Dynamic Capacity Management for Voice Over Packet Networks

Nail Akar
Electrical and Electronics Eng. Dept.,
Bilkent University
06800 Bilkent, Ankara, Turkey
akar@ee.bilkent.edu.tr

Cem Sahin
Electrical and Electronics Eng. Dept.,
Bilkent University
06800 Bilkent, Ankara, Turkey
csahin@ee.bilkent.edu.tr

Abstract

In this paper, dynamic capacity management refers to the process of dynamically changing the capacity allocation (reservation) of a pseudo-wire established between two network end points. This process is based on certain criteria including instantaneous traffic load for the pseudo-wire, network utilization, time of day, or day of week. Frequent adjustment of the capacity yields a scalability issue in the form of a significant amount of message processing in the network elements involved in the capacity update process. On the other hand, if the capacity is adjusted once and for the worst possible traffic conditions, a significant amount of bandwidth may be wasted depending on the actual traffic load. There is then a need for dynamic capacity management that takes into account the tradeoff between scalability and bandwidth efficiency. This problem is motivated by voice over packet networks in which end-to-end reservation requests are initiated by PSTN voice calls and these reservations are aggregated into one single reservation in the core packet network for scalability. In this paper, we introduce a Markov decision framework for an optimal reservation aggregation scheme for voice over packet networks. Moreover, for problems with large sizes, we provide a suboptimal scheme using reinforcement learning. We show a significant improvement in bandwidth efficiency in voice over packet networks using aggregate reservations.

1. Introduction

In this paper, dynamic capacity management refers to the process of dynamically changing the capacity reservation of a pseudo-wire or a VP (Virtual Path) set up between two network end points based on certain criteria including instantaneous traffic load for the virtual path, network utilization, time of day, or day of week. "Pseudo-wire" in this definition is to be viewed as a generic path carrying aggregate traffic between two network end points. The route of the pseudo-wire is fixed and the capacity allocated to it can dynamically be resized on-line (without need for tearing it down and reestablishing it with a new capacity) using signaling. With this generic definition, multiple networking technologies can be accommodated; a pseudo-wire may be an MPLS-TE (MultiProtocol Label Switching - Traffic Engineering) LSP (Label Switched Path) [6], an ATM (Asynchronous Transfer Mode) VP [1], or a single aggregate RSVP (Resource ReserVation Protocol) reservation [2]. The end points of the pseudo-wire will then be LSRs (Label Switch Router), ATM switches, or RSVP-capable routers.

Figure 1 depicts a general voice over packet network. At the edge of the packet network, there are the voice over packet gateways which are interconnected to each other using pseudo-wires. The packet network may be an MPLS, or 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 are received by 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.

An aggregate of voice calls flows through the pseudo-wire in Figure 1. This enables possible aggregation of forwarding, scheduling, and classification state through the packet network, thus enhancing the scalability of core routers and switches. The capacity allocated to the aggregate should ideally track the actual aggregate traffic for optimal use of resources but this policy requires a substantial amount of signaling and message processing and it would not scale to large networks with rapidly changing traffic. For example, consider two "voice over packet" gateways interconnected to each other using a pseudo-wire. Calls from the PSTN are admitted into the pseudo-wire only when there is enough bandwidth and once admitted, traffic is packetized and forwarded from one gateway to the other

Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)
1530-1346/03 $17.00 © 2003 IEEE

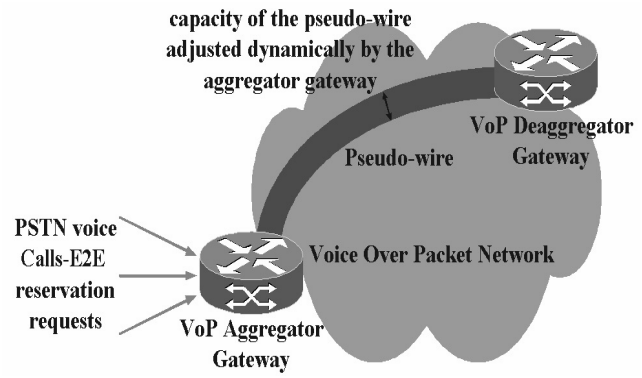

Figure 1. E2E (End-to-End) reservations due to PSTN voice calls are aggregated into one single reservation through the voice over packet network

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 pseudo-wire may be adjusted for optimal use of resources. This approach will be referred to as the SVC (Switched Virtual Circuit) approach throughout this paper since the messaging and signaling requirements of this approach will be very similar to the case where each voice call uses its own SVC. Another approach to engineer the pseudo-wire 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. Motivated by ATM networks, we call this approach the PVP (Permanent Virtual Path) approach. However, the downside of this approach is that the capacity may be vastly underutilized when the load is significantly lower than the allocated capacity, which is the peak load. In this case, this idle capacity would not be available to other aggregates that actually need it and this would lead to inefficient use of resources.

In this paper, we propose a DCM (Dynamic Capacity Management) approach that utilizes the tradeoff between optimality and scalability by resizing the capacity only when the incremental cost of signaling and message processing is less than the reward of updating the capacity. As an example, let us assume that the network nodes in the aggregation region can handle at most N capacity update requests per hour, which is the scalability requirement. Assuming that on the average there are M output interfaces on every node and L pseudo-wires established on every such interface, an individual pseudo-wire may be resized on the average $N/(ML)$ times in every hour. With typical values of $N = 36000$ (10 capacity updates per second for an individual network node), M=16, and L=100, one can afford adjusting the capacity of each pseudo-wire 22.5 times in a single hour. The goal of the DCM approach is to minimize the idle capacity between the allocated capacity and the actual bandwidth over time while satisfying the scalability requirement, i.e., by resizing the capacity of the pseudo-wire less than 22.5 times per hour (on the average).

There are several techniques proposed in the literature to solve the dynamic capacity allocation problem. In [10], the capacity of the pseudo-wire 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 [7], blocking rates are calculated for the pseudo-wire 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.

Our proposed approach is based on the average reward Markov decision framework [12] which has been a popular paradigm for sequential decision making under uncertainty. Such problems can be solved by Dynamic Programming (DP) [12] which provides a suitable framework and algorithms to find optimal policies. Policy iteration and relative value iteration [12] are the most commonly used DP algorithms for average reward Markov decision problems. However, these algorithms become impractical when the underlying state-space of the Markov decision problem is large, leading to the so-called "curse of dimensionality". Recently, an adaptive control paradigm, the so-called "Reinforcement Learning" (RL) [11], [3] has attracted the attention of many researchers in the field of Markov decision processes. RL is based on a simulation scenario in which an agent learns by trial and error to choose actions that maximize the long-run reward it receives. RL methods are known to scale better than their DP counterparts [11]. In this paper, we propose a DP and an RL-based algorithm to find optimal capacity management policies for voice over packet networks.

The remainder of the article is organized as follows. In Section 2, general QoS architectures including the aggregate reservations concept are reviewed and compared and contrasted with each other in terms of performance and scalability. The Markov decision framework for optimal aggregate reservations as well as a reinforcement learning approach are presented in Section 3. Section 4 provides numerical examples to demonstrate the efficacy of the proposed approach. The final section is devoted to conclusions

and future work.

2 QoS Models

Several QoS architectures that are proposed by the IETF (Internet Engineering Task Force) for IP networks will now briefly be reviewed and how they relate to the scalable aggregate reservation scenario will then be presented.

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. 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. 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.

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 [9], [4]. 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, 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.

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 [2]. 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 [5] 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 basic 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.

3 Semi-Markov Decision Framework

We consider a voice over packet network as in Figure 1 that supports aggregate reservations. We assume E2E reservation requests are identical and they arrive at the aggregator 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), and we assume that there is an upper limit C_{max} units for the aggregate reservation. We suggest to set C_{max} to the minimum capacity required to achieve a desired blocking probability p. C_{max} is typically 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. In our simulation studies, we take $p = 0.01$.

A tool to analyze such systems and to find optimal policies is the semi-Markov decision model [12]. This model concerns a dynamic system which at random points in time is observed and classified into a possible number of states. We denote the set of possible states in our model by $\mathbf{S}$:

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

where s_a refers to the number of active voice calls using the pseudo-wire just after an event which is defined either as a call arrival or a call departure. The notation s_r denotes the amount of aggregate reservation before the event. For each $s = (s_a, s_r) \in S$, 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 is a random variable that depends only on s_a and its expected value is denoted by τ_s. We assume two types of incremental costs incurred when at state $s = (s_a, s_r)$ until the next decision epoch; the first cost is the expected cost of reserved bandwidth which is expressed as $b\tau_s s'_r$ where b is the cost of reserved unit bandwidth per unit time. 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. 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 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. This formulation fits very well into a semi-Markov decision model where the long-run average cost is taken as the optimality criterion. We propose the relative value iteration algorithm for this problem [12].

3.1 Relative Value Iteration (RVI)

Our goal is to minimize the average cost per unit time as opposed to the total cumulative discounted cost, because our problem has no meaningful discount criteria. Our approach is outlined below but we refer the reader to [12] for details. A data transformation is first used to convert the semi-Markov decision problem to a discrete-time Markov decision model with the same state space [12]. For this purpose, let $c_s(s'_r)$ denote the expected cost until next state when the current state is $s = (s_a, s_r)$ and action s'_r is chosen. Also let $\tau_s(s'_r)$ denote the expected sojourn time in state s when action s'_r is chosen. Expected immediate costs and one-step transition probabilities of the converted Markov decision model are given as [12]:

$$\widetilde{c}_s(s'_r) = \frac{c_s(s'_r)}{\tau_s(s'_r)} \quad (1)$$

$$\widetilde{p}_{s,s'}(s'_r) = \frac{\tau}{\tau_s(s'_r)} p_{s,s'}(s'_r), s' \neq s \quad (2)$$

$$\widetilde{p}_{s,s'}(s'_r) = \frac{\tau}{\tau_s(s'_r)} p_{s,s'}(s'_r) + (1 - \frac{\tau}{\tau_s(s'_r)}), \quad s' = s. \quad (3)$$

where

$$0 < \tau \leq min_{(s,s'_r)}\tau_s(s'_r)$$

With this transformation, the relative value iteration algorithm is given as follows [12]:

Step 0 Select $V_0(s)$ from $0 \leq V_0(s) \leq \min_{s'_r} \widetilde{c}_s(s'_r)$ and $n := 1$.

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

$$\begin{aligned} V_n(s) &= \min_{s'_r}[\widetilde{c}_s(s'_r) + \frac{\tau}{\tau_s(s'_r)} \sum_{s'} p_{s,s'}(s'_r)V_{n-1}(s') \\ &\quad +(1 - \frac{\tau}{\tau_s(s'_r)})V_{n-1}(s)] \qquad (4) \\ V_n(s) &= V_n(s) - V_n(s_0) \qquad (5) \end{aligned}$$

Where s_0 is the pre-specified reference state.

Step 2 Compute the following bounds

$$\begin{aligned} m_n &= \min_s(V_n(s) - V_{n-1}(s)) \\ M_n &= \max_s(V_n(s) - V_{n-1}(s)) \qquad (6) \end{aligned}$$

Algorithm is stopped resulting if the following convergence condition is satisfied

$$0 \leq (M_n - m_n) \leq \varepsilon m_n \qquad (7)$$

Where ε is a pre-specified tolerance. This condition signals that there are no more meaningful changes in the values of the states $\{V_n(s)\}$. If convergence condition is not satisfied, let $n := n+1$ and go to *Step 1*. Otherwise, the optimal policy is obtained by choosing the argument that minimizes the right hand side of (4).

3.2 Reinforcement Learning and Asynchronous Relative Value Iteration (A-RVI)

When the state space of the underlying Markov decision problem is large, dynamic programming algorithms will be intractable and we suggest to use reinforcement learning algorithms in such cases to obtain optimal or sub-optimal solutions. In particular, we propose the asynchronous version of RVI, the so-called Asynchronous Relative Value Iteration (A-RVI) that uses simulation-based learning and asynchronous updating instead of batch updating the values of the states [8]. A-RVI is described 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:

$$\begin{aligned} \min_{s'_r}[\widetilde{c}_s(s'_r) + \frac{\tau}{\tau_s(s'_r)} \sum_{s'} p_{s,s'}(s'_r)V_{n-1}(s') \\ +(1 - \frac{\tau}{\tau_s(s'_r)})V_{n-1}(s)] \qquad (8) \end{aligned}$$

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

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

Step 3 $n := n + 1$, $s := s', V(s_0) = 0$. Stop if $n = max_{steps}$, else goto *STEP 1*.

The algorithm terminates with the stationary policy comprising the actions that minimize (8). κ_n is the learning rate which is forced to die with increasing number of iterations. Exploration is crucial in guaranteeing the convergence of this algorithm and we suggest to use the ϵ-directed heuristic search which means that with some small probability ϵ, we choose an exploratory action (as opposed to the best possible action) at each iteration that would lead the process to the least visited state [8].

4 Numerical Results

We verify our approach by comparing RVI and A-RVI with the two traditional reservation mechanisms, namely SVC and PVP. The problem parameters are chosen as $\lambda = 0.0493$ calls/sec., $\mu = 1/180$ sec., $C_{max} = 16$. We run ten different 12 hour simulations for different values of S/b, and average of these simulations are reported. Figure 2 shows the average performance metrics: average cost, average reserved bandwidth and number of capacity updates per hour using different methods. Irrespective of the cost ratio S/b, policies obtained via RVI and A-RVI give very close results for the average cost. However, there is a slight difference in the optimal policies found using RVI and A-RVI since the average reserved bandwidth and average number of capacity updates with the RVI and A-RVI policies are found to be different using simulations. When the ratio S/b approaches zero, the RVI and A-RVI policies give very close results to that of the SVC approach. This is expected since when the signaling cost is very low, SVCs provide the most bandwidth efficient mechanism. On the other hand, when the ratio $S/b \rightarrow \infty$, RVI and A-RVI policies very much resemble the PVP approach. This is also intuitive since when the signaling cost is very high, the only option is allocating bandwidth for the aggregate for once in a very long period of time.

Table 1 shows the performance of A-RVI for a larger size problem where the RVI solution is numerically intractable. We take $C_{max} = 300$ and $\lambda = 1.5396$ calls/sec. 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 management scenario. Moreover, A-RVI consistently gives better results than both PVP and SVC in terms of the overall average cost.

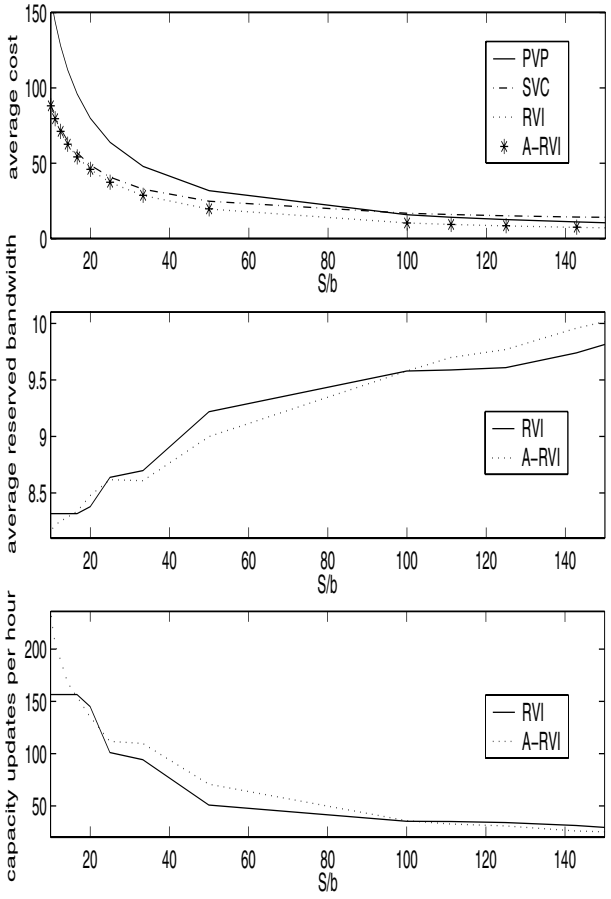

Figure 2. Average cost, average reserved bandwidth, and average number of capacity updates using PVP, SVC, RVI, and A-RVI for the case $\lambda = 0.0493$ calls/sec., $\mu = 1/180$ sec., $C_{max} = 16$

	$S/b = 100$	$S/b = 50$	$S/b = 20$
A-RVI average cost	272.2	524.0	1277
SVC average cost	526.6	775.8	1523
PVC average cost	300	600	1500
A-RVI average reserved bandwidth	272	261	254
A-RVI # of capacity updates per hour	45	550	2418

Table 1. Performance results of the policy obtained via A-RVI for the case $C_{max} = 300$

5 Conclusions

In this paper, we introduce a dynamic capacity management problem that arises in a number of networking scenarios including voice over packet networks. This capacity management problem is posed as a semi-Markov decision problem and the relative value iteration algorithm is proposed in this paper to find optimal policies. In case when the underlying state-space dimensionality is large, we also introduce a reinforcement learning approach, the so-called asynchronous relative value iteration. Through a numerical example motivated by voice over packet networks, we show that with the two methods proposed in this paper, one can achieve substantial bandwidth efficiencies (in contrast with their static counterparts) with proper choices of the cost parameters of the underlying semi-Markov decision problem. Also we show that reinforcement learning solutions scale very well up to large sized problems.

References

[1] Atm forum specification version 4.0, atm user network interface (uni). *AF-UNI-4.0*, July 1996.

[2] F. Baker, C. Iturralde, F. L. Faucheur, and B. Davie. Aggregation of rsvp for ipv4 and ipv6 reservations. *RFC 3175*, September 2001.

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

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *RFC 2475*, 1998.

[5] 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.

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

[7] B. Groskinsky, D. Medhi, and D. Tipper. An investigation of adaptive capacity control schemes in a dynamic traffic environment. *IEICE Trans. Commun.*, E00-A(13), 2001.

[8] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms and empirical results. *Machine Learning*, (22):159–196, 1996.

[9] 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.

[10] S. Shiodam, H. Saito, and H. Yokoi. Sizing and provisioning for physical and virtual path networks using self-sizing capability. *IEICE Trans. Commun.*, E80-B(2), February 1997.

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

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