacity of the link, which can be regarded as the processing rate of the link. We also define x_{ij} be the amount of load on it. In addition, we define $c_{ij}(x_{ij})$ be the cost associated with a amount of x_{ij} being placed in the link (i, j), where $c_{ij}(\cdot)$ can be any convex function, such- as $c_{ij}(x_{ij}) = 1/(\mu_{ij} - x_{ij})$ if we employ the M/M/1 queuing delay model. For each switch i, we denote $b(i)$ as the the net outflow of node i. A negative value means a net inflow.

In our model, we assume there are n MPLS switches in the MPLS network. For simplicity, we also assume the source switch is switch 1, and the destination switch is n. There is a poisson traffic input in node 1 with average

rate λ. As a result, $b(1) = \lambda$ and $b(n) = -\lambda$. We want to find out the paths and the corresponding amount of flow in each path that can minimize the aggregate delay in the network, so the problem is equivalent to the minimum cost flow problem in graph theory:

Minimize $c_{ij}(x_{ij})$ .....(5.1)

Subject to $0 \le x_{ij} < \mu_{ij} \forall i, j$ .....(5.2)

$$\sum_k x_{ik} - \sum_k x_{ki} = b(i) \quad \text{.....(5.3)}$$

and $b(1) = \lambda$, $b(n) = -\lambda$.....(5.4)

If the cost functions associated to the network links are linear to the flow amount assigned, the network simplex algorithm can be used to find the minimum cost flow given (5.1) – (5.4). However, cost associated with a link in general is not linear, thus we have to modify the network simplex algorithm to make it practical in any type of cost functions.

To deal with this problem, we can consider the unit-increment cost of each link instead of the per-unit cost of each link. This is equivalent to dividing the domain of the delay function into partitions, with the width of each partition being one unit. As a result, each partition can be roughly regarded as a linear function with the flow assigned. In the context of the network, it is equivalent to transform each link into several links, with each of the transformed link having a linear cost function. Now the problem is equivalent to the following:

Minimize $\sum_l C_{ij}^l(x_{ij}^l)$ .....(5.5)

Subject to $x_{ij}^l = \{0,1\} \forall i, j, l$ .....(5.6)

$$\sum_{k,l} x_{ik}^{l} - \sum_{k,l} x_{ki}^{l} = b(i) \quad \text{.....(5.7)}$$

and $b(1) = \lambda,\ b(n) = -\lambda$ .....(5.8)

where $x_{ij} = \sum_{l} x_{ij}^{l}$ and $c_{ij}(x_{ij}) = \sum C_{ij}^{l}(x_{ij}^{l})$ .....(5.9)

Yet the traditional network simplex algorithm cannot be applied directly to the modified minimum cost flow problem. We describe the modified network simplex algorithm in the next subsection.

5.2.2. Algorithm

In the network simplex algorithm, an initial flow is assigned according to (5.2) – (5.4) first. Then according to the flow, the links are divided into three categories:

Tree links (**T**): A spanning tree of the network, where $0 \le x_{ij} < \mu_{ij}$.

Upper links (**U**): All the links except the tree links with $x_{ij} = \mu_{ij} - 1$. Flow cannot be increased in these links.

Lower links (**L**): All the links except the tree links with $x_{ij} = 0$. Flow cannot be decreased in these links.

Each link $(i, j) \in \mathbf{U} \cap \mathbf{L}$ will form a cycle with links in **T**. The network simplex algorithm then checks if the following conditions exist:

For any link $(i, j) \in \mathbf{U}$, if decreasing flow on *e* (which will then increase flow

in the rest of the cycle) can decrease the cost of the overall flow, then decrease the flow of *e* as large as possible, and increase the flow of the rest of the cycle accordingly.

For any link $(i, j) \in \mathbf{L}$, if decreasing flow on *e* (which will then decrease flow in the rest of the cycle) can decrease the cost of the overall flow, then increase the flow of *e* a1+s large as possible, and increase the flow of the rest of the cycle accordingly.

After the change of the flow, update the three categories **T**, **U** and **L** accordingly. The algorithm terminates when none of the conditions exists.

Yet the above algorithm cannot work with the modified minimum cost flow problem. It is because the cost function is not linear, in updating the flow in edges in **U** or **L** we cannot just decrease or increase the maximum possible amount of flow in the edges. Thus we have made the following modification to the algorithm:

1. In addition to the three categories, we add a fourth category – Free links (**F**). Free links include all the links except the tree links with $0 \le x_{ij} < \mu_{ij}$.
2. In updating the flow on links in **U** or **L**, we try to search the maximum amount of increment or decrement that can reduce the aggregate cost most. This is not necessarily equal to the maximum possible amount of increment or decrement.
3. Since the flow of any link $(i, j) \in \mathbf{F}$ can both be increased and decreased, we have to test both the increment and decrement of flow to see if the

change can reduce the aggregate flow cost.

5.2.3. Implementation

We consider implementing the modified simplex method in the ISDN3 network. In the ISDN3 network, active packets known as aglets can be used to gather the network links information and setup the corresponding LSPs. The calculation and comparison of the cost between links can be done in the FEEs.

5.3. Analysis

We analyze the modified network simplex method with three other simple load balancing schemes. We study the scenario where three LSPs have already been setup between the source and destination. The scenario is shown in Figure 5.1.

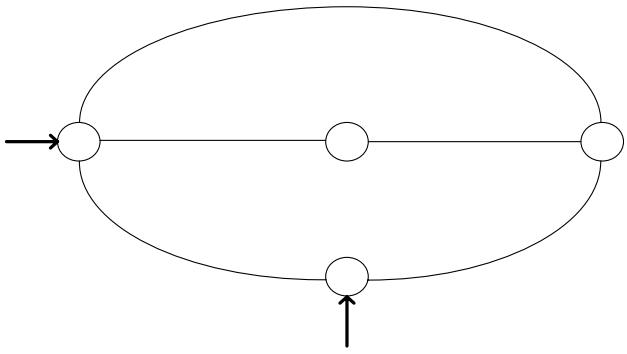


Fig. 5.1 Scenario under analysis

In the network under analysis, the three LSPs refer to three different types of LSPs. LSP1, which consists of edge (1, 4), refers to a high speed LSP. LSP 2, which consists of edge (1, 2) and (2, 4), refers to a standard path with moderate bandwidth. LSP 3, which consists of edge (1, 3) and (3, 4), refers to an LSP which is also shared by other traffic. We want to study the preference

in LSP when the traffic input is increased. We also want to compare the performance of the modified simplex method with other simple load balancing schemes.

5.3.1. Schemes Compared

We compare the modified simplex network algorithm with three other simple schemes. The three schemes are as follows:

1. *Even*: The source traffic flow is assigned to the three LSPs evenly, i.e. 1 / 3 of the total traffic flow is assigned to each LSP. In case the capacity of an LSP is less than 1 / 3 of the input traffic, the surplus traffic of that particular LSP is shared evenly among the rest LSPs.
2. *Proportion*: The source traffic flow is assigned to the three LSPs in proportion to the capacity of each traffic flow.
3. *Shortest path*: The source traffic flow is assigned to the high speed LSP first if there is still room for the traffic. If the source traffic exceeds the capacity of the high speed LSP, the surplus traffic is assigned to LSP 2, and so on.

5.3.2. Results

We set the parameters of the network under analysis as follows: $b(3)$=50MB/s, $\mu_{14} = 108$ MB/s, $\mu_{12} = 51$ MB/s, $\mu_{24} = 51$ MB/s, $\mu_{13} = 51$ MB/s and $\mu_{34} = 101$ MB/s. Thus the capacity for LSP 1 is 108MB/s, that of LSP 2 is 51MB/s and that of LSP 3 is 51MB/s. Also we assume the delay function of each link follows the M/M/1 queuing delay model. We vary the traffic input, λ MB/s, to study the effect on the aggregate delay of the network.

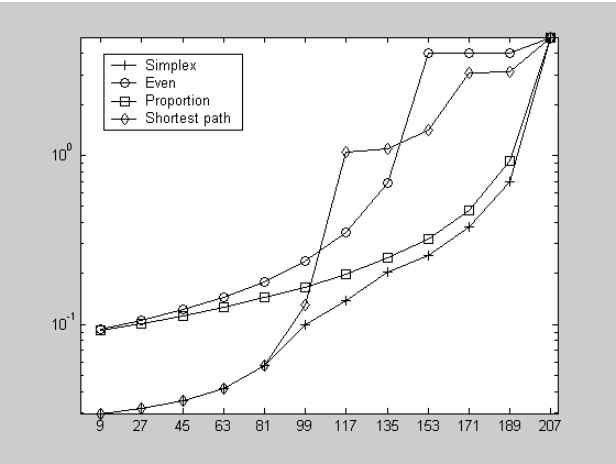


Fig. 5.2 Aggregate delay of different schemes

Figure 5.2 shows the variation in the aggregate delay of each scheme with a change of the traffic input. From the graph, we can see that the modified network simplex method outperforms or at least has the same performance as all other schemes no matter how much the input traffic is. We can analyze the figure by dividing it into three zones.

When the input traffic load is low, the modified network simplex algorithm and the shortest path scheme have the same promising performance. Yet the traditional load balancing method like even assignment and proportional assignment behave unsatisfactory. We can conclude that for low input traffic rate, it is not intelligent to balance the load among the different LSPs.

As the input traffic load increases, the modified network simplex algorithm starts to outperform the shortest path scheme. In addition as the input traffic rate increases further, the even assignment and proportional assignment schemes can perform better than the shortest path scheme. It is because when the input traffic load reaches a moderately large value, it becomes intelligent to

balance the load among the different LSPs. And we can see that the modified network simplex method algorithm has a preference to balance the load to the LSP which is also shared by other traffic. It is because it is preferable to minimize the number of links that have traffic flowing on it. The majority of the traffic is still assigned to the high speed LSP.

When the input traffic load reaches a relatively high rate, the even assignment and shortest path scheme perform badly as compared to the proportional assignment and the modified network simplex algorithm. The difference in aggregate delay between the proportional assignment and the modified network simplex algorithm becomes smaller as the input traffic load increases. We can see that it is still preferable to place the majority of the traffic load in the high speed LSP and then evenly distribute the rest between LSP 2 and LSP 3.

5.4. Conclusion

To conclude, we have presented the modified network simplex algorithm and compared its performance with three other simple load balancing schemes. We can see that it is not worthwhile to do load balancing when the traffic load is low. Among all the schemes, the modified network simplex algorithm is the most adaptive.

Chapter 6. Conclusion

We have presented the background of this project, including the working mechanisms of the present routing protocols, active network and mobile agent technologies, MPLS, ISDN3 and some of the literature on the use of mobile agents to assist network routing.

We have presented an active routing service in Chapter 3 for the next-generation network called ISDN3, which involves using active packets (packlets) to configure customized communication paths based on the network information. With the aim of minimizing network costs, both a finite horizon and an infinite horizon Markov decision model have been formulated to support active routing. According to the models, the communication paths should be reconfigured if a particular cost saving threshold is reached. Theoretical analysis confirms that by using the proposed routing policy, the average network cost can be minimized under various conditions as compared with other approaches. Yet we have not considered the pricing issue, i.e. network users are free to reserve any portion of the network.

As a result, we have also presented an active MPLS service with bandwidth reservation using a dynamic pricing mechanism in Chapter 4. With the aim of maximizing the profits of the network service provider, an optimal bandwidth selling policy has been obtained for both the finite and infinite horizon models based on the Markov decision theory. In both models the optimal policy is

threshold-based with the threshold depending mainly on the amount of remaining bandwidth. We have also presented analytical results to demonstrate the advantage of the proposed policy over the conventional “allocate-if-available” method. In particular, the proposed policy generates increased profits when the mean price is small, the standard deviation of the prices is large, and the expected session duration is long. It is expected that the Markov decision model will also be adaptable to the analysis of similar bandwidth selling problems in other networking scenarios.

We have focused on only one LSP in Chapter 4. If we consider more LSPs concurrently, the rejected reservation may be accommodated by other not-so-busy LSPs. In this scenario, the network operator needs to consider not only the issue of profit maximization but also the issue of load balancing. If there are a set of LSPs that can accommodate the user application's request, which one of the LSPs should be used? As a result, we tried to deal with this scenario in Chapter 5. In that chapter, we apply a modified network simplex algorithm in the ISDN3 network which can employ the traditional network simplex method to not only find multiple MPLS paths for a user application, but also balance the load to obtain aggregate minimum network delay for the user application. In ISDN3, active packets can be used to implement our modified network simplex algorithm and to setup the concerned LSPs. Preliminary results comparing the proposed scheme with other simplex schemes are also presented. We can see that it is not intelligent to do load balancing when the traffic load is low. Of all the schemes, the modified network simplex algorithm is the most adaptive.

Much of the works presented in this thesis are in theory. Further work can be done on implementing the actual mobile agent-based routing for the next-generation Internet. Simulation results can thus be collected to help us study the various aspects of mobile agent-based routing for the next-generation Internet.

REFERENCES

[1] M. Steenstrup, *Routing in communications networks*, Prentice Hall, 1995.

[2] Kyas and G. Crawford, *ATM networks*, 1st edition, Prentice Hall, 2002.

[3] W.R. Stevens, *TCP/IP illustrated volume 1*, Addison Wesley Longman, 1994.

[4] H.C.B. Chan, H. M. Alnuweiri and V.C.M. Leung, "A framework for optimizing the cost and performance of next-generation IP routers," *IEEE Journal of Sel. Areas in Comm.*, vol. 17, no. 6, pp. 1013-1029, Jun. 1999.

[5] Partridge *et al.*, "A 50-Gb/s IP router," *IEEE/ACM Trans. on Networking*, vol. 6, no. 3, pp. 237-248, Jun. 1998.

[6] E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," *Internet RFCs 3031*, Jan. 2001.

[7] U.D. Black, *MPLS & label switching networks*, 2nd edition, Prentice Hall, 2002.

[8] L.L. Peterson and B.S. Davie, *Computer Networks: A Systems Approach*, 2nd edition, Morgan Kaufmann, 2000.

[9] G. Malkin, "RIP version 2," *Internet RFCs 2453*, Nov. 1998.

[10] J. Moy, "OSPF version 2," *Internet RFCs 2328*, Apr. 1998.

[11] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," *Internet RFCs 1771*, Mar. 1995.

[12] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64-79 Nov./Dec. 1998.

[13] K.L. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz, "Directions in active networks," *IEEE Comm. Mag.*, pp. 72-78, Oct. 1998.

[14] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden, "A survey of active network research," *IEEE Comm. Mag.*, vol. 35, no. 1, pp. 80-86, Jan. 1997.

[15] M. Ott, G. Welling, S. Mathur, D. Reininger, and R. Izmailov, "The JOURNEY active network model," *IEEE Journal on Sel. Areas in Comm.*, vol. 19, no. 3, pp. 527-537, Mar. 2001.

[16] S. da Silva, Y. Yemini, and D. Florissi, "The NetScript active network system," *IEEE Journal on Sel. Areas in Comm.*, vol. 19, no. 3, pp. 538-551, Mar. 2001.

[17] N.F. Maxemchuk and S.H. Low, "Active routing," *IEEE Journal on Sel. Areas in Comm.*, vol. 19, no. 3, pp. 552-565, Mar. 2001.

[18] C. Tschudin, H. Lundgren and H. Gulbrandsem, "Active routing for ad hoc networks," *IEEE Comm. Mag.*, pp. 122-127, Apr. 2000.

[19] R.Y.W. Lam, H.C.B. Chan, V.O.K. Li, T.S. Dillon and V.C.M. Leung, "Active routing service for the next generation network/ISDN3," *Proc. IEEE GLOBECOM'02*, vol. 3, pp. 2375-2379, Nov. 2002.

[20] D.B. Lange, and M. Oshima, M, *Programming and Deploying Java Mobile Agents with Aglets*, Addison Wesley Longman, 1998.

[21] Wave: http://www-zorn.ira.uka.de/wave/wave.html

[22] G.D. Caro and M. Dorigo, "Mobile Agents for Adaptive Routing," *Proc. 31st Int'l. Conf. Sys. Hawaii*, vol. 7, pp. 74-83, Jan. 1998.

[23] N. Minar, K.H. Kramer and P. Maes, "Cooperating Mobile Agents for Dynamic Network Routing," Software Agents for Future Communications Systems, Springer-Verlag, 1999.

[24] K. Oida and M. Sekido, "ARS: An Efficient Agent-Based Routing System for QoS Quarantees," *Comp. Commun.*, vol. 23, no. 14-15, pp. 1437-1447, Aug. 2000.

[25] S.G. Valenzuela and V.C.M. Leung, "QoS Routing for MPLS Networks Employing Mobile Agents," *IEEE Network*, vol. 16, no. 3, pp. 16-21, May/June 2002.

[26] N. Migas, W.J. Buchanan and K.A. McArtney, "Mobile Agents for Routing, Topology Discovery, and Automatic Network Reconfiguration in Ad-Hoc

Networks," *Proc.10th IEEE Int'l Conf. and Workshop on the Engineering of Computer-Based Systems*, pp. 200-206, Apr. 2003.

[27] A. Selamat and S. Omatu, "Route Selection by Mobile Agents for Query Retrieval Using Genetic Algorithm," *Proc. 2003 IEEE Int'l Symp. On Computational Intelligence in Robotics and Automation*, pp. 854-859, Jul. 2003.

[28] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *ACM Computer Communications Review*, vol. 26, pp. 5 – 18, Apr. 1996.

[29] R. Y. W. Lam, *Quantum Packet for Integrated Services Data Network - the Third Generation ISDN*, MPhil Thesis, Department of Computing, The Hong Kong Polytechnic University, 2003.

[30] W. T. Liu and H. C. B. Chan, "Design of an active forwarding scheme for ISDN3," *Proc. IEEE Int'l Conf. on Systems, Man and Cybernetics, the Hague*, Oct. 10-13, 2004.

[31] K.Rothermel and R. Popescu-Zeletin, Eds., *Mobile Agents*, vol. 1219, Springer, 1997.

[32] G.C. Kessler and P.V. Southwick, *ISDN: Concepts, Facilities and Services*, 4th edition, McGraw Hill, 1998.

[33] M. Kawarasaki and B. Jabbari, "B-ISDN architecture and protocol," *IEEE Journal on Sel.Areas in Comm.*, vol. 9, no. 9, pp. 1405-1415, Dec. 1991.

[34] H.C.B. Chan, R.Y.W. Lam, T.S. Dillon, V.O.K. Li, and V.C.M. Leung, "ISDN3: The next generation networks," *Proc. IEEE PACRIM'01*, vol. 2, pp. 607-610, Victoria, B.C., Canada, 2001.

[35] K.M. Sim and W.H. Sun, "Multiple Ant-Colony Optimization for Network Routing," *Proc. 1st IEEE Int'l Symp. On Cyber Worlds*, pp. 271-281, Nov 2002.

[36] Akashi *et al*, "Agents Support for Flexible Inter-AS Policy Control," *Proc. IEEE 2003 Symp. On Applications and the Internet Workshops*, pp. 294-298, Jan. 2003.

[37] T.R. Nieto, H.M. Deitel, P.J. Deitel, H. Deitel, and P. Deitel, *Complete Internet and World Wide Web programming training course*, Prentice Hall, 2000.

[38] M.L. Puterman, Markov decision processes: discrete stochastic dynamic programming, 1st edition, Wiley-Interscience, 1994.

[39] H. Jiang and S. Jordan, "A pricing model for high-speed networks with guaranteed QoS," *Proc. IEEE INFOCOM'96*, pp. 888-95, Mar. 1996.

[40] P. Kirkby, "Business models and system architectures for future QoS guaranteed Internet services," *IEE Colloquium on Charging for ATM – The Reality Arrives*, pp. 11/1-11/8, 1997.

[41] A. M. Odlyzko, "Paris metro pricing for the Internet," *Proc. of the ACM Conference on Electronic Commerce*, pp. 140-147, 1999.

[42] P. Maille and B. Tuffin, "Multi-bid auctions for bandwidth allocation in communication networks," *Proc. IEEE INFOCOM 2004*, Mar. 2004.

[43] M. L. Puterman, "Dynamic programming," *Encyclopedia of physical science and technology*, vol. 4, Academic Press Inc., pp. 438 – 463, 1987.

[44] J. L. Marzo, E. Calle, C. Scoglio and T. Anjali, "QoS online routing and MPLS multilevel protection: a survey," *IEEE Communications*, vol. 41, no. 10, Oct. 2003, pp. 126 – 132.

[45] Y-X. Xu and G-D. Zhang, "Models and algorithms of QoS-based routing with MPLS traffic engineering," *5th IEEE Int'l. Conf. on High Speed Networks and Multimedia Communications*, Jul. 2002, pp. 128 – 132.