DYNAMIC CAPACITY ADJUSMENT FOR VIRTUAL-PATH BASED NETWORKS USING NEURO-DYNAMIC PROGRAMMING

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Cem Şahin

September, 2003

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Nail Akar (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ömer Morgül

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Ezhan Karaşan

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute Engineering and Science

ABSTRACT

DYNAMIC CAPACITY ADJUSMENT FOR VIRTUAL-PATH BASED NETWORKS USING NEURO-DYNAMIC PROGRAMMING

Cem Şahin
M.S. in Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Nail Akar
September, 2003

Dynamic capacity adjustment is the process of updating the capacity reservation of a virtual path via signalling in the network. There are two important issues to be considered: bandwidth (resource) utilization and signaling traffic. Changing the capacity too frequently will lead to efficient usage of resources but has a disadvantage of increasing signaling traffic among the network elements. On the other hand, if the capacity is adjusted for the highest possible value and kept fixed for a long time period, a significant amount of bandwidth will be wasted when the actual traffic rate is small. We proposed two formulations for dynamic capacity adjustment problem. In the first formulation cost parameters are assigned for bandwidth usage and signalling, optimal solutions are reached for different values of these parameters. In the second formulation, our aim is to maximize the bandwidth efficiency with a given signaling requirement. In this formulation, a leaky bucket counter is used in order to regulate the signaling rate. We used dynamic programming and neuro-dynamic programming techniques and we applied our formulations for voice traffic scenario (voice over packet networks) and a general network architecture using flow-based Internet traffic modelling. In the Internet traffic modelling case, we tested two different control strategies: event-driven control and time-driven control. In event-driven control, capacity update epochs are selected to be the time instants of either a flow arrival or a flow departure. In time-driven control, decision epochs are selected to be the equidistant time instants and excessive amount of traffic that cannot be carried will be buffered.

Keywords: Dynamic capacity adjustment, virtual path, voice over packet networks, dynamic programming, neuro-dynamic programming, leaky bucket counter, flow-based internet traffic modelling.

ÖZET

SANAL-YOL TABANLI AĞLARDA SİNİRSEL-DİNAMİK PROGRAMLAMA KULLANILARAK DİNAMİK KAPASİTE AYARLANMASI

Cem Şahin
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yard. Doç. Dr. Nail Akar
Eylül, 2003

Dinamik kapasite problemi, sinyalleşme yardımıyla bir sanal yolun kapasite rezervasyonunun değiştirilmesi işlemidir. Gözönüne alınması gereken iki önemli nokta, kaynak kullanımı ve sinyalleşme trafiğidir. Kapasitenin sık bir biçimde değiştirilmesi, etkin bir kaynak kullanımını sağlar fakat bu yöntemin dezavantajı, ağ elemanları arasındaki sinyalleşme trafiğinin artmasıdır. Diğer taraftan, eğer kapasite en yüksek değerine ayarlanıp uzun bir zaman diliminde değiştirilmezse, trafik yoğunluğunun az olduğu zamanlarda büyük miktarda kapasite boşa harcanır. Dinamik kapasite problemi için iki farklı formulasyon önerdik. Birinci formulasyonda, her sinyalleşme maliyeti ve aynı zamanda birim zamanda kullanılan kaynak maliyeti için parametreler atanmıştır ve bu parametrelerin değişik değerleri için optimal çözümlere ulaşılmıştır. İkinci formulasyondaki amacımız, verilen bir sinyalleşme kısıtına uyarak, kaynak kullanım verimini arttırmaktır. Bu formulasyonda sinyalleşme oranını ayarlamak için sızdıran kova sayıcısı kullanılmıştır. Ses trafiği ve genel akış bazlı İnternet trafiği modelleri için, dinamik programlama ve sinirsel-dinamik programlama teknikleri kullanılmıştır. İnternet trafiği senaryosu için, zaman-sürümlü ve olay-sürümlü kontrol stratejileri kullanılmıştır. Olay-sürümlü kontrolda karar zamanları, akışların geliş ve gidiş zamanları olarak atanmıştır. Zaman-sürümlü kontrolde ise karar zamanları eşit aralıklı zaman noktalarıdır ve taşınamayan trafik için tampon sistemi olduğu varsayılmıştır.

Anahtar sözcükler: Dinamik kapasite ayarlanması, sanal yol, paket üzerinde ses ağları, dinamik programlama, sinirsel-dinamik programlama, sızdıran kova sayıcısı, akıntı tabanlı internet trafik modellemesi.

Acknowledgement

I would like to express my gratitude to my supervisor Assist. Prof. Dr. Nail Akar for his instructive comments in the supervision of the thesis.

Contents

List of Figures

List of Tables

Table of Abbreviations

VP	Virtual Path
QoS	Quality of Service
MPLS-TE	Multiprotocol Label Switching - Traffic Engineering
ATM	Asynchronous Transfer Mode
RSVP	Resource Reservation Protocol
LSR	Label Switched Router
VoP	Voice over Packet
PSTN	Public Switched Telephone Network
E2E	End to End
SVC	Switched Virtual Circuit
PVP	Permanent Virtual Path
PSNP	Poisson Shot Noise Process
PPBP	Poisson Pareto Burst Process
DP	Dynamic Programming
NDP	Neuro-Dynamic Programming
IETF	Internet Engineering Task Force
DSCP	Diffserv Code Point
BA	Behavior Aggregate
DS	Diffserv Domain
PHB	Per Hop Behavior
SLA	Service Level Agreement
TCA	Traffic Conditioning Agreement
VPI	Virtual Path Identifier
VCI	Virtual Channel Identifier
PSFFA	Pointwise Stationary Fluid Flow Approximation

BB	Bandwidth Broker
MDP	Markov Decision Process
RVI	Relative Value Iteration
A-RVI	Asynchronous Relative Value Iteration
A-RVI-FA	Asynchronous Relative Value Iteration with Function Approximation
TD	Temporal Difference
GA	Gosavi Algorithm
SA	State Aggregation
FA	Function Approximation
BG	Bandwidth Gain
SR	Signaling Rate
DCM	Dynamic Capacity Management

Chapter 1

Introduction

1.1 Motivation

Dynamic capacity adjustment refers to the process of dynamically changing the capacity reservation (bandwidth) of a virtual-path (VP) via signaling in the network domain. This process depends heavily on some certain criteria including instantaneous traffic load for the VP, current capacity reservation, hour of day or day of week and Quality of Service (QoS) parameters (e.g., signaling constraints).

A VP or a pseudo-wire stands for a generic path carrying aggregate traffic between two network end points. The route of the VP is fixed and the capacity allocated to it can be resized on-line dynamically (without a need for tearing it down and reestablishing it with a new capacity). With this generic definition, multiple networking technologies can be accommodated; a virtual path may be an MPLS-TE (MultiProtocol Label Switching - Traffic Engineering) LSP (Label Switched Path) [1], an ATM (Asynchronous Transfer Mode) VP [2], or a single aggregate RSVP (Resource ReserVation Protocol) reservation [3]. The end points of the virtual path will then be LSRs (Label Switch Router), ATM switches, or RSVP-capable routers, respectively.

At the first stage, we are motivated by "Voice Over Packet (VoP)" networks

(see Figure 1.1) where individual voice calls are aggregated into a VP in the packet-based network. VoP networks can be easily simulated and for the sake of simplicity, a good start point for testing our solution methodologies since all the voice calls require same amount of bandwidth in the network domain. At the edge of the packet network, there are the voice over packet gateways which are interconnected to each other using VPs. The packet network may be an MPLS, an ATM, or a pure IP network supporting dynamic aggregate reservations. In this scenario, end to end reservation requests that are initiated by PSTN (Public Switched Telephone Network) voice calls and which are destined to a particular voice over packet gateway arrive at the aggregator gateway. These reservations are then aggregated into a single dynamic reservation through the packet network. The destination gateway then deaggregates these reservations and forwards the requests back to the PSTN. Capacity update decision epochs are assumed to be the instants of either a call arrival or a call departure.

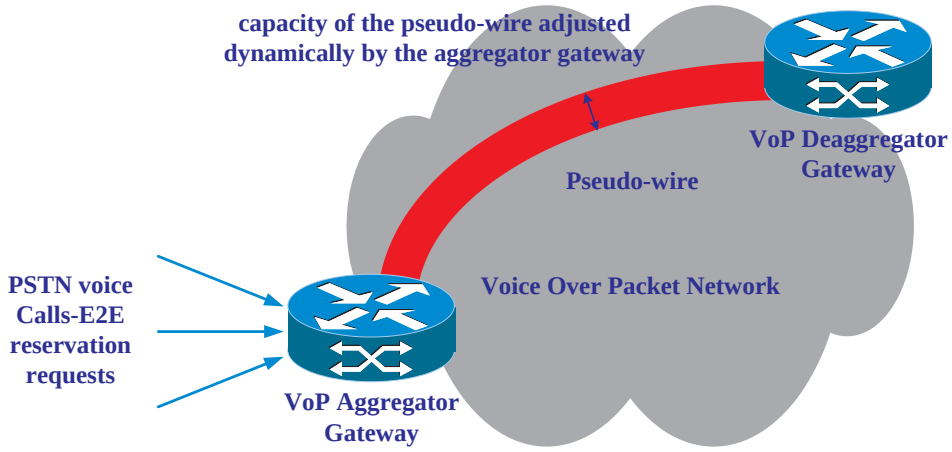

Figure 1.1: E2E (End-to-End) reservations due to PSTN voice calls are aggregated into one single reservation through the voice over packet network

There are two important issues in the dynamic capacity adjustment problem: utilization of bandwidth resources and signaling traffic in the network. If we adjust the bandwidth of the VP too frequently, under-utilization of bandwidth resources will decrease but this causes a huge amount of signaling traffic in the network which will be inefficient for the networks where traffic intensity changes

rapidly. On the other hand if we fix the bandwidth of the VP to a constant value and not change it over a long time period, signaling traffic in the network will decrease with the cost of increasing bandwidth under-utilization. From this point of view two different capacity adjustment approaches will be introduced: SVC (Switched Virtual Circuit) and PVP (Permanent Virtual Path) approaches. For optimal usage of the bandwidth, the capacity allocated to the VP should ideally track the actual aggregate traffic but this policy requires a substantial amount of signaling rates and it would not scale to large networks with rapidly changing traffic. For example, consider two VoP gateways interconnected to each other using a VP. Calls from the PSTN are admitted into the VP only when there is enough bandwidth and once admitted, traffic is packetized and forwarded from one gateway to the other in which it will be depacketized and forwarded back to the PSTN. Every time a new voice call arrives or an existing call terminates, the capacity of the VP may be adjusted for optimal use of resources. This approach will be referred to as the SVC approach throughout this thesis since the messaging and signaling requirements of this approach will be very similar to the case where each voice call uses its own SVC as in SVC-based ATM networks. The best way for reducing the signaling traffic in the network is through allocating capacity for the highest load over a long time window (e.g., 24-hour period). This approach would not suffer from signaling and message processing requirements since each capacity update would take place only once in a very long time window. Again motivated by ATM networks, this approach will be called the PVP approach. However, the downside of the PVP approach is that the capacity may be vastly under-utilized when the load is significantly lower than the allocated capacity (peak load). In this case, this idle capacity would not be available to other VP's that actually need it and this would lead to inefficient usage of resources. Figure 1.2 shows a snapshot of the behaviors of these approaches. In this figure the solid line shows the variation of the number of active calls in the system with respect to time. As mentioned above, the SVC policy tracks this signal ideally that means that the variation of the bandwidth assigned to the VP with respect to time is the same signal with the solid line. Dashed line shows the PVP bandwidth usage that is the bandwidth assigned to the VP is selected same as the peak load in the system. Dotted line gives an idea of the bandwidth usage scheme that we

are trying to find in this thesis since it represents a scheme that is in the middle of the SVC and the PVP approaches. Added to this, the optimal bandwidth usage scheme will depend on the relation between signaling and bandwidth usage efficiency.

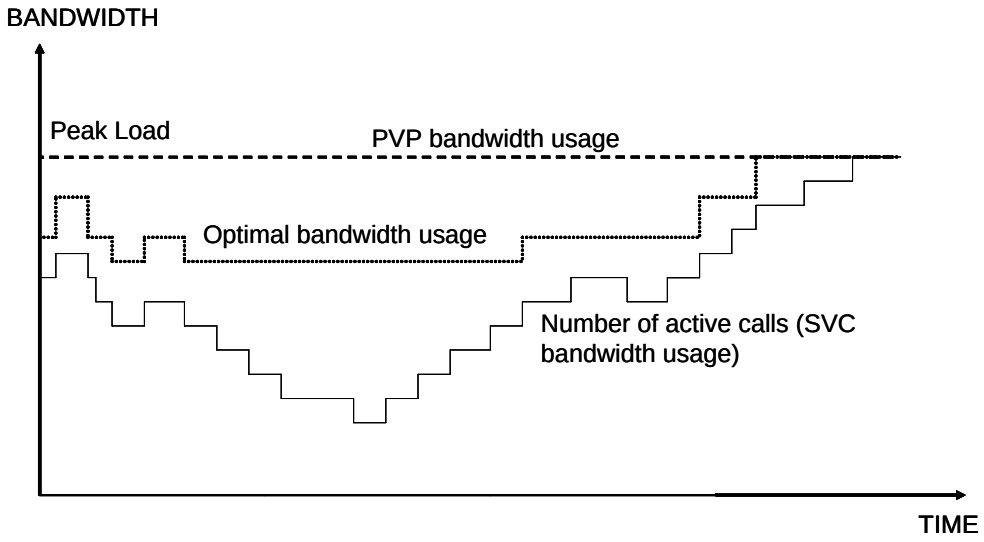

Figure 1.2: Sample figure depicting the behavior of SVC and PVP

For modelling the problem, the semi-Markov decision process is used and in order to relate the signaling and bandwidth utilization, two different formulations are introduced. In the first formulation, we assign cost parameters for bandwidth usage per unit time (denoted by b) and for every signaling required for a new capacity update (denoted by S) and for a wide range of S/b ratio our solution methodology will be applied. However the drawback of this formulation is the need for a mechanism for tuning the cost parameters to achieve an optimal bandwidth usage under a specific signaling constraint that occurs due to the practical limit on the number of capacity updates per unit time for a VP. For example, let us assume that the network nodes in the aggregation region can handle at most N capacity update requests per hour, which is assumed to be the scalability requirement. Assuming that on the average there are I output interfaces on every node and L VP's established on every such interface, an individual VP may be resized on the average $N/(IL)$ times in every hour. With typical values

of $N = 36000$ (10 capacity updates per second for an individual network node), I=16, and L=100, one can afford adjusting the capacity of each VP 22.5 times in an hour. In order to cope with this situation a novel formulation that has a goal of minimizing the idle capacity between the allocated capacity and the actual bandwidth requirement over time while satisfying the scalability requirement(e.g., by resizing the capacity of the VP less than 22.5 times per hour) will be used . A version of the leaky bucket counter is used to regulate the number of the capacity updates per unit time with the cost of adding a new dimension to the state space in our first formulation. Such counters have successfully been used for usage parameter control in ATM networks [2] and traffic conditioning at the boundary of a diffserv (Differentiated Services architecture) domain [4].

Solution techniques proposed in this thesis are more general and are amenable to dynamic capacity adjustment for non-voice scenarios as well. In the second stage, we are motivated by general (non-voice) traffic types (e.g., data, video) and we implemented a flow-based Internet traffic using the traffic modelling techniques given in [5] and [6]. In these techniques, traffic flowing through the VP is assumed to be the superposition of flows that are described in [5] by the use of PSNP (Poisson Shot Noise Process) and in [6] PPBP (Poisson Pareto Burst Process). Both event-driven and time-driven control methods are implemented and signaling rate is again regulated with the use of the leaky bucket counter. In event-driven control, the decision epochs for the capacity update process are taken to be the arrival or departure instants of the flows (assuming that the aggregator end point is able to detect these instants). In time-driven control, decisions are made whenever a pre-specified time interval (denoted by T) expires and this control strategy is more applicable than the event-driven one since it can be difficult to detect the instants of flow arrival or departure. Decision epochs are pre-specified time instants and so there is a need for a buffering mechanism that will be implemented by the aggregator gateway. For different values of traffic rate and T, our methods are applied for both of the control strategies.

Solution techniques that we proposed can be classified into two groups namely: dynamic programming (DP) methods (see [7], [8]) and neuro-dynamic programming (NDP) (or reinforcement learning (RL)) methods (see [9], [10], [11]). These

methods were applied to some existing networking problems successfully and the list below summarizes some of these works:

- In [12], the problem of call admission control and routing in an integrated services environment with several classes of calls with different service requirements are considered. The problem of maximizing the average number of admitted calls in the system per unit time is solved by NDP methods.
- In [13], the problem of call admission control in integrated services environment is studied for the single link case. DP methods are applied and when the problem size gets bigger, NDP methods that are more scalable than their DP counterparts are applied.
- In [14], the problem of dynamic channel allocation for cellular telephone systems is considered. The problem is formulated as a DP problem and a reinforcement learning solution is applied in order to maximize service and they show that the results perform better than the existing heuristics.
- In [15], the problem of call admission control and routing in multimedia networks are considered under QoS constraints. Problem is formulated by a semi-Markov decision process and an NDP algorithm is used and they showed that NDP provides better results than the alternative heuristics.
- In [16], dynamic routing and wavelength assignment problem in optical networks is studied. Assuming memoryless inter-arrival and holding times for calls, the problem is modeled as a Markov decision problem and NDP techniques are applied to problems where DP algorithms become intractable.
- In [17], the dynamic link sharing problem is solved under signaling constraints using NDP methods and it is shown that NDP solutions are scalable and perform better then the existing heuristic solutions.

In the next section, several QoS architectures that are proposed by the IETF (Internet Engineering Task Force) for IP networks will briefly be reviewed and how they relate to dynamic capacity adjustment will then be presented. After that

related work about dynamic capacity adjustment problem and different solution techniques will be reviewed.

1.2 QoS Architectures

1.2.1 Integrated Services

The integrated services architecture defines a set of extensions to the traditional best effort model of the Internet so as to provide end-to-end QoS commitments to certain applications with quantitative performance requirements [18], [19]. An explicit setup mechanism like RSVP will be used in the integrated services architecture to convey information to IP routers so that they can provide requested services to flows that request them [20]. Upon receiving per-flow resource requirements through RSVP, the routers apply admission control to signaled requests. The routers also employ traffic control mechanisms to ensure that each admitted flow receives the requested service irrespective of other flows. These mechanisms include the maintenance of per-flow classification and scheduling states. One of the reasons that have impeded the wide-scale deployment of integrated services with RSVP is the excessive cost of per-flow state and per-flow processing that are required for integrated services.

The integrated services architecture is similar to the ATM SVC architecture in which ATM signaling is used to route a single call over an SVC that provides the QoS commitments of the associated call. The fundamental difference between the two architectures is that the former typically uses the traditional hop-by-hop IP routing paradigm whereas the latter uses the more sophisticated QoS source routing paradigm.

1.2.2 Differentiated Services

In contrast with the per-flow nature of integrated services, differentiated services (diffserv) networks classify packets into one of a small number of aggregated flows or "classes" based on the Diffserv Codepoint (DSCP) in the packet's IP header [21], [22]. This is known as Behavior Aggregate (BA) classification. At each diffserv router in a Diffserv Domain (DS domain), packets receive a Per Hop Behavior (PHB), which is dictated by the DSCP. Since diffserv is void of per-flow state and per-flow processing, it is generally known to scale well to large core networks. Differentiated services are extended across a DS domain boundary by establishing a Service Level Agreement (SLA) between an upstream network and a downstream DS domain. Traffic classification and conditioning functions (metering, shaping, policing, and remarking) are performed at this boundary to ensure that traffic entering the DS domain conforms to the rules specified in the Traffic Conditioning Agreement (TCA) which is derived from the SLA.

1.2.3 Aggregation of RSVP Reservations

In the integrated services architecture, each E2E reservation requires a significant amount of message exchange, computation, and memory resources in each router along the way. Reducing this burden to a more manageable level via the aggregation of E2E reservations into one single aggregate reservation is addressed by the IETF [3]. Although aggregation reduces the level of isolation between individual flows belonging to the aggregate, there is evidence that it may potentially have a positive impact on delay distributions if used properly [23] and aggregation is required for scalability purposes.

In the aggregation of E2E reservations, we have an aggregator router, an aggregation region, and a deaggregator. Aggregation is based on hiding the E2E RSVP messages from RSVP-capable routers inside the aggregation region. To achieve this, the IP protocol number in the E2E reservation's Path, PathTear, and ResvConf messages is changed by the aggregator router from RSVP (46) to RSVP-E2E-IGNORE (134) upon entering the aggregation region, and restored

to RSVP at the deaggregator point. Such messages are treated as normal IP datagrams inside the aggregation region and no state is stored. Aggregate Path messages are sent from the aggregator to the deaggregator using RSVP's normal IP protocol number. Aggregate RESV messages are then sent back from the deaggregator to the aggregator via which an aggregate reservation with some suitable capacity will be established between the aggregator and the deaggregator to carry the E2E flows that share the reservation. Such establishment of a smaller number of aggregate reservations on behalf of a larger number of E2E flows leads to a significant reduction in the amount of state to be stored and the amount of signaling messages exchanged in the aggregation region.

One fundamental question to answer related to aggregate reservations is on sizing the reservation for the aggregate. A variety of options exist for determining the capacity of the aggregate reservation, which presents a tradeoff between optimality and scalability. On one end (i.e., SVC approach), each time an underlying E2E reservation changes, the size of the reservation is changed accordingly but one advantage of aggregation, namely the reduction of message processing cost, is lost. On the other end (i.e., PVP approach), in anticipation of the worst-case token bucket parameters of individual E2E flows, a semipermanent reservation is made. Depending on the actual pattern of E2E reservation requests, the PVP approach, despite its simplicity, may lead to a significant waste of bandwidth. Therefore, a policy is required which maintains the amount of bandwidth required on a given aggregate reservation by taking account of the sum of the bandwidths of its underlying E2E reservations, while endeavoring to change it infrequently. If the traffic trend analysis suggests a significant probability that in the next interval of time the current aggregate reservation will be exhausted, then the aggregator router will have to predict the necessary bandwidth and request it by an aggregate Path message. Or similarly, if the traffic analysis suggests that the reserved amount will not be used efficiently by the future E2E reservations, some suitable portion of the aggregate reservation may be released. We call such a scheme a dynamic capacity management scheme.

Classification of the aggregate traffic is another issue that remains to be solved. IETF proposes that the aggregate traffic requiring a reservation may all be marked

with a certain DSCP and the routers in the aggregation region will recognize the aggregate through this DSCP. This solves the traffic classification problem in a scalable manner.

Aggregation of RSVP reservations in IP networks is very similar in concept to the Virtual Path in ATM networks. In this framework, several ATM virtual circuits can be tunneled into one single ATM VP for manageability and scalability purposes. A Virtual Path Identifier (VPI) in the ATM cell header is used to classify the aggregate in the aggregation region (VP switches) and the Virtual Channel Identifier (VCI) is used for aggregation/deaggregation purposes. A VP can be resized through signaling or management.

1.3 Related Work

There are several other techniques proposed in the literature to solve the dynamic capacity adjustment problem. In the list below, some of these techniques are discussed briefly.

- In [24], the capacity of the VP is changed at regular intervals based on the QoS measured in the previous interval. A heuristic multiplicative increase multiplicative decrease algorithm in case of stationary bandwidth demand gives the amount of change. If the bandwidth demand exhibits a cyclic variation pattern, Kalman filtering is used to extract the new capacity requirement.

- In [25], blocking rates for the VP are calculated using the Pointwise Stationary Fluid Flow Approximation (PSFFA) and capacity is updated based on these blocking rates. Their approach is mainly based on the principle that if the calculated blocking rate is much less than the desired blocking rate, then the capacity is decreased by a certain amount and it is increased otherwise.

- In [26], the problem of traffic estimation and resource allocation for bandwidth brokers (BB) is addressed. A BB is defined to be a resource manager for network providers and neighboring BB's communicate with each other for establishing inter-domain resource reservation agreements. In this study the same trade-off between signaling and resource (bandwidth) utilization is addressed. If the allocation follows the traffic demand very tightly this will lead to a huge amount of inter-BB signaling but the resources will be used efficiently. Also if large resources are allocated and the modifications are far spaced in time, signaling traffic will decrease but this time resource usage efficiency will decrease seriously. As a solution methodology, they used a new scheme using Kalman filtering for estimating the traffic and forecasting its capacity requirement based on measurement of the current usage. Their method allows an efficient resource utilization while decreasing the number of reservation modifications.

- In [27], same problem is addressed for ATM networks. Here the problem is to adjust the network resources to be allocated to VP (ATM virtual path) authorities in order to balance resource waste and connection setup load (signaling) in the network. The problem is modelled by accounting both for bandwidth utilization and for signaling constraints. For a single link problem an approximate model is derived and the optimal rule is expressed as a closed form square-root allocation. The single link model is generalized and an algorithm based on the single link allocation is proposed.

- A practical example is the Cisco's *MPLS AutoBandwidth Allocator* (see [28]). This allocator automatically adjusts the bandwidth size of an MPLS-TE tunnel based on the traffic flowing through the tunnel. In Figure 1.3, an example of this process can be seen. Allocator monitors the largest X (e.g., 5 minutes) average of the traffic flow over a large configurable interval Y (e.g., 24 hours) and then readjusts the bandwidth upon the largest average output rate for the next Y interval. The downside of this approach is that whenever the traffic intensity is lower or higher than the adjusted value, system will suffer from resource under-utilization and service blockage respectively.

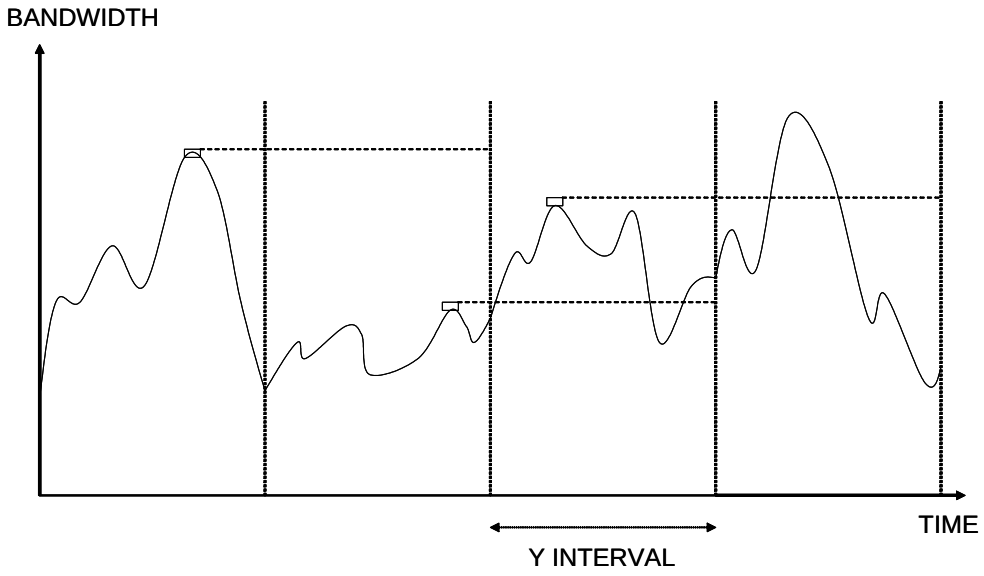

Figure 1.3: Sample figure depicting the behavior of Cisco's AutoBandwidth Allocator

1.4 Organization of the Thesis

In Chapter 2, semi-Markov decision process that we used will be described, DP and NDP algorithms will be given in a detailed way and a comparison between these algorithms will be given also. Chapter 3 presents the two formulations (formulation with cost parameters and formulation under signaling constraints) for VoP networks. Solution methods and numerical results will be given. Chapter 4 is devoted to the Internet flow-based traffic modelling, and solution methods with numerical results will be given for two different control strategies (time-driven and event-driven). Finally, Chapter 5 will conclude this thesis.

Chapter 2

Dynamic and Neuro-Dynamic Programming

Firstly, the notion of *Markov Decision Processes* (MDP) that is used as a framework for our formulations will be described. Based on this framework, solution techniques and algorithms that we used will be given in two major categories (DP and NDP). Also at the end of the chapter, a detailed comparison between these two categories will be given.

2.1 Markov Decision Processes

MDP is used to describe the controller-system interactions that hold the *Markov property* for system state transitions. It means that the probability distribution over the next state depends only on the current state and current action of the controller. Figure 2.1 shows a sample state-transition diagram for an MDP. As it is seen in the figure, different actions may lead the process to different states and for a certain action (e.g., a1), there may be different successor states with given transition probabilities.

MDP's are defined by the following elements:

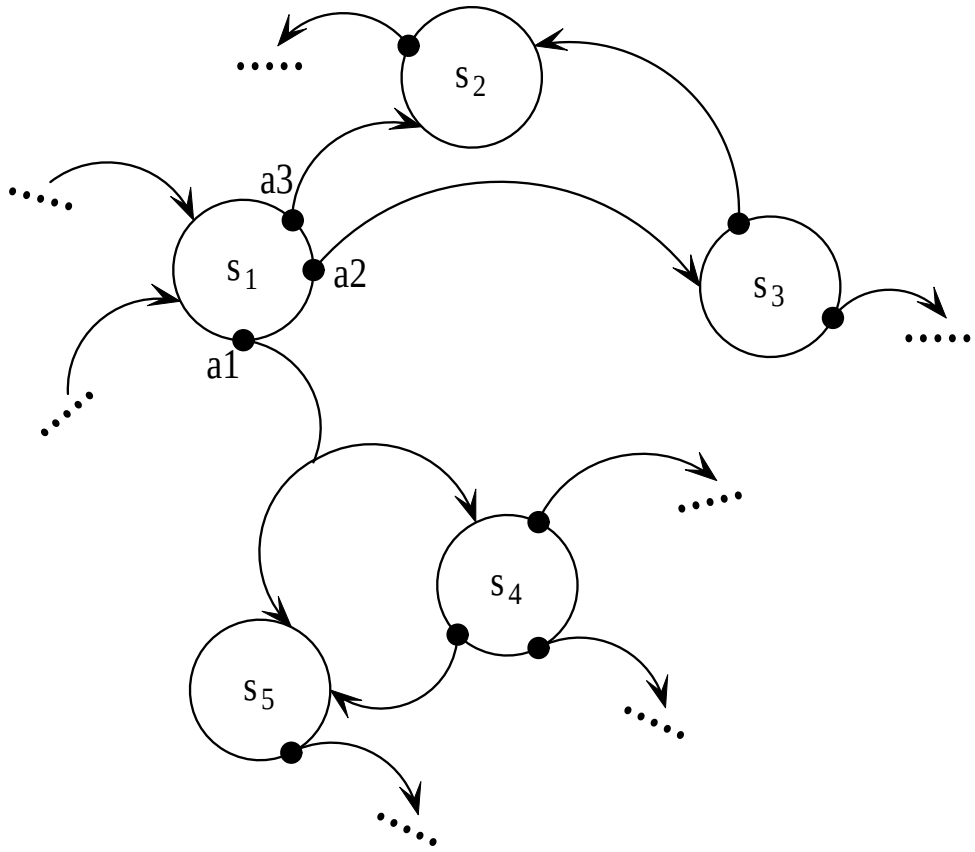

Figure 2.1: Sample state-transition diagram for a Markov Decision Process. Circles denote the states and black dots (e.g., a1, a2, a3) represent the actions that the controller can take.

State Space State space of the system will be denoted by **S** and consists of a *finite* set of elements $\{s_1, s_2, s_3, s_4, ..., s_N\}$.

Action Space For each state, a finite set of actions is defined, $A(s) = \{a_1^s, a_2^s, a_3^s, a_4^s, ..., a_M^s\}$.

State Transition Probabilities At a specific state (denoted by s), for each action that is selected by the controller (denoted by a) the time-homogeneous state transition probabilities are defined for each successor state (denoted by s'), $P_a(s, s')$.

State Transition Time Discrete time MDP's have constant state transition

times (denoted by 1 unit), continuous time MDP's (also known as semi-Markov Decision Processes) generally have state transition times that are continuous-time random variables. Specifically we will focus on semi-Markov Decision Processes (SMDP) since our problem formulations have continuous time nature.

Immediate Cost (Reward) Function For each state transition an immediate reward or cost that is a function of the current state, action, next state and state transition time (denoted by t) is incurred. This one step cost (reward) can be denoted by $c_s(a, s', t)$ $(r_s(a, s', t))$.

A stationary (time-invariant) *policy* $(\pi(\cdot))$ is defined to be the rule that assigns an action value for each state in **S**. Policies may be deterministic or randomized. Randomized policies define a probability distribution over the actions for each state in **S**. In this thesis only stationary and deterministic policies will be discussed.

The aim of dynamic and neuro-dynamic programming techniques that we will cover next, is minimization of the total discounted cost or minimization of the long-run average cost. Assuming there are at most H state transitions in the process, total discounted cost is given as:

$$\Sigma_H = c_0 + (\gamma \times c_1) + (\gamma^2 \times c_2) + (\gamma^3 \times c_3) + ... + (\gamma^H \times c_H) \tag{2.1}$$

where $c_{(\cdot)}$ stands for the incurred immediate cost for each iteration. Here $\gamma \in [0, 1]$ stands for the discount factor. A famous example for the discount factor is the interest rate in economic theory problems. Also the long run average cost is defined to be the time average of the undiscounted $(\gamma = 1)$ total cost.

If the number of state transitions (H) is finite, the problem is called a finite horizon optimization problem. Conversely if H is infinite or very large that means process continues over an infinite horizon, the problem is called an infinite horizon optimization problem. In this thesis, only infinite horizon, undiscounted problems will be considered and our optimization criteria is the minimization of the long run average undiscounted cost. Algorithms that will be presented next,

will lead us to optimal or sub-optimal, stationary and deterministic policies. In other words, these policies will rule how much bandwidth will be assigned to the virtual path for a given state of the network and VP.

2.2 Dynamic Programming

Dynamic programming algorithms include the value iteration, policy iteration and linear programming algorithms. A detailed analysis of these algorithms can be found in [7] and [8]. We use in this thesis one of the most scalable and time-efficient one, the so-called relative value iteration (RVI) algorithm. In order to apply the algorithm firstly a data transformation method for converting the semi-Markov decision problems to a discrete-time Markov decision model with the same state space is used. This transformation method and the RVI algorithm is given next [7].

2.2.1 Data Transformation

With this transformation method the expected immediate cost and state transition probabilities are converted as follows. Let $c_s(a)$ denote the expected immediate cost until next state when the current state is s and action a is chosen. Also let $\tau_s(a)$ denote the expected state transition time and $p_{s,s'}(a)$ denote the state transition probability from the initial state s to a next state s' when action a is chosen. Expected immediate costs ($\widetilde{c}_s(a)$) and one-step transition probabilities ($\widetilde{p}_{s,s'}(a)$) of the converted Markov decision model are given as the following [7]:

$$\widetilde{c}_s(a) = \frac{c_s(a)}{\tau_s(a)} \tag{2.2}$$

$$\widetilde{p}_{s,s'}(a) = \frac{\tau}{\tau_s(a)} p_{s,s'}(a), s' \neq s \tag{2.3}$$

$$\widetilde{p}_{s,s'}(a) = \frac{\tau}{\tau_s(a)} p_{s,s'}(a) + (1 - \frac{\tau}{\tau_s(a)}), s' = s \tag{2.4}$$

where τ should be chosen to satisfy

$$0 < \tau \leq min_{(s,a)} \tau_s(a) \tag{2.5}$$

After these transformations, the original semi-Markov decision process is converted to a auxiliary discrete time Markov decision process and these two systems are equivalent [7].

2.2.2 Relative Value Iteration Algorithm (RVI)

Let $V_n(s)$ denote the minimal total expected undiscounted immediate costs of the n state transitions starting with the initial state s. With these definitions and transformations the RVI algorithm is given as follows:

Step 0 Select $V_0(s)$, $\forall s \in \mathbf{S}$, from $0 \leq V_0(s) \leq \min_a \widetilde{c}_s(a)$ and $n := 1$.

Step 1a Compute the function $V_n(s)$, $\forall s \in \mathbf{S}$, from the equation

$$V_n(s) := \min_a \left[\widetilde{c}_s(a) + \sum_{s' \in \mathbf{S}} \widetilde{p}_{s,s'}(a) V_{n-1}(s') \right] \tag{2.6}$$

Step 1b Perform the following for all $s \in \mathbf{S}$ where s_0 is a pre-specified reference state:

$$V_n(s) := V_n(s) - V_n(s_0) \tag{2.7}$$

Step 2 Compute the following bounds

$$\begin{aligned} M_n &= \max_s (V_n(s) - V_{n-1}(s)), \\ m_n &= \min_s (V_n(s) - V_{n-1}(s)). \end{aligned} \tag{2.8}$$

If the following convergence condition is satisfied go to *Step 3*,

$$0 \leq (M_n - m_n) \leq \varepsilon m_n, \tag{2.9}$$

Else, let $n := n + 1$ and go to *Step 1a*.

Step 3 Find the optimal policy from the relation below and exit.

$$\pi^*(s) := arg \min_a \left[\widetilde{c}_s(a) + \sum_{s' \in \mathbf{S}} \widetilde{p}_{s,s'}(a) V_{n-1}(s') \right] \tag{2.10}$$

where ε is a pre-specified tolerance number. The condition 2.9 asserts that there is no more significant change in the value function of the states $\{V_n(\cdot)\}$. Also as we will mention, the optimal policy (denoted by $\pi^*(\cdot)$) is obtained by choosing the argument that minimizes the right hand side of (2.6). Also the role of *Step 1b* is to prevent the divergence of the values $(V_n(\cdot))$ when the number of iterations increases.

2.3 Neuro-Dynamic Programming

Neuro-dynamic programming or equivalently *reinforcement learning* methods are optimization techniques that seek for optimal or sub-optimal solutions using simulation-based methods. The optimal results of DP methods are approximated using stochastic approximation techniques, neural network based function approximations and state aggregation techniques. The NDP methods that we use in this thesis are given below in detail.

2.3.1 Asynchronous Relative Value Iteration Algorithm (A-RVI)

Asynchronous relative value iteration (A-RVI) is a simulation-based method that tries to approximate the optimal result of the RVI algorithm. In particular, we use the asynchronous version of RVI [11], [29], that uses simulation-based learning. Instead of updating all the values at a single iteration (batch updating) using the expected value of immediate cost and state transition probabilities, the real system (equivalently the auxiliary discrete-time Markov system that is obtained by the transformation mentioned before) is simulated and only the visited state's value is updated at a single iteration (single updating). This time $V(s)$ denotes the value of the state s and the optimal or sub-optimal policy is found by using these updated values that are learned throughout the process. Again with the same definitions for the terms $\widetilde{c}_s(a)$ and $\widetilde{p}_{s,s'}(a)$, the A-RVI algorithm is given as follows:

Step 0 Initialize $V(s) = 0, \forall s \in \mathbf{S}$, $n := 1$, average cost $\rho = 0$ and fix a reference state s_0, that $V(s_0) = 0$ for all iterations. Select a random initial state and start simulation.

Step 1 Choose the best possible action from the information gathered so far using the following local minimization problem:

$$\min_a \left[\widetilde{c}_s(a) + \sum_{s' \in \mathbf{S}} \widetilde{p}_{s,s'}(a) V(s') \right] \tag{2.11}$$

Step 2 Carry out the best or another random exploratory action. Observe the incurring immediate cost c_{inc} and next state s'. If best action is selected, perform the following updates:

$$\begin{aligned} V(s) &:= (1-\kappa_n)V(s) + \kappa_n(c_{inc} - \rho + V(s')) \\ \rho &:= (1-\kappa_n)\rho + \kappa_n(c_{inc} + V(s') - V(s)) \end{aligned}$$

Step 3 $n := n+1$, $s := s'$. Stop if $n = max_{steps}$, else go to *Step 1.*

where κ_n is the learning rate which is forced to die with increasing number of iterations. Maximum number of iterations that is denoted by max_{steps} is a problem dependent pre-specified number and must be larger for large-sized problems. Exploration is crucial for guaranteeing the convergence of NDP algorithms and will be discussed later in detail. The algorithm terminates with the stationary policy $\pi(\cdot)$ obtained in a same manner like RVI:

$$\pi(s) := arg\min_a \left[\widetilde{c}_s(a) + \sum_{s' \in \mathbf{S}} \widetilde{p}_{s,s'}(a) V_{n-1}(s') \right] \tag{2.12}$$

2.3.2 A-RVI with Value Function Approximation A-RVI-FA

Function approximation can be used for approximating the value function ($V(\cdot)$) defined over the state space $\mathbf{S}$. In this method we approximate the value of a

state by a linear architecture that is a function of the features of the state and in the algorithm, instead of the updating the value entry of the states in the tabular representation $V(\cdot)$, the weights of the feature components are updated. As an example, in our first formulation, the state variable s includes two different state values $s = (s_a, s_r)$, where s_a denotes the actual number of voice calls in the system and s_r denotes the reserved number of trunks (bandwidth required for a single voice communication) in the system. With this state definition, we choose our state feature vector as follows:

$$\overline{F(s)} = (1, s_a, s_r, s_a^2, s_r^2, s_a s_r) \tag{2.13}$$

Feature selection depends on problem formulation and selecting a large number of features may decrease the error of approximation but the number of iterations that is needed for convergence of the weight vector will increase. Similarly selecting a small number of features may be very efficient for convergence purposes but this time the approximation error may increase. Generally let $\overline{F(s)}$ denote the feature vector of the state s then the approximated value is expressed as follows:

$$\widetilde{V}(s, \overline{w}) = \overline{F(s)} \cdot \overline{w}^T \tag{2.14}$$

where $(\cdot)$ denotes the scalar vector product and $\overline{w}$ denotes the weight vector that is equal to $(w_1, w_2, w_3, ..., w_n)$ and n is the number of features. In this approximation scheme, the approximated value function depends on the weight parameters linearly. There are other approximation schemes in the literature that include nonlinear dependence (e.g., a feedforward neural network with a single hidden layer that includes sigmoidal functions), but we selected this linear architecture because the weight parameter update rule is easier to perform because of the linear dependence. Figure 2.2 shows a general block diagram depicting this linear, feature-based function approximation architecture.

Method for updating the weight vector is a stochastic approximation method which is called *temporal difference (TD) learning* (see [9] for details). In this method, the weight vector is updated based on a value called the *temporal difference* that accounts for the difference between the estimated and observed value in the simulation. The update scheme is a gradient based stochastic approximation method and the A-RVI algorithm with function approximation and weight

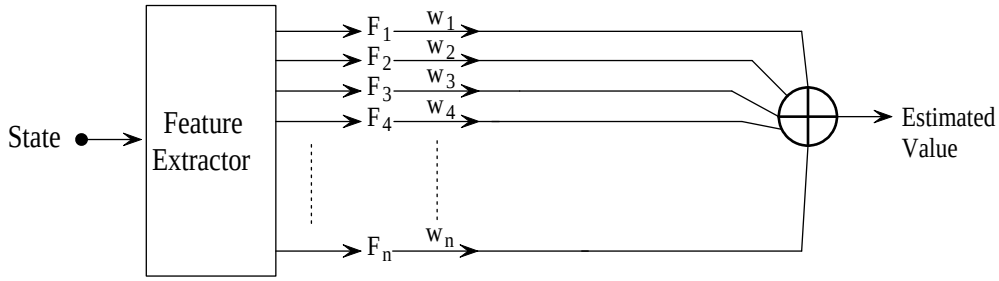

Figure 2.2: Block diagram of the linear architecture for value function approximation

(parameter) updating is given below:

Step 0 Initialize elements of the weight vector $\overline{w}_0$ to random numbers uniformly distributed in the interval $[0, 1]$, $n := 1$, average cost $\rho = 0$. Select a random initial state and start simulation.

Step 1 Choose the best possible action from the information gathered so far using the following local minimization problem:

$$\min_{a} \left[\widetilde{c}_s(a) + \sum_{s' \in \mathbf{S}} \widetilde{p}_{s,s'}(a) \widetilde{V}(s', \overline{w}_n) \right] \tag{2.15}$$

Step 2 Carry out the best or another random exploratory action. Observe the incurring immediate cost c_{inc} and next state s'. If best action is selected, calculate the TD from the relation below:

$$TD = c_{inc} - \rho + \widetilde{V}(s', \overline{w}_n) - \widetilde{V}(s, \overline{w}_n) \tag{2.16}$$

and perform the following updates:

$$\begin{aligned} \overline{w}_n &:= \overline{w}_n + (TD \times \kappa_n \times (\nabla_{\overline{w}} \widetilde{V}(s, \overline{w}) \mid_{\overline{w}=\overline{w}_n})) \\ \rho &:= (1 - \kappa_n)\rho + \kappa_n (c_{inc} + V(s', \overline{w}_n) - V(s, \overline{w}_n)) \end{aligned}$$

Step 3 $n := n + 1$, $s := s'$ and $\overline{w}_{n+1} = \overline{w}_n$. Stop if $n = max_{steps}$, else go to *Step 1*.

While calculating the *TD*, the term $\widetilde{V}(s, \overline{w}_n)$ accounts for the estimated value for the value $V(s)$ and the term $c_{inc} - \rho + \widetilde{V}(s', \overline{w}_n)$ accounts for the new estimate for the value and the weight vector is updated using this difference between the *estimated* and *observed* values in a gradient basis. Other parts of the algorithm is similar to the original A-RVI.

2.3.3 Gosavi Algorithm (GA)

DP and NDP algorithms given above require the full knowledge of the transition probabilities of the system. For problems with more complex formulations (e.g, our formulation with signaling constraints) the calculation and storage of these transition probabilities may be very hard. To cope with this situation we use a different NDP algorithm developed by Gosavi (see [30] for details). This algorithm is proposed for semi-Markov decision problems under the optimization criteria of minimization of the long-run average cost. Convergence proof and other details can be found in [30]. Algorithm is a model-free method that means the agent does not need to know the transition probability matrix of the underlying process and a new term (*Q-value*) is defined. *Q-function* or *Q-value* ($Q(s, a)$) is a variant of the value function ($V(s)$) and it is function of both the state variable s and action value a. With this new term, the Gosavi algorithm is as follows:

Step 0 Initialize $Q(s, a) = 0$, $\forall s \in \mathbf{S}$, $\forall a \in A(s)$, set $n := 1$, cumulative cost $c_{cum} = 0$, total time $T = 0$, average cost $\rho = 0$ and start simulation after selecting an initial starting state.

Step 1 Choose the best possible action by finding

$$arg \min_{a} Q(s, a) \tag{2.17}$$

Step 2 Carry out the best or another random exploratory action. Observe the incurring cost c_{inc}, state transition time τ_s and next state s'. Perform the following update

$$Q(s, a) := (1 - \kappa_n)Q(s, a) + \kappa_n(c_{inc} - \rho\tau_s + \min_{a} Q(s', a)) \tag{2.18}$$

If the best action is selected, perform the following updates:

$$\begin{aligned} c_{cum} &:= (1-\varsigma_n)c_{cum} + \varsigma_n c_{inc} \\ T &:= (1-\varsigma_n)T + \varsigma_n \tau_s \\ \rho &= \frac{c_{cum}}{T} \end{aligned}$$

Step 3 $n := n+1$, $s = s'$. Stop if $n = max_{steps}$, else go to ***Step 1***.

where κ_n and ς_n are the learning rates which is forced to die with increasing number of iterations. When the algorithm terminates, policy is evaluated from the relation:

$$\pi(s) = arg\min_a Q(s,a) \tag{2.19}$$

2.3.4 Exploration in NDP Algorithms

Exploration is crucial for guaranteeing the convergence of simulation-based NDP algorithms [9], [10], [11]. Especially it is required that all of the state-action pairs (s,a) are infinitely often tried for convergence of the algorithms. This is accomplished by *not* choosing the optimal action at every iteration. Instead a sub-optimal action is chosen according to the selected exploration strategy. The list below shows some of these existing exploration strategies and details can be found in [11].

- Boltzmann exploration
- Semi-uniform exploration
- Receny-based exploration
- Uncertainty exploration
- Darken-Chang-Moody exploration scheme (look in [31])

We propose a different exploration strategy and it will be denoted by *least visited search*. In this method, the action that leads the system to the least visited

state-action pair is chosen with some small probability ϵ. In our simulations ϵ is gradually decreased from some starting value (e.g., 0.5) to zero throughout the process in order to decrease the exploration when number of iterations increase.

2.3.5 State Aggregation in NDP Algorithms

When the problem size increases, function approximation and state aggregation are the proposed techniques for increasing scalability in NDP algorithms. Generally function approximation (FA) techniques may yield unstable behavior and a general FA architecture that works well for every MDP does not exist. Because of the instability and problem-dependent nature of FA techniques, state aggregation (SA) may be a more suitable technique for enhancing scalability. In SA, state space is partitioned into clusters and clusters are formed by joining the states that are in proximity in a certain sense. Figure 2.3 shows an example of SA. We used state aggregation with the Gosavi algorithm and our cluster forming technique and the experimental results will be given in the next chapter.

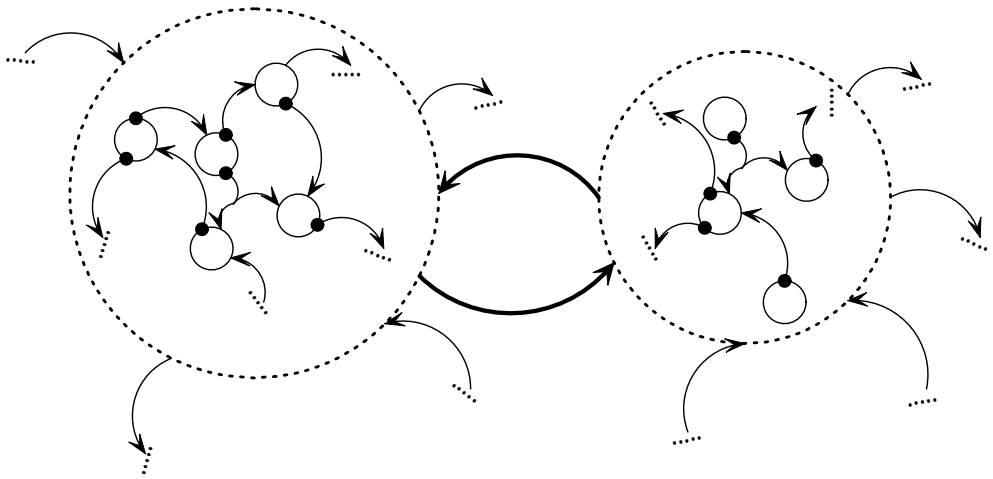

Figure 2.3: State aggregation example

2.4 DP versus NDP

DP and NDP algorithms differ mainly in the scalability issue. That means that when the problem size grows DP algorithms become intractable and the optimal solution can be approximated via NDP algorithms. Reasons behind this is that, firstly DP algorithms are *batch-update* algorithms. They swap all the state space at a single iteration and when the state space dimensionality increases the computational complexity increases and the algorithm will become slower and slower. On the contrary, NDP algorithms use a *single update*. At a single iteration only the visited state's value is updated. The other reason is that DP requires the full knowledge and the storage of the transition probabilities of the underlying MDP. Most of the NDP algorithms do not need the full knowledge of the transition probability matrix and this makes NDP very powerful because it can be applied to problems without a model. Generally speaking, DP represents *model-based* techniques. On the other hand, NDP represents *model-free* techniques that can be applied to many complex practical problems that are hard to model.

Chapter 3

Voice Traffic Modelling

In this chapter, we consider a VoP network as in Figure 1.1 that supports aggregate reservations. We assume end-to-end (E2E) reservation requests (voice calls) are identical and they arrive at the VoP aggregator gateway according to a homogeneous Poisson process with rate λ. We also assume exponentially distributed holding times for each E2E reservation with mean $1/\mu$. In this model, each individual reservation request is identical (i.e., one unit accounts for the bandwidth needed for the voice call communication), and we assume that there is an upper limit C_{max} units for the aggregate reservation. In other words C_{max} accounts for the maximum number of voice calls that the system can handle simultaneously. We decide to set C_{max} to the minimum capacity required to achieve a desired blocking probability that is denoted by p. C_{max} is derived using $p = EB(C_{max}, \lambda/\mu)$ where EB represents the Erlang's B formula. This ensures that the E2E reservation requests will be rejected when the instantaneous aggregate reservation is exactly C_{max} units and total rejection ratio cannot exceed p. In our simulation studies, we take $p = 0.01$. In this study, we do not consider the blocking due to unavailability of the bandwidth in the network. As we mentioned before, two important issues that affect the dynamic capacity adjustment policy is the bandwidth usage efficiency and signaling traffic in the network. We propose two different formulations for this problem. In the first formulation, we assign cost parameters for bandwidth usage and signaling and we find the optimal

policies for this cost parameters. Results are compared with the two approaches mentioned before, SVC and PVP. In the second formulation (which is more realistic than assigning cost parameters) we try to find the optimal policy under a signaling constraint. We used a variant of the leaky bucket counter for regulating the signaling rate. Next two sections are devoted to these two different formulations and numerical results.

3.1 Formulation with Cost Parameters (S, b)

In this formulation, we assign a cost for every capacity update (S) and a cost for allocated bandwidth unit per unit time (b). Our goal is to minimize the long run average cost per unit time as opposed to the total cumulative discounted cost, because our problem has no meaningful discount criteria. We denote the set of possible states in our model by $\mathbf{S}$:

$$\mathbf{S} = \{s | s = (s_a, s_r), 0 \leq s_a \leq C_{max}, \max(0, s_a - 1) \leq s_r \leq C_{max}\},$$

where s_a refers to the number of active calls using the VP just after an event which is defined either as a call arrival or a call departure. The notation s_r denotes the reserved bandwidth of the aggregate reservation (VP) before the event. For each $s = (s_a, s_r) \in \mathbf{S}$, one has a possible action of reserving $s'_r, s_a \leq s'_r \leq C_{max}$ units of bandwidth until the next event. The time until the next decision epoch (state transition time) is a random variable denoted by τ_s that depends only on s_a and its average value is given by:

$$\bar{\tau}_s = \frac{1}{\lambda + s_a \mu} \tag{3.1}$$

As described, at a decision epoch, the action s'_r (whether to update or not and if an update decision is made, how much allocation/deallocation of bandwidth will be performed) is chosen at state (s_a, s_r), then the time until, and the state at, the next decision epoch depend only on the present state (s_a, s_r) and the subsequently chosen action s'_r, and are thus independent of the past history of

the system that satisfies the Markov property. Upon the chosen action s'_r, the state will evolve to the next state $s' = (s'_a, s'_r)$ and s'_a will equal to either $(s_a + 1)$ or $(s_a - 1)$ according to whether the next event is a call arrival or departure. The probability of the next event being a call arrival or call departure is given as follows:

$$p(s'_a \mid s_a) = \begin{cases} \frac{\lambda}{\lambda + s_a\mu}, & \text{for } s'_a = s_a + 1, \\ \frac{s_a\mu}{\lambda + s_a\mu} & \text{for } s'_a = s_a - 1. \end{cases}$$

Two types of immediate costs are incurred when at state $s = (s_a, s_r)$ and action s'_r is chosen; first one is the cost of reserved bandwidth which is expressed as $b\tau_s s'_r$ where b is the cost parameter of reserved unit bandwidth per unit time. Secondly, since each reservation update requires message processing in the network elements, we also assume that a change in the reservation yields a fixed cost S. Immediate cost $(c_s(a))$ can be expressed mathematically as the following:

$$c_s(a) = \begin{cases} b\tau_s s'_r, & \text{for } s'_r = s_r, \\ b\tau_s s'_r + S & \text{for } s'_r \neq s_r. \end{cases}$$

This formulation fits very well to a semi-Markov decision model where the minimization of the long-run average cost is taken as the optimality criterion. We propose the RVI, A-RVI and A-RVI-FA for this problem based on [7], [11], and [29].

In the subsections next we will present the experimental results of our algorithms with respect to changing S/b ratio, $Cmax$ and arrival rate λ. Also a large-sized problem (where DP algorithm is intractable) is tested and we will present the results of the A-RVI for this case.

3.1.1 Varying S/b Ratio

The S/b ratio is changed from 0.1 to 1000 and the dynamic capacity adjustment results are presented in terms of average bandwidth gain (BG), signaling rate

Table 3.1: Results for the RVI, A-RVI and A-RVI-FA policies for varying S/b ratio.

	RVI			A-RVI			A-RVI-FA		
S/b	$BG\%$	SR	ρ	$BG\%$	SR	ρ	$BG\%$	SR	ρ
1000	16.93	6.26	15.03	0.00	0.00	16.00	0.00	0.00	16.00
750	19.59	8.12	14.56	0.00	0.00	16.00	0.00	0.00	16.00
500	22.59	10.87	13.89	0.00	0.00	16.00	0.00	0.00	16.00
250	28.29	20.60	12.90	20.46	10.41	13.45	29.10	34.68	13.75
200	30.68	26.37	12.56	24.81	14.98	12.86	29.34	35.24	13.26
150	31.91	31.00	12.19	29.21	24.46	12.35	40.91	128.40	14.80
100	33.96	39.11	11.65	31.63	32.74	11.85	44.32	324.07	17.91
50	36.50	57.36	10.96	37.13	68.00	11.00	44.71	351.92	13.73
25	40.36	112.26	10.32	40.74	128.60	10.37	44.71	351.92	11.29
10	42.56	172.57	9.67	43.93	279.59	9.75	44.71	351.92	9.82
5	44.71	351.92	9.34	44.53	334.32	9.34	44.71	351.92	9.34
1	44.71	351.92	8.95	44.66	347.23	8.95	44.71	351.92	8.95
0.1	44.71	351.92	8.86	44.66	347.23	8.86	44.71	351.92	8.86

(SR) which is equal to the number of capacity updates per hour and long run average cost normalized with the parameter (b) that is denoted by ρ. Bandwidth gain is the percentage of bandwidth that is preserved with respect to PVP average bandwidth usage which is equal to $Cmax$ units. The problem parameters are chosen as $\lambda = 0.0493$ calls/sec., $1/\mu = 180$ seconds, $C_{max} = 16$. Maximum number of iterations for A-RVI and A-RVI-FA is taken to be $max_{steps} = 10^7$. We run ten different 12 hour simulations for different values of S/b, and average of these simulations are reported. Results for the RVI, A-RVI, A-RVI-FA, SVC and PVP policies are given in the tables 3.1 and 3.2. Also Figure 3.1 shows a sample behavior of the policies found by RVI, A-RVI and A-RVI-FA with respect to increasing value of the ratio S/b. In the subplots, the x-axis shows the time in *seconds* and the y-axis shows the number of bandwidth units. Blue line shows the variation of actual number of voice calls in the system and the red line shows the corresponding number of bandwidth units assigned to the VP.

As we compare the results in tables 3.1 and 3.2 we see that, when the S/b ratio decreases the policies found by the methods converge to the SVC policy. Conversely, if this ratio is very high, solutions converge to the behavior of the PVP

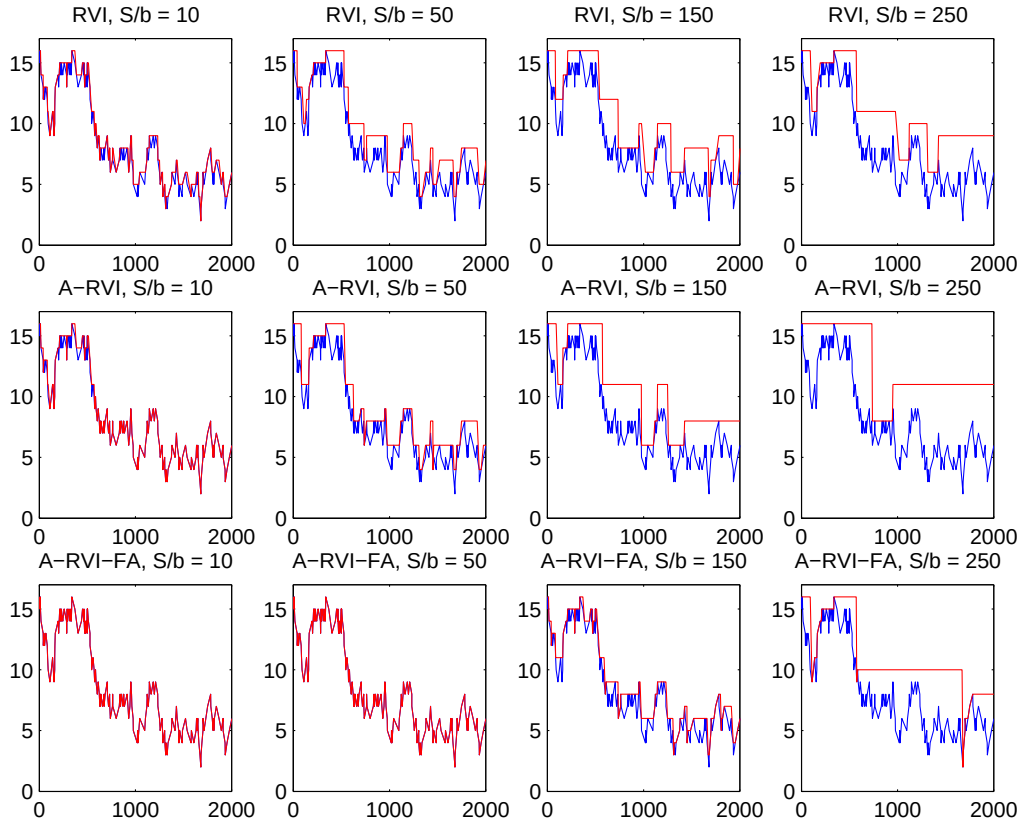

Figure 3.1: Snapshot of the policy behaviour of RVI, A-RVI and A-RVI-FA for different values of S/b

Table 3.2: Results for the SVC and PVP policies for varying S/b ratio.

	SVC			PVP		
S/b	$BG\%$	SR	ρ	$BG\%$	SR	ρ
1000	44.71	351.96	106.60	0.00	0.00	16.00
750	44.71	351.96	82.16	0.00	0.00	16.00
500	44.71	351.96	57.72	0.00	0.00	16.00
250	44.71	351.96	33.29	0.00	0.00	16.00
200	44.71	351.96	28.40	0.00	0.00	16.00
150	44.71	351.96	23.51	0.00	0.00	16.00
100	44.71	351.96	18.62	0.00	0.00	16.00
50	44.71	351.96	13.74	0.00	0.00	16.00
25	44.71	351.96	11.29	0.00	0.00	16.00
10	44.71	351.96	9.83	0.00	0.00	16.00
5	44.71	351.96	9.34	0.00	0.00	16.00
1	44.71	351.96	8.95	0.00	0.00	16.00
0.1	44.71	351.96	8.86	0.00	0.00	16.00

policy. This result is expected because the SVC and PVP are the two optimal approaches when the signaling cost is very low or very high, respectively. As we compare the average cost performance of RVI and A-RVI we see that they are almost equal for values of the ratio $S/b < 500$. They achieve the same average cost performance with different policies since for lots of the ratio values in $S/b < 500$, A-RVI signaling rate is smaller than the RVI signaling rate. Conversely A-RVI average bandwidth gain is usually smaller. A-RVI-FA performs poorly compared with A-RVI in the performance of average cost criteria. Also for the values $S/b \geq 500$, both A-RVI and A-RVI-FA policies are same with the PVP approach. A-RVI and RVI policies converge to the SVC approach for the values $S/b < 10$, instead A-RVI-FA converges to the SVC for the values $S/b < 100$ and this shows the unstable nature of the function approximation algorithm. Only for the values $100 < S/b < 500$, A-RVI-FA performs better with respect to SVC and PVP and for all the values it performs poorly with respect to RVI and A-RVI in average cost performance.

Figure 3.1 shows snapshot from the policy behaviors for different values of S/b. For small values of the ratio, reserved bandwdith covers the number of active calls ideally which is the behavior of the SVC approach. When S/b increases, the

Table 3.3: Results for the RVI and A-RVI policies for varying λ and C_{max} values.

		RVI			A-RVI		
λ	C_{max}	$BG\%$	SR	ρ	$BG\%$	SR	ρ
0.03	10	43.62	100.17	6.33	44.66	128.60	6.43
0.21	50	21.58	181.03	40.47	20.50	132.27	40.67
0.47	100	13.76	213.73	87.72	12.48	124.29	88.38

signaling rate decreases and they converge to the PVP behavior.

3.1.2 Varying λ

Systems with different arrival rate and C_{max} values are simulated. Again λ and $Cmax$ are related to each other via the relation $p = EB(C_{max}, \lambda/\mu)$. S/b ratio is selected to be 25 and $1/\mu = 180$ seconds. Again maximum number of iterations for A-RVI and A-RVI-FA are taken to be $max_{steps} = 10^7$. Tables 3.3, 3.4, 3.5 and 3.6 show the results for this study. When the arrival rate increases (so C_{max} increases) the problem size increases and this affects the scalability of our algorithms. Table 3.6 shows the number of iterations that is needed for convergence of the RVI. As the arrival rate increases, number of iterations increases and solution via RVI becomes more and more slower. Also when the arrival rate increases, the speed of the process increases. For this reason bandwidth gain decreases for larger arrival rates and in order to achieve a good bandwidth gain, we must use smaller values for S/b with increasing value of the arrival rate. Results for RVI and A-RVI are promising but A-RVI-FA results converge to the SVC policy behavior due to unstable nature of the algorithm. As a result we can say that, as arrival rate increases RVI and A-RVI results are better than the SVC and PVP results but there occurs an overall degradation of the bandwidth gain because of the increasing arrival rate of the process.

Table 3.4: Results for the A-RVI-FA policy for varying λ and C_{max} values.

		A-RVI-FA		
λ	C_{max}	$BG\%$	SR	ρ
0.03	10	47.11	208.63	6.74
0.21	50	24.92	1498.37	47.95
0.47	100	15.94	3384.00	106.81

Table 3.5: Results for the SVC and PVP policies for varying λ and C_{max} values.

		SVC			PVP		
λ	C_{max}	$BG\%$	SR	ρ	$BG\%$	SR	ρ
0.03	10	47.11	208.67	6.74	0.00	0.00	16.00
0.21	50	24.93	1499.32	47.95	0.00	0.00	16.00
0.47	100	16.38	3339.48	106.81	0.00	0.00	16.00

Table 3.6: Number of iterations needed for convergence of the RVI with changing λ and C_{max}.

λ	C_{max}	Number of Iterations
0.03	10	163
0.21	50	1710
0.47	100	2631

Table 3.7: Performance results of the A-RVI policy for the case C_{max}=300.

	$S/b = 100$	$S/b = 50$	$S/b = 20$
A-RVI average cost	272.20	262.00	212.83
SVC average cost	526.60	387.90	304.60
PVP average cost	300.00	300.00	300.00
A-RVI bandwidth gain %	9.33	13.00	15.33
A-RVI signaling rate	45.00	550.00	2418.00

3.1.3 A Larger Sized Problem: C_{max}=300

Table 3.7 shows the performance of A-RVI for a larger size problem where the RVI solution is numerically intractable. We take $C_{max} = 300$, λ= 1.5396 $calls/sec$ and $1/\mu = 180$ seconds. This table demonstrates that with a suitable choice of the ratio S/b, one can limit the frequency of capacity updates in a dynamic capacity adjusment scenario. Moreover, A-RVI consistently gives better results than both PVP and SVC in terms of the overall average cost and this shows that NDP algorithms can be applied to large problems with success.

3.2 A Disadvantage: Tuning the Cost Parameters

The disadvantage of the previous formulation is that there is no immediate mechanism to tune the cost parameters for a desired signaling rate (constraint). Moreover, cost parameters have no practical meaning from the perspective of a network administrator (agent). Cost of signaling in the network and bandwidth usage cost cannot be related with each other easily. So a revised formulation of the problem is needed. In this new formulation that will be presented in the next section, a new state variable is introduced. This new state variable stands for the value of the leaky bucket counter that is used for regulating the signaling rate. The aim of our revised formulation is a more practical one from the point of view of the network agent. In this formulation, our aim will be to find the optimal bandwidth

usage that leads us to the maximum bandwidth gain together with a desired rate of signaling. The details of the revised formulation and the working principle of the leaky bucket counter will be given in the following section.

3.3 Formulation with the Signaling Rate Constraint

In this new formulation, we introduce a desired signaling rate D (number of desired capacity updates per hour). Our goal will be to minimize the average reserved bandwidth usage subject to the constraint that the frequency of capacity updates will be smaller or equal to the desired rate D. A variant of the leaky bucket counter is used in this formulation. There will be no more cost parameters and only bandwidth usage will incur cost during the process. There is no signaling cost in the network, instead it is regulated by the counter.

A generic leaky bucket counter is a counter that is incremented by unity each time an event occurs and that is periodically decremented by a fixed value. We suggest to use a modified leaky bucket counter for the dynamic capacity adjusment problem to regulate the actual signaling rate to the desired value. Let $X, 0 \leq X \leq B_{max}$ be the value of the counter where B_{max} denotes the size of the counter. The working principle of our modified leaky bucket counter is given as follows:

When a new capacity update request occurs, then

a) If $X < B_{max} - 1$, then the bucket counter is incremented by one,

b) If $X = B_{max}$, then the capacity update request will be rejected,

c) If $X = B_{max} - 1$, then the new reserved capacity for the VP will be forced to be C_{max} which is the maximum allowable bandwidth that can be assigned to the VP and the counter will be incremented by one to B_{max}.

In the meantime, the counter is decremented with the desired rate. This means that buffer *leak* rate will be equal to the value D and bucket will be decremented by unity every $3600/D$ seconds. The difference between the modified counter introduced above and the generic leaky bucket counter is the operation under the condition c). The motivation behind the operation c) is that if the capacity of the VP was not set to C_{max}, then in the worst case scenario, the blocking probability would have exceeded the value p until the next epoch of decrementing the counter. Also this condition has a key role in regulating the actual signaling rate to the value D. Whenever the process enters the state $X \geq B_{max} - 1$, the action will be setting the bandwidth to C_{max} and by this choice the cost of this state will be the maximum since only bandwidth usage incurs cost in the formulation. This results that the resulting policy will learn *not* to enter this state and this can be handled only by regulating the signaling rate to be smaller than the bucket leak rate. But in the optimal case actual rate of the policy will be equal to the desired signaling rate because learning process will do its best to minimize bandwidth usage by achieving a signaling rate with the maximum value it can take (D). We also note that B_{max} is analogous to the maximum burst size in ATM networks and its role in this paper is to limit the number of successive capacity update requests. In our simulations, we fix $B_{max} = 10$ and leave a detailed study of the impact of B_{max} for future work.

Our re-defined state space is as follows:

$$\mathbf{S} = \{s | s = (s_a, s_r, s_b), 0 \leq s_b \leq B_{max}\},$$

where s_a and s_r are as defined before and s_b refers to the value of the leaky bucket counter. For each state (s_a, s_r, s_b) an action value s'_r satisfying $s_a \leq s'_r \leq C_{max}$ will be chosen. With addition of this new state variable, the transition probabilities of the model become harder to calculate and we used the *model-free* Gosavi algorithm to find the dynamic capacity adjustment scheme. Also for a big size problem we used a state aggregation technique. The experimental results are given in the following subsections.

3.3.1 Varying Desired Rate D

The problem parameters are chosen as $\lambda = 0.0493$ calls/sec., $1/\mu = 180$ seconds, $C_{max} = 16$. Maximum number of iterations for the Gosavi algorithm is taken to be $max_{steps} = 10^7$. Different desired rate D values are tested from the value 10 to 140 capacity updates per hour. Figure 3.2 shows the average bandwidth usage in terms of capacity units of our policy which is denoted by DCM (Dynamic Capacity Management). Results are obtained from 24 hour simulations. As it is seen from the figure when D increases, bandwidth usage converges to the SVC approach and the converse is true also. This result is expected since when the maximum allowed rate of signaling (D) increases, bandwidth gain will increase. Added to this, the actual rate of our policy that is observed through the simulations, ideally tracks the desired rate and stays within the 2% neighborhood of D irrespective of the value of D. Figure 3.3 shows one hour snapshot of the behavior of our policy for different values of D. Again the blue line shows the actual number of voice calls in the system and the red envelope shows the reserved bandwidth for the VP determined by our policy. It is seen clearly that when D increases, number of capacity updates increases while the average reserved bandwidth decreases.

3.3.2 Varying λ

The performance of our algorithm is tested for different values of arrival rate λ and increasing number of iterations. Number of iterations (max_{steps} in the Gosavi algorithm) here denotes the number of state transitions that takes place during the simulation for learning. Figure 3.4 shows the results for this part. With the increasing value of λ, the problem becomes harder to solve because of the growth in the state space of the underlying process. From the figure 3.4 it is seen that for a fixed value of D, when the arrival rate increases the average bandwidth gain with respect to the PVP approach decreases. This is expected since the larger the arrival rates, the smaller the variance of the number of ongoing calls at a given instance and the reduction in this variance causes the reduction of gain from the dynamic capacity adjustment scheme. When desired signaling grate D is allowed

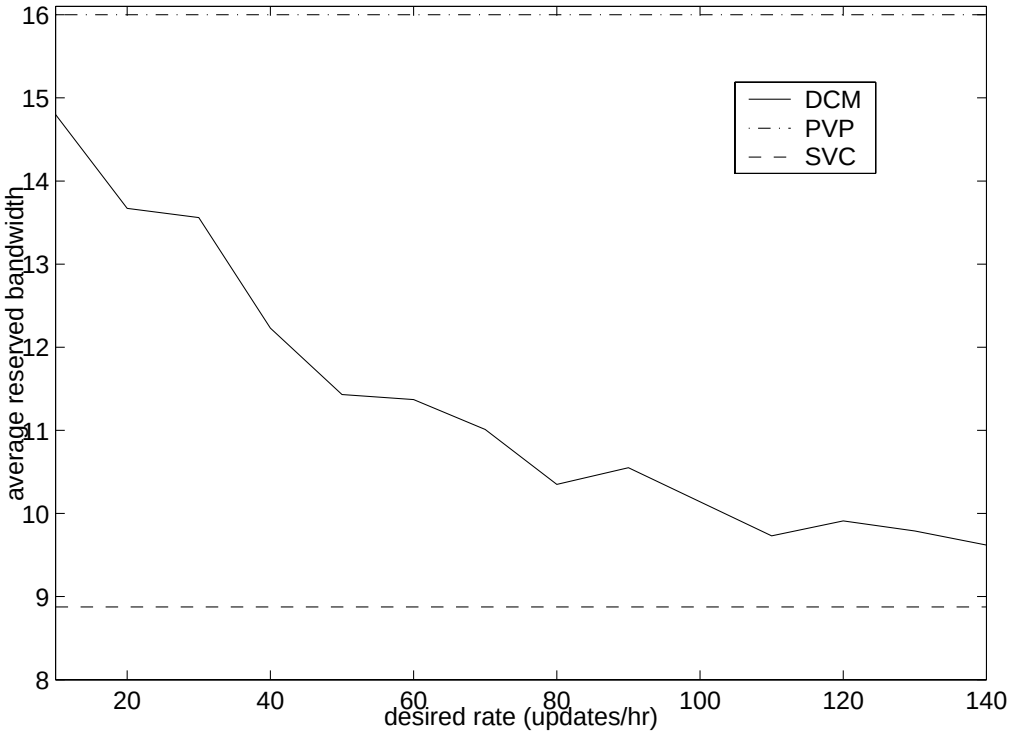

Figure 3.2: Average reserved bandwidth for the VP for different values of D

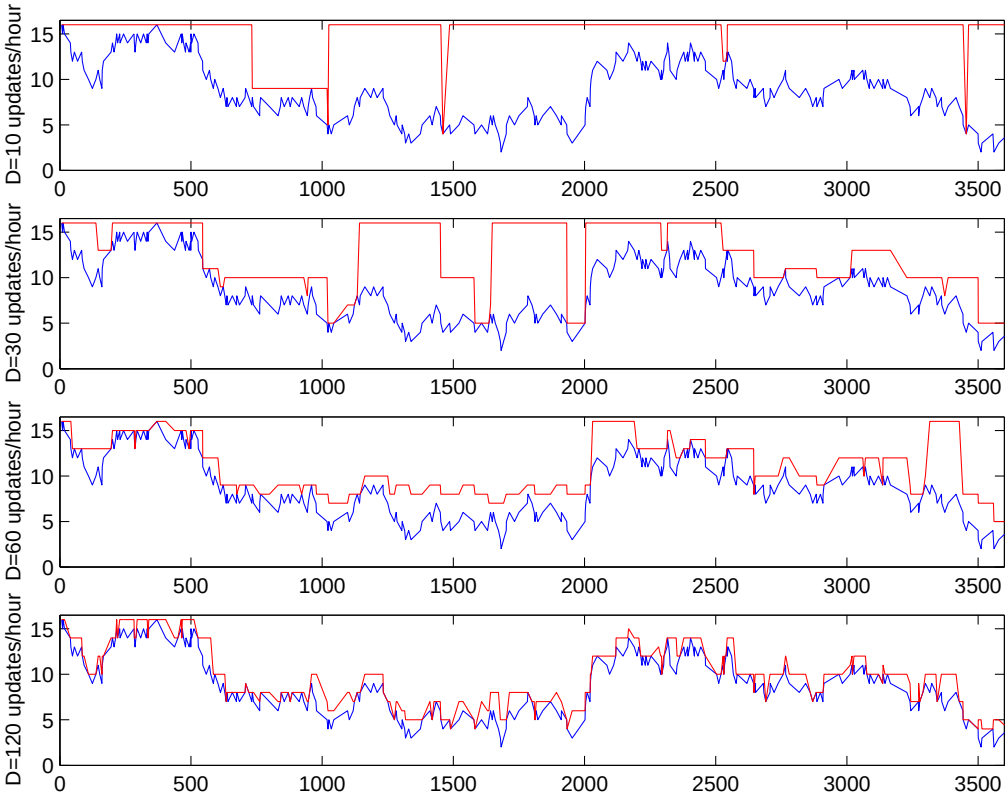

Figure 3.3: Snapshot of the policy behaviour for different values of D

to vary, increasing this value has a positive effect on both bandwidth gain and algorithm convergence times as seen from the Figure 3.4. Again, the actual rate of our policy that is observed through the simulations, ideally tracks the desired rate and stays within the 2% neighborhood of D irrespective of the value of D.

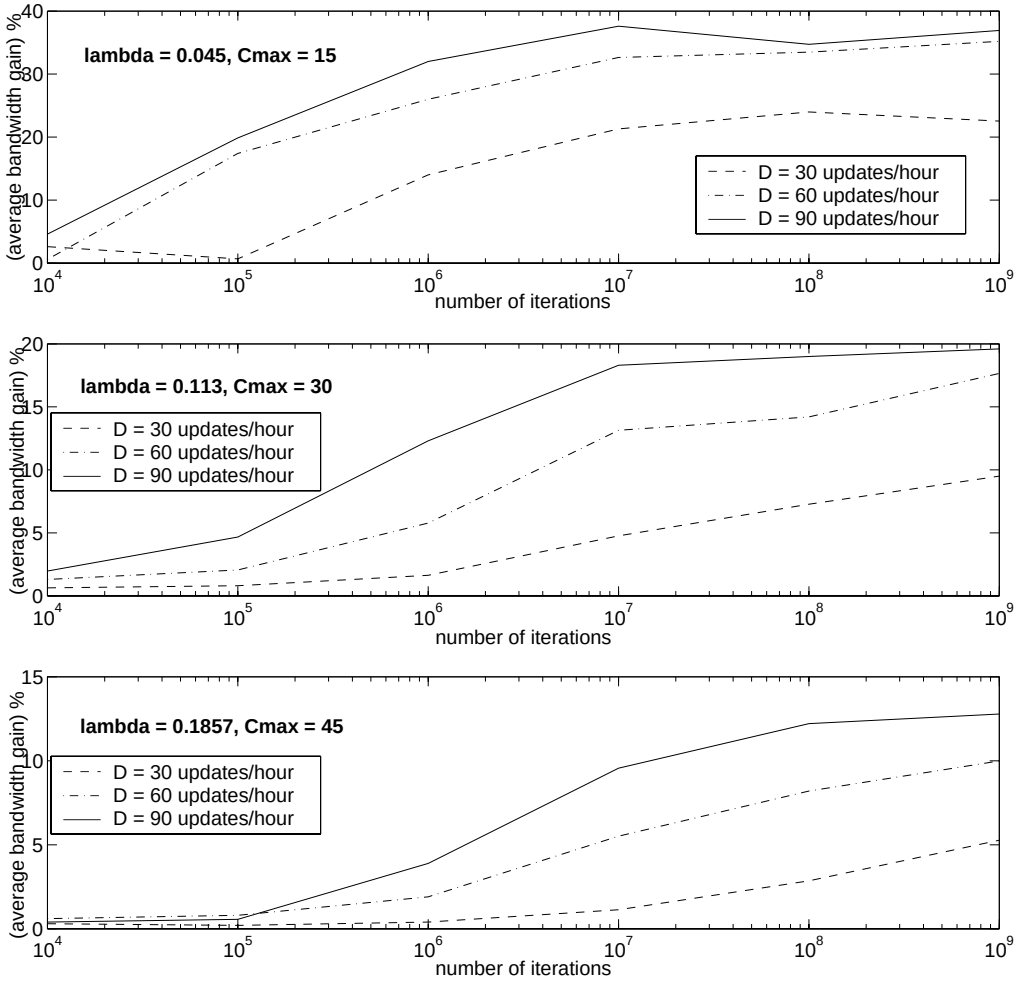

Figure 3.4: Average bandwidth gain for different values of λ and D

3.3.3 State Aggregation with Gosavi Algorithm

For the problems with larger state-space dimensionality, the number of iterations needed for good performance increases and our method becomes intractable as

seen from the results in Figure 3.4. One major reason for that is, in order for the learning algorithm to converge as we mentioned before, each state-action pair must be visited many times during the simulation (in theory infinitely many times). When the number of iterations during the learning is kept fixed, each state-action pair will be visited fewer times when the state space dimensionality increases and this has a negative effect on the performance of our method. To cope with this situation we used a state-aggregation (cluster forming) technique, namely the *uniform* state aggregation. To express it in detail, we aggregated the state variables uniformly in the first two dimensions. Figure 3.5 demonstrates this procedure and will help the reader to understand it.

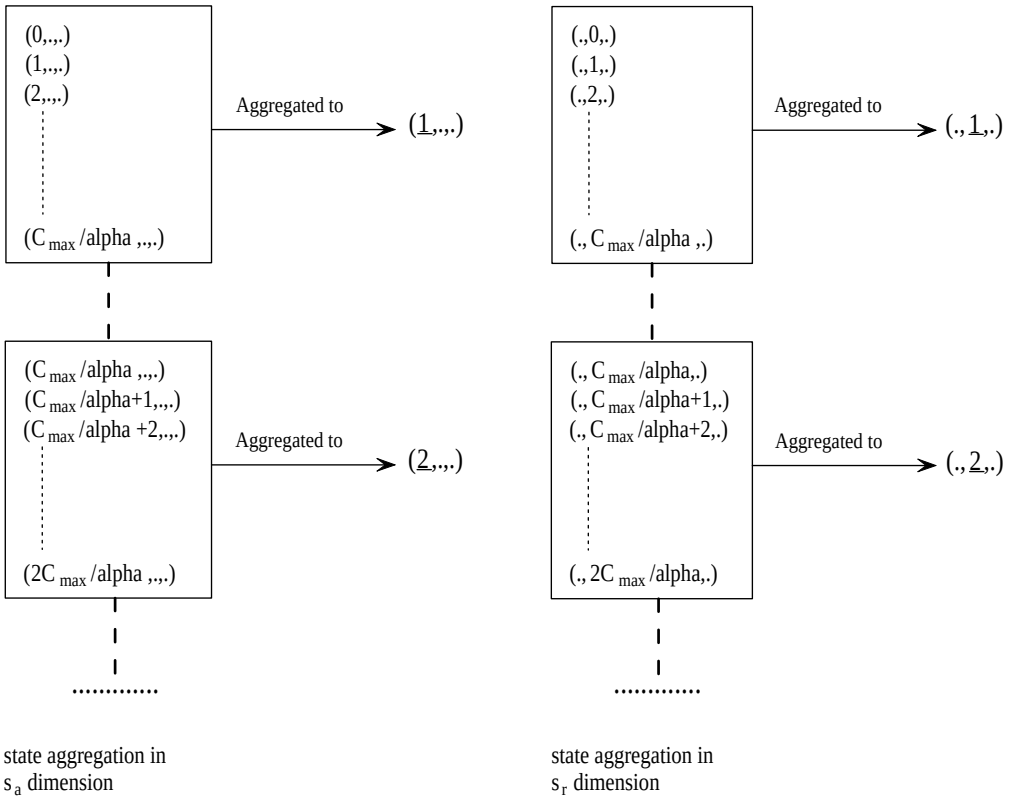

Figure 3.5: Uniform state aggregation in two dimensions

The original problem has a state space dimensionality of $(C^2+3C-2)(B_{max}+1)/2$ states where C is defined to be the value $C_{max}+1$. With the use of the cluster forming technique represented in Figure 3.5, our state space dimensionality will

reduce to the value $(\overline{C}^2+3\overline{C}-2)(B_{max}+1)/2$ where $\overline{C}$ is defined to be C/α and α is called the *aggregation factor* in this study. When the aggregation factor increases the state space will be aggregated more aggressively. The results obtained via state aggregation is shown in the Figure 3.6.

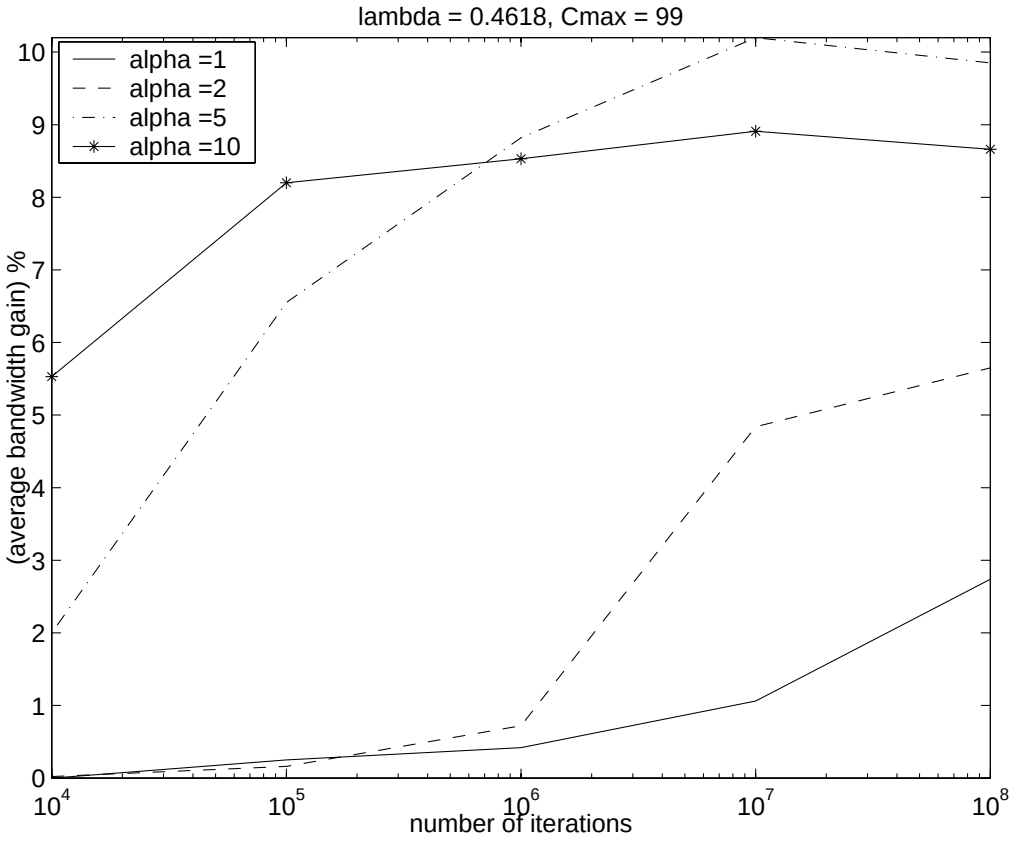

Figure 3.6: Average bandwidth gain for different levels of aggregation for a problem with $C_{max} = 99$

There are two main issues that must be considered during the state aggregation process. If the clusters are formed aggressively that means α in our approach increases and clusters cover large number of states, this leads to better performance in convergence but the information lost during the learning process increases. This trade-off can be seen easily from the results in Figure 3.6. For example for the case where number of iterations is equal to 10^8 steps, best strategy is to choose $\alpha = 5$, $\alpha = 10$ converges very fast but when the number of iterations increase best performance is obtained by a conservative strategy where $\alpha = 5$ and

$\alpha = 1$ denotes the original problem where there is no state aggregation is applied and it can be seen that even for 10^8 iterations it still does not converge to a good result. $\alpha = 2$ is better than the $\alpha = 1$ in the performance of bandwidth usage and convergence but in the overall case best strategy is selecting $\alpha = 5$.

Chapter 4

Flow-Based Internet Traffic Modelling

NDP methods can be applied to more complex traffic modelling structures. In this chapter, a flow-based Internet traffic modelling is used, based on the theory in [5] and [6]. Flow is used as a generic term and it can represent different traffic types including voice, data and video. Total traffic on the VP is assumed to be the superposition of individual flows and the total rate is computed using the flows' characteristics (arrival times, size, duration). Traffic flows are represented by *shots* whose duration and size is generated according to some distribution and arrival process of the flows is assumed to be a homogeneous Poisson process with rate λ. Different shot shapes (rectangular, triangular, sublinear, superlinear shapes and etc.) are presented in [5] for modelling the Internet traffic. In this study we used a general rectangular shot shape and Figure 4.1 shows it in detail.

The arrowheads show the arrival times of the flows that are generated by the Poisson process with rate λ. Two other important parameters are the duration (denoted by d) and size (equals to the area under the rectangular region of the shot and denoted by s) of the flow. The rate of the flow (r) is assumed to be constant and computed by the simple relation s/d. Key point is selecting the distributions for the duration and size.

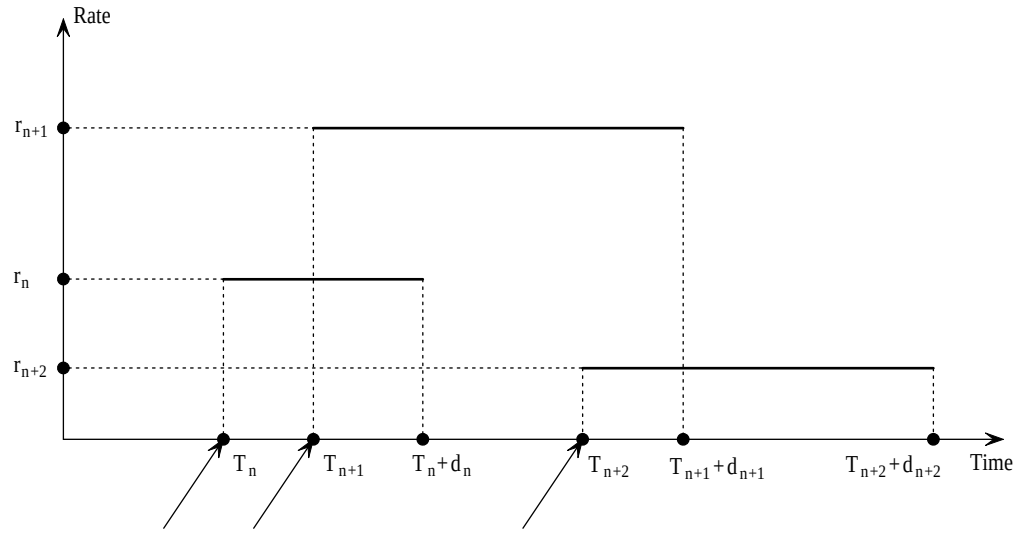

Figure 4.1: Rectangular shots representing individual flows

Many studies in the literature (e.g., [32], [33] and [34]) have shown the self-similar nature of the Internet traffic over large time scales. Self-similarity (burstiness) has been explained by the heavy-tailed distribution of transfer durations. In this study, durations of the flows are generated using the *Pareto* model. According to this model, duration of an individual flow d is generated by using the complementary distribution function given below:

$$Pr(d > x) = \begin{cases} (x/\delta)^{-\gamma}, & \text{for } x \geq \delta, \\ 1 & \text{otherwise.} \end{cases}$$

where $\delta > 0$ and $1 < \gamma < 2$ and according to these the mean value of the duration is given by the following:

$$E(d) = \frac{\delta\gamma}{(\gamma - 1)} \tag{4.1}$$

Size of the flow is generated using the normal distribution with mean μ in terms of $Mbits$ and variance σ^2 and for all the simulations in this chapter, flow size and durations are assumed to be independent random variables.

Using the modelling structures given above, dynamic capacity adjustment problem is applied to flow-based Internet traffic and our optimization criteria will again be to minimize the bandwidth usage cost for long-term given a desired signaling rate. In our formulations we used two different control strategies. In *event-driven* control, decision epochs for sizing the bandwidth of the VP is selected to be the arrival or departure times of the individual flows assuming that the aggregator gateway is able to detect those instants. In *time-driven* control, decision epochs are equidistant time instants and the controller has an opportunity to update the bandwidth of the VP at every pre-specified time interval T. We propose this control strategy because it can be applied to real traffic traces easily since the detection of the arrival and departure instants of the flows can be difficult. In the following sections these control strategies and experimental results are given in detail.

4.1 Event-Driven Control

We define our state space as the following:

$$\mathbf{S} = \{s | s = (s_a, s_r, s_b), 0 \leq s_a \leq C_{max}, \max(0, s_a - 1) \leq s_r \leq C_{max}, 0 \leq s_b \leq B_{max}\},$$

For each state (s_a, s_r, s_b) an action value s'_r satisfying $s_a \leq s'_r \leq C_{max}$ will be chosen. In this part, s_a refers to the total rate of the active flows in the system that is rounded to an integer value. Let NF denotes the number of active flows in the system and let r_i denote the rate of the i'th flow. Then s_a will be computed by the following:

$$s_a = round(\sum_{1 \leq i \leq NF} r_i) \tag{4.2}$$

Total rate can be a non-integer value and this causes a problem since this is a continuous-state space problem. In order to apply the Gosavi algorithm we approximate the continuous state space problem by a discrete state space formulation by rounding the total rate to the nearest integer by the $round(\cdot)$

operation. Again s_r denotes the current bandwidth of the VP and it is an integer value in the range $[0, C_{max}]$ where C_{max} is an integer value and defined to be the maximum amount of bandwidth that can be assigned to the VP in terms of $Mbits/s$. Definitions for the s_b and B_{max} are the same as they are defined before.

C_{max} is selected to be 60 $Mbits/s$. Again average flow duration ($E(d)$) is selected to be 180 $seconds$. Then the flow arrival rate λ is found from the formula given below [35]:

$$p_k = \frac{(\lambda \overline{x})^k}{k!} e^{-\lambda \overline{x}} \tag{4.3}$$

In this formula, p_k is defined to be the steady state probability of the number of the customers in the $M/G/\infty$ system being equal to the value k. Also $\overline{x}$ denotes the mean holding time of each customer and it represents the mean duration of the flows in our formulation. We assume all the flows have constant rate which is equal to 1 $Mbits/s$ and we found the arrival rate λ by selecting the k to be equal to 60 (this case denotes that VP's capacity is full) and we choose the probability (p_{60}) to be equal to 0.01.

In the subsections below different simulation results will be presented according to varying values of desired signaling rate D, mean size of the flows μ, standard deviation of the size of the flows σ and mean duration of the flows $E(d)$.

4.1.1 Varying D

Gosavi algorithm is applied to different values of D. Other problem parameters are chosen to be as, C_{max}=60 $Mbits/s$, $E(d)$=180 $seconds$, λ=0.0668, μ=540 $Mbits$, σ=108 $Mbits$ and the maximum number of iterations in the Gosavi algorithm is selected to be 10^7 steps. Table 4.1 shows the results in terms of bandwidth gain BG and signaling rate SR for different values of D. Also Figure 4.2 shows the policy behavior of the Gosavi algorithm for a time period of 1 hour and for different values of desired signaling rate. It is seen from the results that when the allowed signaling rate increases, average bandwidth usage decreases as expected. Also observed signaling rate ideally tracks the desired signaling rate for all the values of D. The results are very similar to the results obtained by

Table 4.1: Results for varying D.

$D(updates/hour)$	$BG\%$	SR
10	0.63	9.96
20	1.18	20.04
30	2.82	30.00
60	9.63	60.04
100	21.15	100.00
120	22.47	120.08
140	24.10	140.00

Table 4.2: Results for varying μ.

$\mu(Mbits)$	$BG\%$	SR
720	2.65	60.00
540	9.63	60.04
360	19.67	60.00
180	44.98	59.96
90	57.57	59.96

voice traffic modelling and this is shows that our method still performs well for the case of a more complex traffic modelling structure.

4.1.2 Varying μ

In this part different values for the mean of size of the flows are tested. D is taken to be 60 $updates/hour$. Other problem parameters are the same as above. Table 4.2 and Figure 4.3 show the results for this case. As the mean size of the flows decreases, average bandwidth usage decreases and for all the values of μ, actual signaling rate ideally tracks the desired one. In Figure 4.3, there are some huge jumps to the value $C_{max} = 60$. The reason of these jumps is that the leaky bucket is full and according to our working principle, the bandwidth assigned to the VP is forced to be the maximum value that can be assigned.

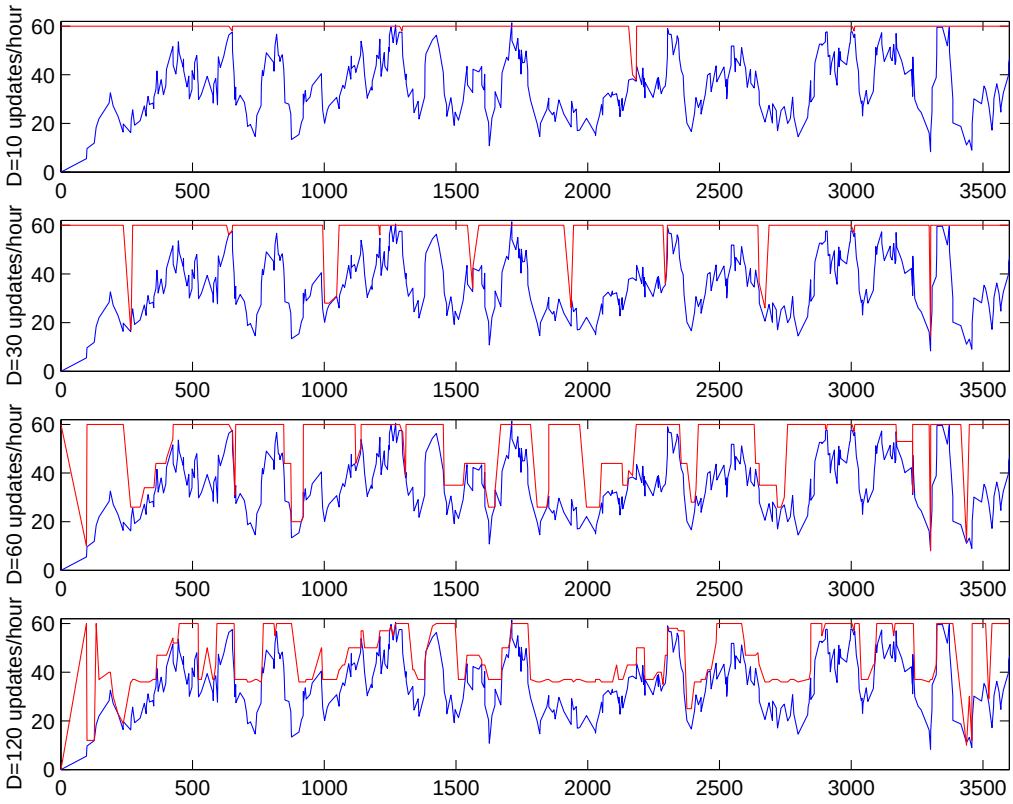

Figure 4.2: Policy behaviour for different values of D

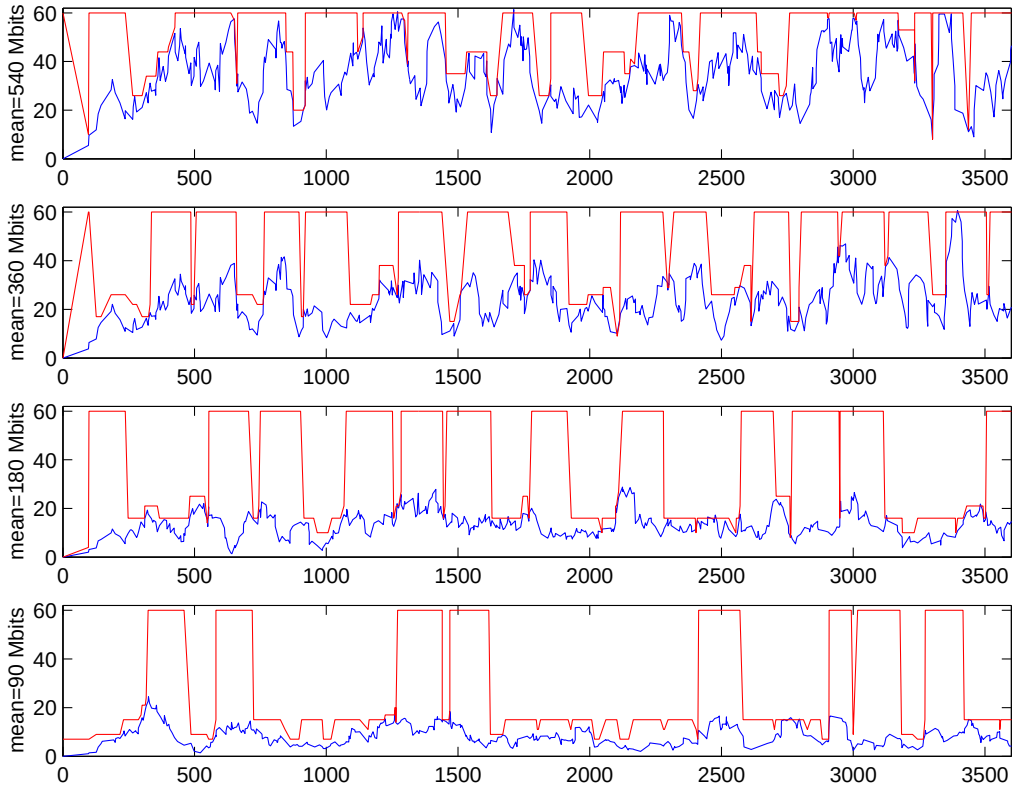

Figure 4.3: Policy behaviour for different values of μ

Table 4.3: Results for varying $E(d)$.

$E(d)(seconds)$	$BG\%$	SR
18	0.65	4.50
54	10.13	59.92
180	9.64	60.04
360	10.75	60.00

4.1.3 Varying $E(d)$

In this part, numerical results are obtained for different values of mean flow duration times. D is selected to be 60 $updates/hour$ and the mean size of the flows is selected to be 540 $Mbits$. Table 4.3 and Figure 4.4 show the results for this case. For a very small value of $E(d)$=18, system behaves very oscillatory (since the mean duration of the flows decreases and flow departure rate increases) and the bandwidth gain for this case is small due to this oscillatory behavior. For larger values of mean duration time, bandwidth gain increases and the system behaves more stable as seen from the Figure 4.4.

4.1.4 Varying σ

The effect of the standard variation of the size of the flows is studied in this part. D is selected to be 60 $updates/hour$ and the mean size of the flows is selected to be 360 $Mbits$. Other parameters are chosen as the ones in Section 4.1.1. Table 4.4 and Figure 4.5 show the results for this case. As the standard deviation decreases, bandwidth gain increases since the huge jumps and oscillations in the traffic behavior decreases resulting a stable learning behavior with less jumps to C_{max}.

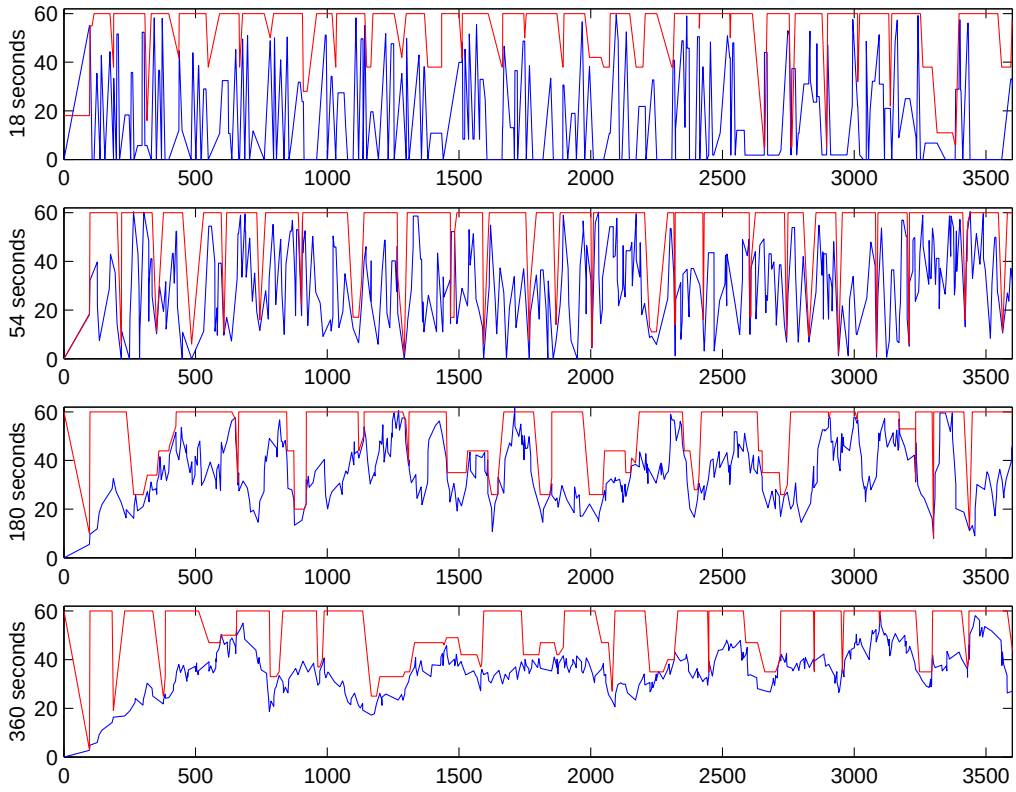

Figure 4.4: Policy behaviour for different values of $E(d)$

Table 4.4: Results for varying σ.

$\sigma(Mbits)$	$BG\%$	SR
360	16.10	60.00
270	15.05	59.96
180	20.95	59.96
90	23.07	60.00

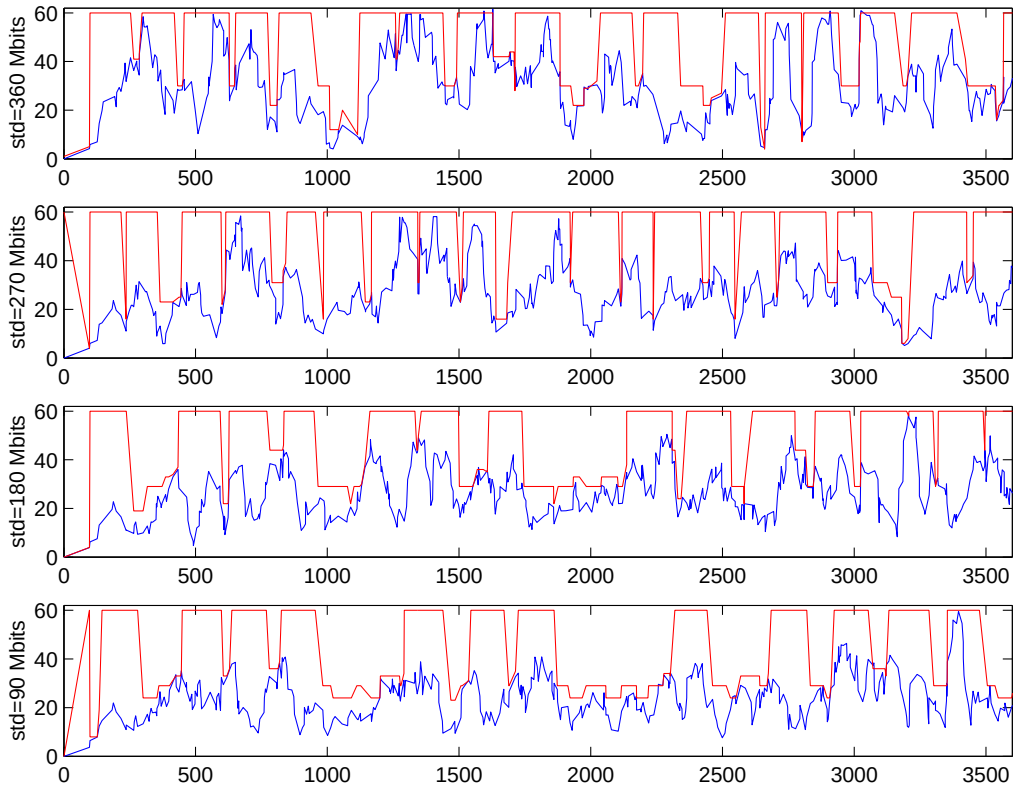

Figure 4.5: Policy behaviour for different values of σ

4.2 Time-Driven Control

In this part, decision epochs are selected to be the equidistant time instants and the time interval between two successive decision instants is denoted by the constant T. At the start of a new interval, a decision is made according to the value of the traffic in the previous interval, current reserved bandwidth of the VP and the state of the leaky bucket counter. With this decision the bandwidth of the VP is set to new value which will be fixed throughout the new interval. Since amount of the traffic can exceed the bandwidth assigned to the VP, we assumed that the excessive amount of traffic that can't be carried will be buffered by the aggregator. For the sake of simplicity we assume that buffer limit is infinite. Since there is a buffering mechanism, there will be no blocking in the system and excessive traffic will be buffered. We define our state space as follows:

$$\mathbf{S} = \{s | s = (s_a, s_r, s_b), 0 \leq s_a \leq C_{max}, 0 \leq s_r \leq C_{max}, 0 \leq s_b \leq B_{max}\},$$

For each state (s_a, s_r, s_b) an action value s'_r satisfying $s_a \leq s'_r \leq C_{max}$ will be chosen. State space looks like the same as defined previously for the other formulations but here the definition for the variable s_a is different. In this definition, s_a denotes the average traffic in the previous interval. Let $NF(t)$ denotes the number of flows in the system at the previous interval and let r_i denotes the rate of the i'th flow, s_a will be calculated using the formula given below:

$$s_a = round(\frac{\int_0^T (\sum_{1 \leq i \leq NF(t)} r_i) dt}{T}) \tag{4.4}$$

As seen from the relation, s_a is found by averaging the traffic in the previous interval. Since we assume no blocking in the system, s_a can be larger than the value C_{max}, for these values of s_a we will set it to the value C_{max} again. Otherwise state space dimension will grow without a bound. With this choice, the state $s_a = C_{max}$ will denote the states that hold the relation $s_a \geq C_{max}$ and represents the instants that total traffic in the system is larger or equal to the total maximum capacity of the link. The other variables s_r and s_b are defined in the same manner in the previous section. With these assumptions a new dimension will be added

Table 4.5: Results for varying T.

$T(seconds)$	$BG\%$	SR	$BF(Mbits)$
10	8.75	60.04	4.97
20	12.87	59.96	9.53
30	20.63	59.75	17.38
40	21.87	59.96	21.16

to the problem: assigning the value of the time interval T. If it is chosen very large, the averaging operation will cause the total traffic in the system will be smoothed very much and oscillations in the traffic will be neglected resulting a smaller amount of bandwidth usage. If it is chosen very small, than the system reacts to the oscillations and the jumps in the system rapidly so the resulting policy will be an unstable and oscillatory one resulting in an inefficient usage of bandwidth. In the numerical results that will be presented below, different values of T and desired rate D is tested.

4.2.1 Varying T

Different values for T is tested. Other problem parameters are selected to be C_{max}=60 $Mbits/s$, $E(d)$=180 $seconds$, λ=0.0668, μ=540 $Mbits$, σ=108 $Mbits$ and D is selected to be 60 *updates per hour*. Same traffic is generated for all the values of T. Results in Table 4.5 are given in terms of bandwidth gain, signaling rate and average buffer size in the time interval T denoted by BF. When time interval T increases, smoothing of the traffic will increase and the oscillations with jumps will be neglected more. This results a higher buffer occupancy with decreasing bandwidth usage as seen from the results and sample Figure 4.6.

4.2.2 Varying D

Again different values of desired rate is tested. T is chosen to be 30 *seconds*. Other parameters are same as in the previous subsection. Results are in Table

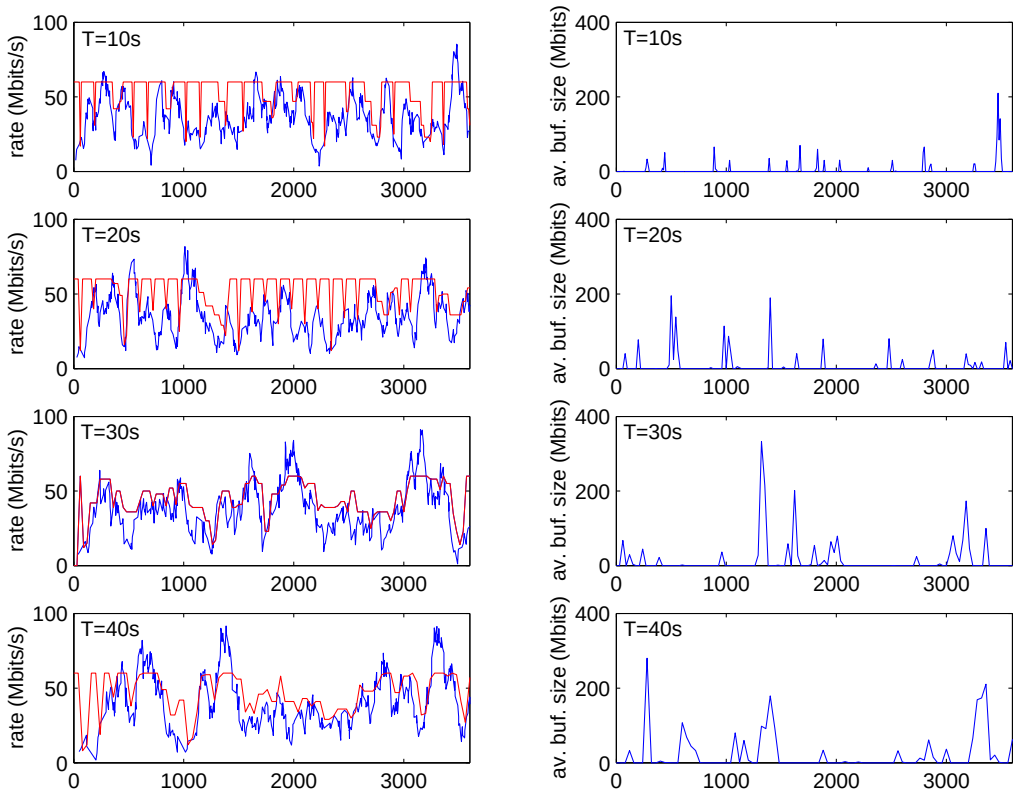

Figure 4.6: Policy behaviour and buffer occupancy for different values of T

Table 4.6: Results for varying D.

$D(updates/hour)$	$BG\%$	SR	$BF(Mbits)$
60	20.63	59.75	17.38
40	14.98	40.00	12.21
20	6.93	19.96	10.86
10	2.77	10.00	8.21

4.6 and Figure 4.7. Again same traffic is generated for all the values of D. As seen from the results, when D decreases, average bandwidth gain decreases with a decreasing average buffer occupancy as expected. Also observed signaling rate ideally tracks the desired rate for all the values of D.

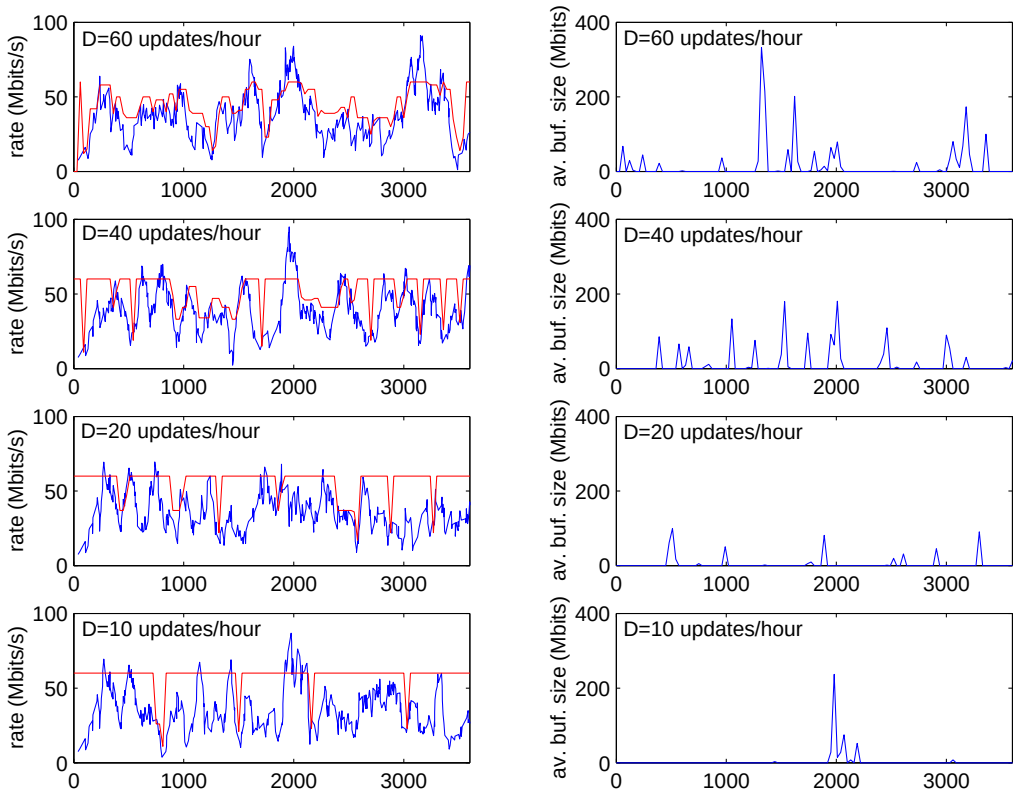

Figure 4.7: Policy behaviour and buffer occupancy for different values of D

Chapter 5

Conclusions and Future Work

In this thesis, we solved the dynamic capacity adjustment problem for virtual path based networks using the semi-Markov decision framework. We used dynamic programming (DP) and neuro-dynamic programming (NDP) techniques for reaching the optimal or sub-optimal capacity adjustment policy. Generally speaking, NDP techniques are more scalable than their DP counterparts. In the experiments, when the state space dimensionality increases NDP methods including state aggregation and function approximation are used successfully.

Two important issues in this problem are the bandwidth usage efficiency and the signaling traffic in the network. We proposed two different formulations for dynamic capacity adjustment. In the first formulation, cost parameters are assigned for a single capacity update (S) in the network and bandwidth usage per unit time (b). For different values of S/b ratio we apply both DP and NDP methods with voice traffic modelling. We saw that, for small-sized problems NDP results behave very similar to the DP results except for the cases where this ratio is very small or very large. At those cases, especially function approximation technique (A-RVI-FA) behaves unstable which is a reason of the unstable and problem-dependent nature of function approximation methods in NDP framework. We compared our results with two different approaches namely the SVC approach and the PVP approach. Results show that, for all the cases of S/b we achieve a better performance in terms of long run average cost compared with the

SVC and the PVP. Also for large sized problems we show that the sNDP method A-RVI scales well and gives good results.

In the second formulation, we try to solve the problem of capacity adjustment under signaling constraints. Our aim is to minimize the bandwidth underutilization given a signaling rate constraint. In order to regulate the signaling rate, we used a variant of the generic leaky bucket counter and this is the key point of this part. With the addition of the new state variable denoting the value of the bucket, our state space complexity and dimensionality increases so we used a model-free NDP method: the Gosavi algorithm and again we used voice traffic modelling. From the experimental results we see that for all the values of the desired signaling rate, our policy achieves an actual signaling rate which ideally tracks the desired value. We achieve a significant bandwidth gain compared with respect to the PVP approach under the signaling constraint. In addition to this, a uniform state aggregation technique using the Gosavi algorithm is used in order to increase the scalability and to solve large-sized problems. We see from the results that when we aggregate the state space aggressively, convergence performance of the algorithm improves with the disadvantage of information loss. For a large-sized problem we achieve larger bandwidth gain with the usage of state aggregation compared to the original problem where no aggregation is applied.

Our methods are applicable for more complex traffic models. We used flow-based Internet traffic modelling using Poisson shot noise process and apply the Gosavi algorithm for this case. We proposed two different control strategies: event-driven and time-driven. In event-driven case, again we achieve a significant amount of bandwidth gain with a given signaling constraint. Also our method reacts successfully to the changes in the traffic intensity which is affected by mean flow duration, mean flow size and the standard deviation for flow size distribution. For time-driven control strategy, we assume a buffering mechanism and we tested our method for different values of constant T which is defined to be the time interval between decision epochs. When T increases, the oscillations in the traffic is neglected and a larger bandwidth gain is achieved with the usage of buffering. Another important result is that, for Internet traffic modelling and both of the control strategies, again the actual desired rate ideally tracks the desired one

which shows that our leaky bucket mechanism is very powerful for regulating the signaling rate.

In this thesis, we introduce an important footstep for the problem of dynamic capacity adjustment for virtual path based networks. We presented our results for voice traffic modelling in [36], [37] and [38]. We list our future research plan below:

- We applied a time-driven control strategy for Internet traffic modelling and our assumption in this thesis is that the buffer limit is infinite. In practice, there is a limit on the buffer size and a new formulation is needed for controlling the buffer occupancy. We plan to add a new variable to our state space which will denote the level of buffer occupancy (e.g., very low, low, medium, large, very large). We are planing to use the similar leaky bucket methodology for controlling the buffer level together with achieving the maximum bandwidth gain with signaling constraint.

- Gosavi algorithm with time-driven control strategy will be applied to real Internet traffic traces. In this case, underlying Markov model will be completely lost and this work will be very useful showing the model-free applicability of NDP methods.

- Finally, a network environment that is composed of different virtual paths will be simulated. VPs will share the same bandwidth pool and our optimization criteria will be the maximization of bandwidth efficiency throughout the network with signaling constraints. Also different classes of traffic with different QoS requirements will be considered.

Bibliography

[1] B. Davie and Y. Rekhter, "MPLS: Technology and Applications", Morgan Kaufmann Publishers, 2000.

[2] "ATM User Network Interface (UNI)", ATM Forum Specification version 4.0, AF-UNI-4.0, July 1996.

[3] F. Baker, C. Iturralde, F. Le Faucheur and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.

[4] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, 1999.

[5] C. Barakat, P. Thiran, G. Iannaccone, C. Diot and P. Owezarski, "A Flow-based Model for Internet Backbone Traffic", ITC, 2002.

[6] M. Zukerman, T. D. Neame and R. G. Addie, "Internet Traffic Modelling and Future Technology Implications", IEEE Infocom, 2003.

[7] H. C. Tijms, "Stochastic Models: An Algorithmic Approach", John Wiley and Sons Ltd., 1994.

[8] D. P. Bertsekas, "Dynamic Programming and Optimal Control, Vols. I and II", Athena Scientific, 1995.

[9] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-Dynamic Programming" Athena Scientific, Belmont, MA, 1996.

[10] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", MIT Press, 1998.

[11] S. Mahadevan, "Average Reward Reinforcement Learning: Foundations, Algorithms and Empirical Results", Machine Learning, 22, 159-196, 1996.

[12] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis, "Call Admission Control and Routing in Integrated Service Networks Using Neuro-Dynamic Programming," IEEE Journal on Selected Areas in Communications, Vol. 18, No. 2, February 2000, pp. 197-208.

[13] P. Marbach, and J.N. Tsitsiklis, "A Neuro-Dynamic Programming Approach to Call Admission Control in Integrated Service Networks: The Single Link Case," Technical Report LIDS-P-2402, Laboratory for Information and Decision Systems, M.I.T., November 1997.

[14] S. Singh and D. P. Bertsekas, "Reinforcement Learning for Dynamic Channel Allocation in Celluar Telephone Systems", Advances in Neural Information Processing Systems 9, MIT Press, 1997

[15] H. Tong and T. X. Brown, "Reinforcement Learning for Call Admission Control under Quality of Service Constraints in Multimedia Networks", Machine Learning, 49, 111-139, 2002.

[16] Serkan Yesildag, "Dynamic Routing and Wavelength Assignment in Wavelength-Division Multiplexed (WDM) Optical Networks Using Neuro-Dynamic Programming", Master Thesis, Bilkent University, Electrical and Electronics Engineering Department, 2003.

[17] Nuri Celik, "Using Reinforcement Learning for Dynamic Link Sharing Problems Under Signalling Constraints", Master Thesis, Bilkent University, Electrical and Electronics Engineering Department, 2003.

[18] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", RFC 2211, 1997.

[19] S. Shenker, C. Partridge and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, 1997.

[20] J. Wroclawski, "The Use of RSVP with IETF Integrated Services ", RFC 2210, 1997.

[21] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[22] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, 1998.

[23] D. Clark, S. Shenker and L. Zhang, "Supporting Real-time Applications in an Integrated Services Packet Network: Architecture and Mechanism", in Proc. SIGCOMM'92, September 1992.

[24] S. Shiodam, H. Saito and H. Yokoi, "Sizing and Provisioning for Physical and Virtual Path Networks Using Self-sizing Capability", IEICE Trans. Commun., Vol. E80-B, No. 2, February 1997.

[25] B. Groskinsky, D. Medhi and D. Tipper, "An Investigation of Adaptive Capacity Control Schemes in a Dynamic Traffic Environment" , IEICE Trans. Commun., Vol. E00-A, No, 13, 2001.

[26] T. Anjali, C. Scoglio and G. Uhl, "A New Scheme for Traffic Estimation and Resource Allocation for Bandwidth Brokers", Computer Networks, vol.41, pp. 761-777, 2003.

[27] H. Levy, T. Mendelson and G. Goren, "Optimal Use of Virtual Paths for Connection Setup Reduction: The Single Link Problem", IEEE Infocom, 2000.

[28] Cisco, White Paper, "Cisco MPLS AutoBandwidth Allocator for MPLS Traffic Engineering: A Unique New Feature of Cisco IOS Software".

[29] A. Jalali and M.Ferguson. "Computationally efficient adaptive control algorithms for Markov chains", In Proceedings of the 28th. IEEE Conference on Decision and Control, pages 1283-1288, 1989.

[30] A. Gosavi "Reinforcement Learning for Long-run Average Cost" To appear in the European Journal of Operational Research.

[31] C. Darken, J. Chang, and J. Moody. "Learning Rate Schedules for Faster Stochastic Gradient Search". Neural Networks for Signal Processing 2– Proceedings of the 1992 IEEE Workshop. 1992.

[32] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, vol. 5,no. 6, pp. 835-846, Dec. 1997.

[33] W. Leland, M. Taqq, W. Wilinger and D. Wilson, "On the self-similar nature Ethernet traffic", ACM SIGCOMM, September 1993.

[34] V. Paxson and S. Floyd, " Wide-Area Traffic: The Failure of Poisson Modelling", IEEE/ACM Transactions on Networking, vol. 3,no. 3, pp. 226-244, June 1995.

[35] L. Kleinrock, "Queueing Systems Volume I: Theory", John Wiley and Sons Ltd., 1975.

[36] N. Akar and C. Sahin, "Reinforcement Learning as a Means of Aggregate QoS Provisioning", Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Volume 2698, pp. 100 - 114, 2003.

[37] N. Akar and C. Sahin, "Dynamic Capacity Provisioning in Virtual Path-Based Networks Using Reinforcement Learning", International Teletraffic Congress, Berlin, Germany, Sep. 2003.

[38] N. Akar and C. Sahin, "Dynamic Capacity Management for Voice over Packet Networks", International Symposium on Computers and Communications, Antalya, Turkey, July 2003.