LIRA: An Approach for Service Differentiation in the Internet [*]

Ion Stoica, Hui Zhang
Carnegie Mellon University
Pittsburgh, PA 15213
e-mail: {istoica,hzhang}@cs.cmu.edu

Abstract

In this paper, we study the Assured Service model proposed by Clark and Wroclawski [3, 4]. While existing schemes use service profiles that are defined in terms of absolute bandwidth, it is difficult, if not impossible, to design provisioning algorithms that achieve simultaneously good service quality and high resource utilization for such services with large spatial granularities.

We propose an Assured Service model, called LIRA (Location Independent Resource Accounting), in which service profiles are defined in units of resource tokens, rather than absolute bandwidth. The number of resource tokens charged for each in-profile packet is a dynamic function of the path it traverses and the congestion level. Defining service profile in terms of resource tokens allows more dynamic and flexible network control algorithms that can simultaneously achieve high utilization and ensure high probability delivery of in-profile packets. We present an integrated set of algorithms that implement the model. Specifically, we leverage the existing routing infrastructure to distribute the path costs to all edge nodes. Since the path cost reflects the congestion level along the path, we use this cost to design dynamic routing and load balancing algorithms. To avoid packet re-ordering within a flow, we devise a lightweight mechanism that binds a flow to a route so that all packets from the flow will traverse the same route. To reduce route oscillation, we probabilistically bind a flow to one of the multiple routes. Simulation results are presented to demonstrate the effectiveness of the approach.

[*]This research was sponsored by DARPA under contract numbers N66001-96-C-8528 and N00174-96-K-0002, and by a NSF Career Award under grant number NCR-9624979. Additional support was provided by Intel Corp., MCI, and Sun Microsystems. Views and conclusions contained in this document are those of the authors and should no be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, MCI, Sun, or the U.S. government.

1 Introduction

As the Internet evolves into a global commercial infrastructure, there is a growing need to support more enhanced services than the traditional best-effort service. To address this issue, several new QoS models (guaranteed, controlled load, committed rate) have been proposed [26, 31]. Collectively, they are called Integrated Services or Intserv models. Recently, there are new efforts in the IETF to develop a new class of service models called Differential Services or Diffserv models [2, 3, 4, 17, 22, 29].

While these schemes differ in details, they are very similar at the architectural level. Usually a scheme consists of the following components: (a) a service profile between each customer (user) and the Internet Service Provider (ISP) that defines the commitment of the ISP to the user, (b) ingress nodes at the ISP edge which police the aggregate traffic from each user to make sure that no user exceeds its service profile, (c) network nodes inside the ISP core which implement a variety of packet forwarding, buffer management, and scheduling behaviors in order to control packet queueing delay, loss, and/or throughput, and (d) a set of bits in the header of each packet used to trigger mechanism for differential processing inside the network. Usually, there are two types of bits. The first type specifies the differential processing behavior requested by the user, such as drop or delay preference. These bits are *not* modified by the routers. The second type of bits can be changed by routers and encodes the dynamic information. An example is the bit that encodes whether the packet is in or out of the service profile.

The key difference between Intserv and Diffserv is that while Intserv provides end-to-end QoS service on a per flow basis, Diffserv is intended to provide service differentiation among the *traffic aggregates* to different users *over a long timescale*. Such difference at the service level has important implications on the complexity of the network-level mechanisms required to implement these services. In particular, to provide Intserv, each router needs to support a flow level signaling protocol such as RSVP [32], maintain per flow state, and perform scheduling and manage buffers on a per flow basis. Since there can be a large number of flows in the Internet, it is an open question whether Intserv can be implemented in a scalable fashion. Diffserv, on the other hand, pushes the complexity to the network edge, and requires very simple priority scheduling/dropping mechanisms inside the core. An important property of the Diffserv schemes considered in this paper is that each router treats identically all packets which have the same bits set. That is, routers only distinguish a small number of aggregated classes of packets, where a class represents all packets with the same marking.

Existing Diffserv schemes are based on the concept of service profile. From the service's point of view, there are three aspects that a Diffserv model needs to specify [3, 4]:

- semantics of the service profile: what exactly is provided to the customer (user)?
- spatial granularity of the service: is the service profile applied to traffic destined to a single destination, a group of destinations, all nodes of an ISP, or everywhere in the Internet?
- level of assurance: how likely is an in-profile packet to be delivered to the destination?

Two examples of differential service models are the Assured Service proposed by Clark and Wroclawski [3, 4] and the Premium Service proposed by Jacobson et. al [22]. The Premium Service provides the equivalent of a dedicated link of fixed bandwidth between two edge nodes. The main advantage of the Premium Service over the current Intserv models such as guaranteed or controlled load is its implementation simplicity – it does not require per flow management at core routers.

The Assured Service supports coarse spatial granularity, i.e., service profiles are applied to traffic defined to more than one destination. It is important to notice that in addition to the implementation simplicity, the Assured Service also provides a service semantic that is richer than those provided by the existing Intserv models. For example, while Intserv and Premium services are better suited for steady and long-lived traffic, the Assured Service provides better support for traffic aggregates that consist of many short-lived bursty flows with different destinations (such as WEB traffic).

We believe that a service model with a coarse spatial granularity embodies some of the fundamental motivations behind the design of the Assured Service. The coarse spatial granularity leads to a smaller number of service profiles, which in turn improves the scalability by reducing the amount of state needed at the edge of the network. In addition, service profiles with coarser spatial granularities also mean higher traffic aggregation for each profile, which allows users to achieve a higher degree of statistical multiplexing gain.

However, as the spatial granularity becomes larger than one destination, it is more difficult to support a service with a fixed bandwidth profile. In this situation, there is a fundamental conflict between maximizing resource utilization and achieving a high service assurance. Since the network does not know in advance where the packets will go, in order to provide high service assurance, it needs to provision enough resources to *all possible* destinations. This will result in a severe resource under-utilization.

In this paper, we show that by defining service profiles in terms of the amount of resource tokens rather than the absolute bandwidth, we can design dynamic and flexible network control algorithms that can achieve high resource utilization, while at the same time delivering in-profile packets with high probability. The key aspect of the model is that the service differentiation is based on resource tokens rather than the exact amount of bits per second. The amount of resource tokens charged for each bit is a dynamic function of the path and the congestion level. This avoids the worst-case provisioning dilemma faced by existing solutions. In addition, as we will discuss in Section 4, our scheme can also be used to implement differential services with service profiles defined in terms of absolute bandwidth.

The rest of the paper is organized as follows. In Section 2 describes our model based on resource tokens and proposes a set of mechanisms to implement the model. Section 3 presents simulation experiments to demonstrate the effectiveness of our solution. Section 4 justifies the new service model and discusses possible ways for our scheme to implement other differential service models. Finally, in Section 5 we present the related work, and in Section 6 we summarize our contributions.

2 *LIRA*: Service Differentiation based on Resource Right Tokens

In this section, we present our differential service model, called *LIRA* (Location Independent Resource Accounting), with service profiles defined in terms of resource tokens rather than absolute amounts of bandwidth.

We consider the following simple two bits encoding scheme. The first bit, called the *preferred* bit, is set by the application or user and indicates the dropping preference of the packet. The second bit, called *marking* bit, is set by the ingress routers of an ISP and indicates whether the packet is in- or out-of profile. More precisely, when a preferred packet arrives at an ingress node, the node marks it if the user has not exceeded its profile; otherwise the packet is left unmarked[1]. The reason to use two bits instead of one is that in an Internet environment with multiple ISPs, even if a packet may be out-of profile in some ISPs on the earlier portion of its path, it may still be in-profile in a subsequent

[1] In the paper, we will use the terminology of *marked* or *unmarked* packets to refer to packets in or out-of the service profile, respectively.

ISP. Having a dropping bit that is unchanged by upstream ISPs on the path will allow downstream ISPs to make the correct decision. Core routers implement a simple behavior of priority dropping. Whenever there is a congestion, a core router always drops unmarked packets first.

In this paper, we focus on mechanisms for implementing LIRA in a single ISP. We assume the following model for the interaction of multiple ISPs: if ISP A is using the service of ISP B, then ISP B will treat ISP A just like a regular user. In particular, the traffic from all ISP A's users will be treated as a single traffic aggregate.

2.1 LIRA Service Model

With LIRA, each user i is assigned a service profile that is characterized by a resource token bucket (r_i, b_i), where r_i represents the resource token rate, and b_i represents the depth of the bucket. Unlike traditional token buckets where each preferred bit entering the network consumes exactly one token, with resource token buckets the number of tokens needed to admit a preferred bit is a dynamic function of the path it traverses.

Although there are many functions that can be used, we consider a simple case in which each link i is associated a cost, denoted $c_i(t)$, which represents the amount of resource tokens charged for sending a marked bit along the link at time t. The cost of sending a marked packet is computed as $\sum_{i \in P} L \times c_i(t)$, where L is the packet length and P is the set of links traversed by the packet. While we focus on unicast communications in this paper, we note that the cost function is also naturally applicable to the case of multicast. As we will show in Section 3, charging a user for every link it uses and using the cost in routing decisions help to increase the network throughput. In fact, it has been shown in [19] that using a similar cost function[2] for performing the shortest path routing gives the best overall results when compared with other dynamic routing algorithms.

It is important to note that the *costs* used in this paper are not monetary in nature. Instead they are reflecting the level of congestion and the resource usage along links/paths. This is different from pricing which represents the amount of payment made by an individual user. Though costs can provide valuable input to pricing policies, in general, there is *no* necessary direct connection between cost and price.

Figure 1 illustrates the algorithm performed by ingress nodes. When a preferred packet arrives at an ingress node, the node computes its cost based on the packet length and the path it traverses. If the user has enough resource tokens in its bucket to cover this cost, the packet is marked, admitted in the network, and the corresponding number of resource tokens is subtracted from the bucket account. Otherwise, depending on the policy, the packet can be either dropped, or treated as best effort. Informally, our goal at the user level is to ensure that users with "similar" communication patterns receive service (in terms of aggregate marked traffic) in proportion to their token rates.

The crux of the problem then is the computation and distribution of the per marked bit cost for each path. In this section, we first present the algorithm to compute the cost of each marked bit for a single *link*, and next present an algorithm that computes and distributes the per-path cost of one marked bit by leveraging existing routing protocols. We then argue that this dynamic cost information is also useful for multi-path routing and load balancing purposes. To avoid route oscillation and packet reordering within one application-level flow, we introduce two techniques. First, a lightweight scheme is devised to ensure that all packets from the same application-level flow always travel the same path. The scheme is lightweight in the sense that no per flow state is needed in any core routers. Second, rather than using a simple greedy algorithm that always selects the path with the current lowest cost, we use a probabilistic scheme to enhance system stability.

2.2 Link Cost Computation

A natural goal in designing the link cost function in *LIRA* is to avoid marked packets being dropped. Since in the worst case all users can compete for the same link at the same time, a sufficient condition is to have a cost function that exceeds the number of tokens in the system when the link utilization approaches unity. Without bounding the number of tokens in the system, this suggests a cost function that goes to infinity when the link utilization approaches unity. Among many possible cost functions that exhibit this property, we choose the following one:

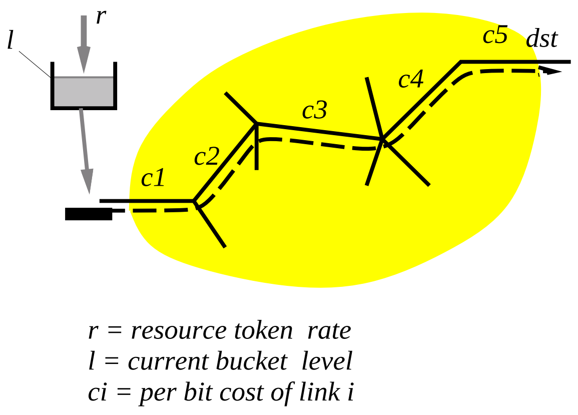
$$c(t) = \frac{a}{1 - u(t)}, \qquad (1)$$

where a is the *fixed* cost of using the link[3] when it is idle, and $u(t)$ represents the link utilization at time t. In particular, $u(t) = R(t)/C$, were $R(t)$ is the traffic throughput at time t, and C represents the link capacity. Recall that $c(t)$ is measured in tokens/bit and represents how much a user is charged for sending a marked bit along that link at time t.

In an ideal system, where costs are instantaneously distributed and the rate of the incoming traffic varies slowly, a cost function as defined by Eq. (1) guarantees that no marked packets are dropped inside the core. However, in a real system, computing and distributing the cost information incur overhead, so they are usually done periodically. In addition, there is always the issue of propagation delay. Because of these, the cost information used in admitting packets at ingress nodes may be obsolete. This may cause

[2] It can be shown that when all links have the same capacity our cost is within a constant factor from the cost of shortest-dist(P, 1) algorithm proposed in [19].

[3] In practice, the network administrator can make use of a to encourage/discourage the use of the link. Simply by changing the fixed cost a, a link will cost proportionally more or less at the same utilization.



Ingress Node Alg.

```
upon packet arrival :
  bit_cost = c1 + c2 + c3 + c4 + c5;
  packet_cost = packet_length * bit_cost;
  if (preferred(packet) && l > packet_cost)
      mark(packet);
      l −= packet_cost;
```

Figure 1. When a preferred packet arrives, the node computes the packet's cost, and the packet is marked if there are sufficient resource tokens.

packet dropping, and lead to oscillations. Though oscillations are inherent to any system in which the propagation of the feed-back information is non-zero, the sensitivity of our cost function when the link utilization approaches unity makes things worse. In this regime, an incrementally small traffic change may result in an arbitrary large cost change. In fact one may note that Eq. (1) is similar to the equation describing the delay behavior in queueing systems [18], which is known to lead to system instability when used as a congestion indication in a heavily loaded system.

To address these issues, we use the following iterative formula to compute the link cost:

$$c(t_i) = a + c(t_{i-1})\frac{\widehat{R}(t_i, t_{i-1})}{C}. \qquad (2)$$

where $\widehat{R}(t', t'')$ denotes the average bit rate of the marked traffic during the time interval $[t', t'')$. It is easy to see that if the marked traffic rate is constant and equal to R, the above iteration converges to the cost given by Eq. (1). The main advantage of using Eq. (2) over Eq. (1) is that it is more robust against large variations in the link utilization. In particular, when the link utilization approaches unity the cost increases by at most a every iteration. In addition, unlike Eq. (1), Eq. (2) is well defined even when the link is congested, i.e., $\widehat{R}(t_{i-1}, t_i) = C$.

Unfortunately, computing the cost by using Eq. (2) is not as accurate as by using Eq. (1). The link may become and remain congested for a long time before the cost increases large enough to reduce the arrival rate of marked bits. This may result in the loss of marked packets, which we try to avoid. To address this problem we use only a fraction of the link capacity, $\bar{C} = \beta C$, for the marked traffic, the remaining being used to absorb the unexpected variations due to inaccuracies in the cost estimation[4]. In this paper we chose β between 0.85 and 0.9.

[4] β is similar to the pressure factor used in some ABR congestion control schemes for estimating the fair share [14, 25].

2.3 Path Cost Computation and Distribution

In *LIRA*, the cost of a marked bit over a path is the sum of the costs of a marked bit over each link on the path. Once the cost for each link is computed, it is easy to compute and distribute the path cost by leveraging existing routing protocols. For link state algorithms, the cost of each marked bit can be included as part of the link state. For distance vector algorithms, we can pass and compute the partial path cost in the same way the distance of a partial path is computed with respect to the routing metric.

2.4 Multipath Routing and Load Balancing

Since our algorithm defines a dynamic cost function that reflects the congestion level of each link, it is natural to use this cost function for the purpose of multi-path routing. In this paper, we compute the k shortest paths for each destination or egress node using unit link metric. While the obvious solution is to send packets along the path with the minimum cost (in the sense of *LIRA*, see Section 2.1) among the k paths, this may introduce two problems: (a) packet reordering within one application-level flow, which may negatively affect end-to-end congestion control algorithms, and (b) route oscillation, which may lead to system instability.

We introduce two techniques to address these problems. First, we present a lightweight mechanism that binds a flow to a route so that all packets from the flow will traverse the same route. Second, to reduce route oscillation, for each new flow, an ingress node probabilistically binds it to one of the multiple routes. By carefully selecting the probability, we can achieve both stability and load-balancing.

2.4.1 Forwarding Algorithm For Multiple Path Routing

As discussed earlier, we will maintain multiple routes for each destination. However, we would like to ensure that all

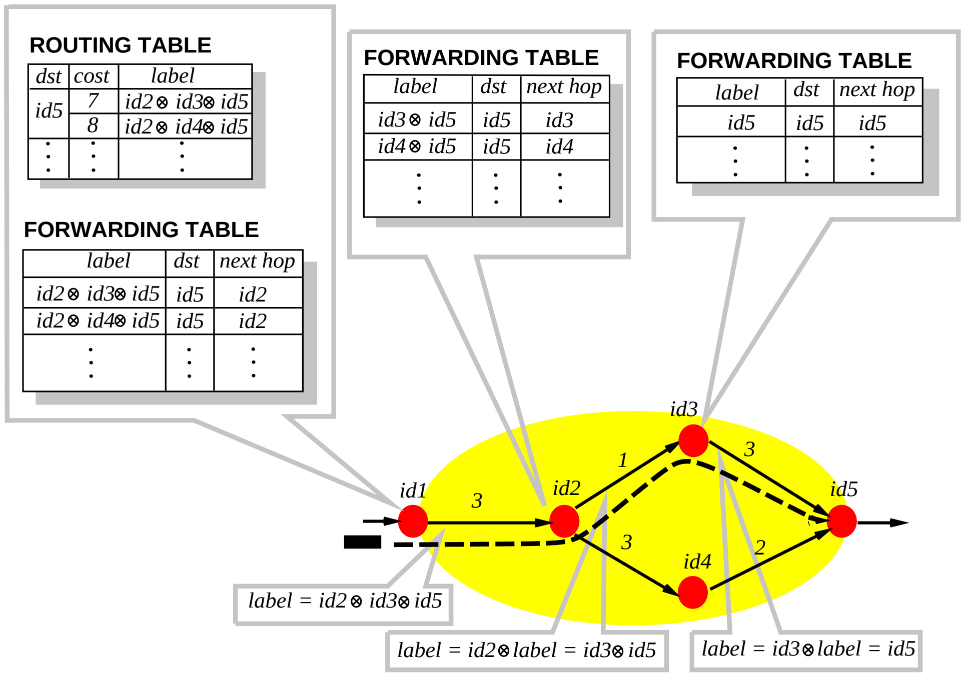


Figure 2. Example of route binding via packet labeling.

packets belonging to the same flow are forwarded along the same path.

The basic idea is to associate with each path a label computed as the XOR over the identifiers of all routers along the path, and then associate this label with each packet of a flow that goes along that path. Here we use the IP address as the identifier. More precisely, a path $P = (id_0, id_1, \ldots, id_n)$, where id_0 is the source and id_n is the destination, is encoded at the source (id_0) by $l_0 = id_1 \otimes id_2 \otimes \ldots \otimes id_n$. Similarly, the path from id_1 to id_n is encoded at id_1 by $l_1 = id_2 \otimes \ldots \otimes id_n$. A packet that travels along path P is labeled with l_0 as it is leaving id_0, and with l_1 as it is leaving d_1. By using XOR we can iteratively re-compute the label based on the packet's current label and the node identifier. As an example, consider a packet that is assigned label l_0 at node id_0. When the packet arrives at node id_1, the new label corresponding to the remaining of the path, $(id_1, \ldots, id_n)$, is computed as follows:

$$\begin{aligned} l_1 \;&=\; id_1 \otimes l_0 = \qquad (3)\\ &\quad id_1 \otimes (id_1 \otimes id_2 \otimes \ldots \otimes id_n) = id_2 \otimes \ldots \otimes id_n . \end{aligned}$$

It is easy to see that this scheme guarantees that the packet will be forwarded exactly along the path P. Here, we implicitly assume that all alternate paths between two end-nodes have unique labels. Although theoretically there is a non-zero probability that two labels may collide, we believe that for practical purposes it can be neglected.

Next we give some details of how this mechanism can be implemented by simply extending the information maintained by each router in the routing and forwarding tables. Besides the destination and the route cost, each entry in the routing table also contains the label associated with that path.

$$< dst, < cost^{(1)}, l^{(1)} >, \ldots < cost^{(k)}, l^{(k)}) >> \qquad (4)$$

Similarly, the forwarding table should contain an entry for each path:

$$< l^{(1)}, dst, next\_hop^{(1)} > \ldots < l^{(k)}, dst, next\_hop^{(k)} > \qquad (5)$$

In Figure 2 we give a simple example to illustrate this mechanism. Assume that nodes id_1 and id_5 are edge nodes, and there are two possible paths from id_1 to id_5 of costs 7, and 8, respectively. Now, assume a packet destined to id_5 arrives at id_1. First the ingress node id_1 searches the classifier table (not shown in the Figure), that maintains a list of all flows, to see whether this is the first packet of a flow. If it is, the router uses the information in the routing table to probabilistically bind the flow to a path to id_5. At the same time it labels the packet with the encoding of the selected route. In our example, assume the path of cost 7, i.e., (id_1, id_2, id_3, id_5), is selected. If the arriving packet is not the first packet of the flow, the router automatically labels the packet with the encoding of the path to which the flow is bound. This can be simply achieved by keeping a copy of the label in the classifier table. Once the packet is labeled, the router checks the forwarding table for the next hop by matching the packet's label and its destination. In our case, this operation gives us id_2 as the next hop. When the packet arrives at node id_2 the router first computes a new label based on the current packet label and the router identifier: $label = id_2 \otimes label$. The new label is then used to lookup the forwarding table.

It is important to note that the above algorithm assumes per flow state only at ingress nodes. Inside the core, there is no per flow state. Moreover, the labels can speed-up the table lookup if used as hash keys.

2.4.2 Path Selection

While the above forwarding algorithm ensures that all packets belonging to the same flow traverse the same path, there is still the issue of how to select a path for a new flow. The biggest concern with any dynamic routing protocol based on congestion information is its stability. Frequent route changes may lead to oscillations.

To address this problem, we associate a probability with each route and use it in binding a new flow to that route. The goal in computing this probability is to equalize the costs along the alternate routes, if possible. For this we use a greedy algorithm. Every time the route costs are updated we split the set of routes in two equal sets, where all the routes in one set have costs larger than the routes in the second set. If there is an odd number of routes, we leave the median out. Then, we decrease the probability of every route in the first set, the one which contains the higher cost routes, and increase the probability of each route in the second set by a small constant δ. It can be shown that in a steady-state system, this algorithm converges to the optimal solution within δ.

2.5 Algorithm Scalability

As described so far, our scheme requires to maintain k entries for each destination in both the forwarding table used by the forwarding engine and the routing table used by the routing protocol, where k is the maximum number of alternate paths. While this factor may not be significant if k is small, a more serious issue that potentially limits the scalability of the algorithm is that in the vanilla form it requires to maintain an entry for each destination, where in reality, to achieve scalability, routers really maintain the longest-prefix of a group of destinations that share the same route [11]. Since our algorithm works in the context of one ISP, we can maintain an entry for each *egress node* instead of each destination. We believe this is sufficient as the number of egress nodes in an ISP is usually not large.

However, assume that the number of egress nodes in an ISP is very large so that significant address aggregation is needed. Then we need to also perform cost aggregation. To illustrate the problem consider the example in Figure 3. Assume the addresses of d_0 and d_1 are aggregated at an intermediate router r_1. Now the question is how much to charge a packet that enters at the ingress node r_0 and has the destination d_0. Since we do

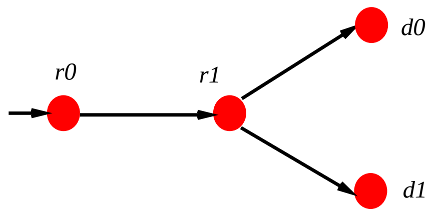


Figure 3. Topology to illustrate the label and cost aggregation.

not keep state for the individual routes to d_0, and d_1 respectively, we need to aggregate the cost to these two destinations. In doing this, a natural goal would be to maintain the total charges the same as in a reference system that keeps per route state. Let $R(r_1, d_i)$ denote the average traffic rate from r_1 to d_i, $i = 1, 2$. Then, in the reference system that maintains per route state, the total charge per time unit for the aggregate traffic from r_1 to d_0 and d_1 is: $cost(r_1, d_0)R(r_1, d_0) + cost(r_1, d_1)R(r_1, d_1)$. In a system that does not maintain per route state, the charge for the same traffic is $cost(r_1, d_0, d_1)(R(r_1, d_0) + R(r_1, d_1))$, where $cost(r_1, d_0, d_1)$ denotes the per bit aggregate cost. This yields

$$cost(r_1, d_0, d_1) = \frac{cost(r_1, d_0)R(r_1, d_0)}{R(r_1, d_0) + R(r_1, d_1)} + \frac{cost(r_1, d_1)R(r_1, d_1)}{R(r_1, d_0) + R(r_1, d_1)}. \quad (6)$$

Thus, any packet that arrives at r_0 and has either destination d_0 or d_1 is charged with $cost(r_0, r_1) + cost(r_1, d_0, d_1)$. Obviously, route aggregation increases the inaccuracies in cost estimation. However, this may be alleviated by the fact that the route aggregation usually exhibits high localities.

Another problem with address aggregation is that a label can no longer be used to encode the entire path to the destination. Instead, it is used to encode the *common* portion of the paths to the destinations in the aggregate set. This means that a packet should be relabeled at every router that performs aggregation involving the packet's destination. The most serious concern with this scheme is that it requires to maintain per flow state and perform packet classification at a core router (r_1 in our example). Fortunately, this scalability problem is alleviated by the fact that we need to keep per flow state *only* for the flows whose destination addresses are aggregated at the *current* router. Finally, we note that this problem is not specific to our scheme; any scheme that (i) allows multiple path routing, (ii) performs load balancing, and (iii) avoids packet reordering has to address it.

3 Simulation Experiments

In this section we evaluate our model by simulation. We conduct four experiments: three involving simple topolo-

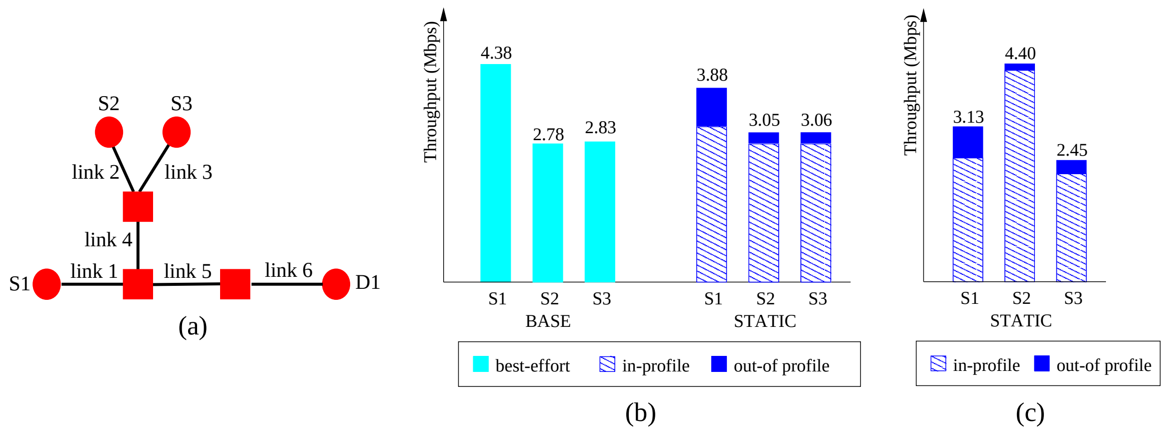


Figure 4. (a) Topology used in the first experiment. Each link has 10 Mbps capacity. $S1$, $S2$, and $S3$ send all their traffic to $D1$. (b) The throughputs of the three users under BASE and STATIC schemes. (c) The throughputs under STATIC when the token rate of $S2$ is twice the rate of $S1$/$S2$.

gies which help to gain a better understanding of the behavior of our algorithms, and one more realistic example with a larger topology and more complex traffic patterns. The first experiment shows that if all users share the same congested path, then each user receives service in proportion to its resource token rate. This is the same result one would expect from using a weighted fair queueing scheduler on every link, with the weights set to the users' token rate. In the second experiment, we show that by using dynamic routing and load balancing, we are able to achieve the same result – that is, each user to receive service in proportion to its token rate – in a more general configuration where simply using weighted fair queueing scheduler on every link is not sufficient. In the third experiment, we show how load balancing can significantly increase the overall resource utilization. Finally, the fourth experiment shows how the behaviors observed in the previous experiments scale to a larger topology.

3.1 Experiment Design

We have implemented a packet level simulator which supports both Distance Vector (DV) and Shortest Path First (SPF) routing algorithms. To support load balancing we extended these algorithms to compute the k-th shortest paths. The time interval between two route updates is uniformly distributed between 0.5 and 1.5 of the average value. As shown in [10] this choice avoids the route-update self-synchronization. In SPF, when a node receives a routing message, it first updates its routing table and then forwards the message to all its neighbors, excepting the sender. The routing messages are assumed to have high priority, so they are never lost. In the followings we compare the following schemes:

- BASE – this scheme models today's best-effort Internet, and it is used as a baseline in our comparison. The routing protocol uses the number of hops as the distance metric and it is implemented by either DV or SPF. This scheme does not implement service differentiation, i.e., both marked and unmarked packets are identically treated.

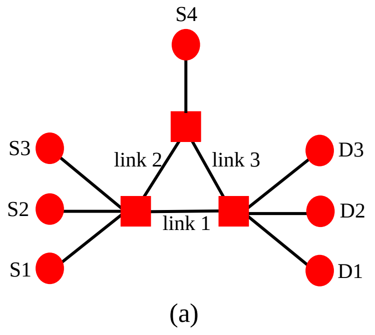
- STATIC – this scheme implements the same static routing as BASE. In addition, it implements *LIRA* by computing the link cost as described in Section 2.2, and marking packets at each ingress node according to the algorithm shown in Figure 1.

- DYNAMIC-k – this scheme adds dynamic routing and load balancing to STATIC. The routing protocol uses a modified versions of DV/SPF to find the first k shortest paths. Note that DYNAMIC-1 is equivalent to STATIC.

Each router implements a FIFO scheduling discipline with a shared buffer and a drop-tail management scheme. When the buffer occupancy exceeds a predefined threshold, newly arrived unmarked packets are dropped. Thus, the entire buffer space from the threshold up to its total size is reserved to the in-profile traffic[5]. Unless otherwise specified, throughout all our experiments we use a buffer size of 256 KB and a threshold of 64 KB.

The two main performance indices that we use in comparing the above schemes are the user *in-profile* and user *overall* throughputs. The user in-profile throughput repre-

[5] We note that this scheme is a simplified version of the RIO buffer management scheme proposed by Clark and Wroclawski [4]. In addition, RIO implements a Random Early Detection (RED) [9] dropping policy, instead of drop-tail, for both in- and out-of profile traffic. RED provide an efficient detection mechanism for the adaptive flows, such as TCP, allowing them to gracefully degrade their performances when congestion occurs. However, since in this study we are not concerned with the behavior of individual flows, for simplicity we chose to not implement RED.



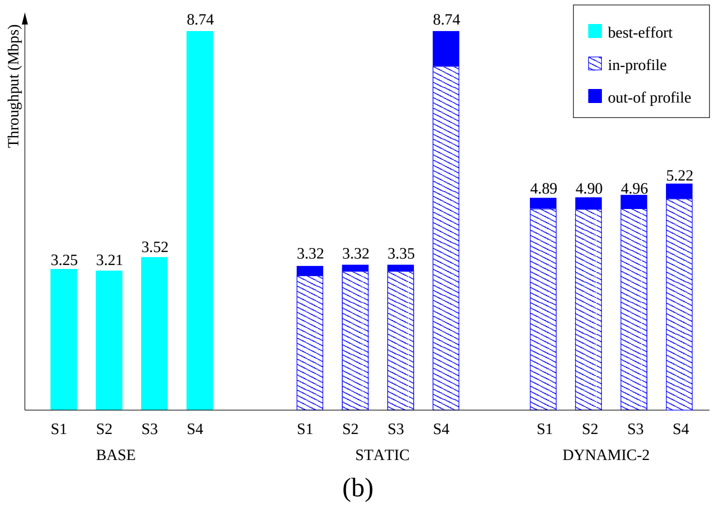


Figure 5. (a) Topology used in the second experiment. $S1$, $S2$, $S3$, and $S4$ send all their traffic to $D1$, $D2$, and $D3$, respectively. (b) The throughputs of all users under BASE, STATIC, and DYNAMIC-2.

sents the rate of the user aggregate *in-profile* traffic delivered to its destinations. The overall throughput represents the user's *entire* traffic — i.e., including both the in- and out-of profile traffic — delivered to its destinations. In addition, we use user *dropping rate* of the in-profile traffic to characterize the level of service assurance.

Recent studies have shown that the traffic in real networks exhibits the self-similar property [5, 23, 24, 30] — that is, the traffic is bursty over widely different time scales. To generate self-similar traffic we use the technique originally proposed in [30], where it was shown that the superposition of many ON-OFF flows with ON and OFF periods drawn from a heavy tail distribution, and which have fixed rates during the ON period results in self-similar traffic. In particular, in [30] it is shown that the aggregation of several hundred of ON-OFF flows is a reasonable approximation of the real end-to-end traffic observed in a LAN.

In all our experiments, we generate the traffic by drawing the length of the ON and OFF periods from a Pareto distribution with the power factor of 1.2. During the ON period a source sends packets with sizes between 100 bytes and 1000 bytes. The time to send a packet of minimum size during the ON period is assumed to be the time unit in computing the length of the ON and OFF intervals.

Due to the high overhead incurred by a packet-level simulator, such as ours, we limit the link capacities to 10 Mbps and the simulation time to 200 sec. We set the average interval between routing updates to 5 sec for the small topologies used in the first three experiments, and to 3 sec for the large topology used in the last experiment. In all experiments, the traffic starts at time $t = 20$ sec. The choice of this time is such that to guarantee that the routing algorithm finds at least one path between any two nodes by time t. In order to eliminate the transient behavior, we start our measurements at time $t = 50$ sec.

3.2 Local Fairness and Service Differentiation

This experiment shows that if all users send their traffic along the same congested path, they get service in proportion to their token rate, as long as there is enough demand. Consider the topology in Figure 4(a), where users $S1$, $S2$, and $S3$ send traffic to $D1$. Figure 4(b) shows the user overall throughputs over the entire simulation under BASE. As it can be seen, $S1$ gets significantly more than the other two. In fact, if the traffic from all sources were continuously backlogged, we expect that $S1$ to get half of the congested links 5 and 6, while $S2$ and $S3$ to split the other half. This is because even though each user sends at an average rate higher than 10 Mbs, the queues are not continuously backlogged. This is due to the bursty nature of the traffic and due to the limited buffer space at each router.

Next, we run the same simulation for the STATIC scheme. To each user we assign the same token rate, and to each link we associate the same fixed cost. Figure 4(b) shows the user overall and in-profile throughputs. Compared to BASE, the overall throughputs are more evenly distributed. However, the user $S1$ still gets slightly better service, i.e., its in-profile throughput is 3.12 Mbps, while the in-profile throughput of $S2/S3$ is 2.75 Mbps. To see why, recall from Eq. (1) that link cost accurately reflects the level of congestion on that link. Consequently, in this case links 5 and 6 will have the highest cost, followed by link 4, and then the other three links. Thus, $S2$ and $S3$ have to "pay" more than $S1$ per marked bit. Since all users have the same token rates, this translates into lower overall throughputs for $S2$ and $S3$, respectively.

To illustrate the relationship between the user's token rate and its performance, we double the token rate of $S2$.

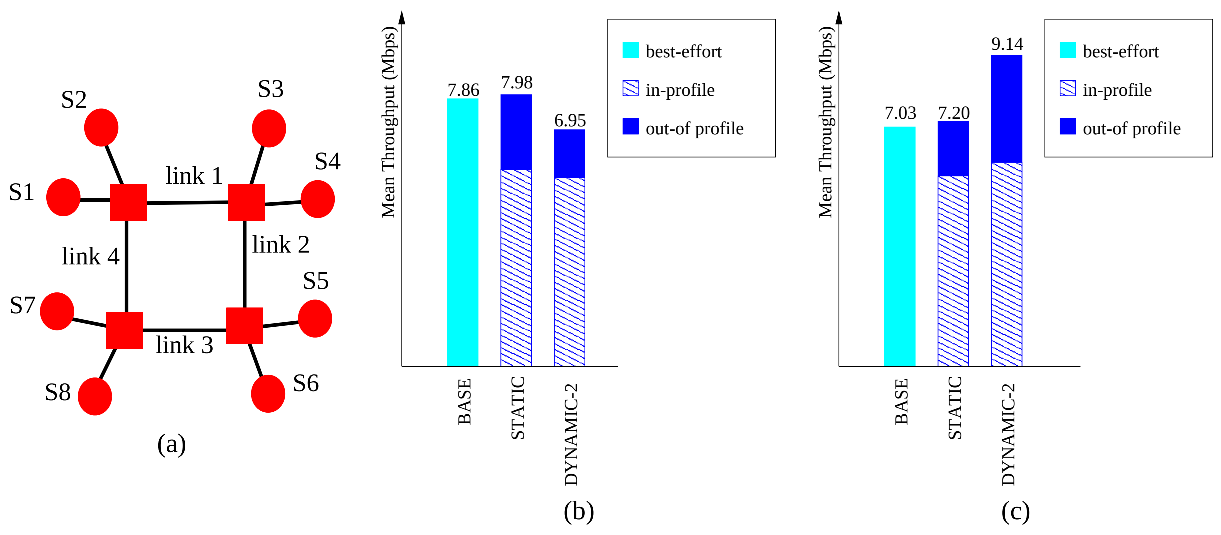


Figure 6. (a) Topology used in the third experiment. Mean throughputs when (b) load is balanced, and (c) when it is unbalanced, i.e, S3 and S4 are inactive.

Figure 4(c) shows the overall and in-profile throughputs of each user. In terms of in-profile traffic user $S2$ gets roughly twice the throughout of $S3$ (i.e., 4.27 Mbps vs. 2.18 Mbps).

Finally, we note that there was no marked packets dropped in any of the above simulations. For comparison more than 60 % of the out-of profile traffic was dropped.

3.3 User Fairness and Load Balancing

In this section we show how dynamic routing and load balancing help to improve user level fairness and achieve better resource utilization. Consider the topology in Figure 5 where users $S1$, $S2$, $S3$ and $S4$ send traffic to users $D1$, $D2$ and $D3$. Again the fixed costs of all links are equal, and all users are assigned the same token rate.

Figure 5(b) shows the overall and in-profile throughputs of $S1$, $S2$, $S3$ and $S4$ under BASE, STATIC and DYNAMIC-2, respectively. When BASE and STATIC are used, each user sends always along the shortest paths. This results in $S1$, $S2$ and $S3$ sharing link 1, while $S4$ using alone link 3. As a consequence $S4$ receives significantly better service than the other three users. Since it implements the same routing algorithm, STATIC does not improve the overall throughputs. However, compared with BASE, STATIC guarantees that in-profile packets are delivered with very high probability (again, in this experiment, no marked packets were dropped). On the other hand, when DYNAMIC-2 is used each user receives almost the same service. This is because, now users $S1$, $S2$ and $S3$ can use both routes to send their traffic, which allow them to compete with user $S4$ for link 3. User $S4$ still maintains a slightly advantage, but now the difference between its overall throughput and the overall throughputs of the other users is less than 7%. In the case of the in-profile traffic this difference is about 5%. As in the previous experiment the reason for this difference is because when competing with $S4$, the other users have to pay, besides link 3, for link 2 as well.

Thus, by taking advantage of the alternate routes, our scheme is able to achieve fairness in a more general setting. At the same time it is worth noting that the overall throughput also increases by almost 7 %. However, in this case, this is mainly due to the bursty nature of $S4$'s traffic which cannot use the entire capacity of link 3 when it is the only one to use it, rather than load balancing.

3.4 Load Distribution and Load Balancing

This experiment shows how the load distribution affects the effectiveness of our load balancing scheme. For this purpose, consider the topology in Figure 6(a). In the first simulation we generate flows that have the source and the destination uniformly distributed among users. Figure 6(b) shows the means of the overall throughputs under BASE, STATIC, and DYNAMIC-2, respectively [6]. Due to the uniformity of the traffic pattern, in this case BASE performs very well. Under STATIC we get slightly larger overall throughput, mainly due to our congestion control scheme, which admits a marked packet only if there is a high probability to be delivered. However, under DYNAMIC-2 the performances degrades. This is because there are times when our probabilistic routing algorithm selects longer routes, which leads to inefficient resource utilization.

Next, we consider an unbalanced load by making users $S3$ and $S4$ inactive. Figure 6(c) shows throughput means under BASE, STATIC, and DYNAMIC-2, respectively. As it can be noticed, using DYNAMIC-2 increases the mean by 30 %. This is because under BASE and STATIC schemes

[6] We have also computed standard deviations for each case: the largest standard deviation was 0.342 for the overall throughput under STATIC scheme, and 0.4 for the in-profile throughput under DYNAMIC-2.

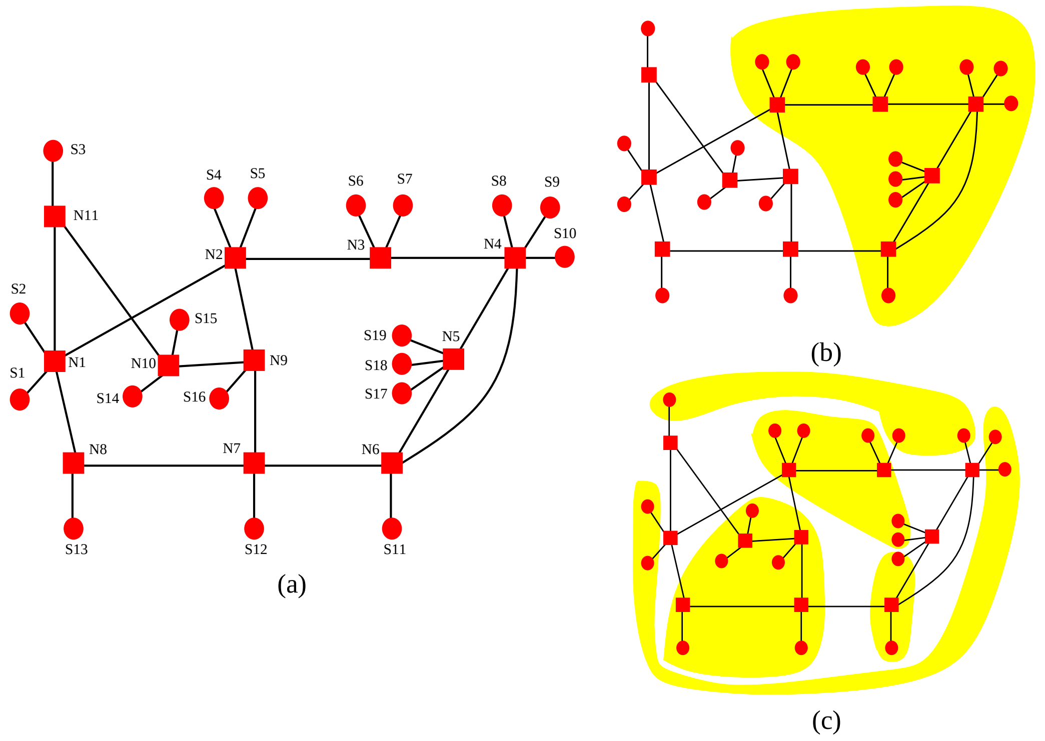


Figure 7. Topology similar to the T3 topology of the NSFNET backbone network containing the IBM NSS nodes.

the entire traffic between $S1$, $S2$ and $S5$, $S6$ is routed through links 3 and 4 only. On the other hand, DYNAMIC-2 takes advantage on the alternate route through links 1 and 2.

Finally, in another simulation not shown here we considered the scenario in which $S5$, $S6$, $S7$, and $S8$ send their entire traffic to $S3$ and $S4$, respectively. In this case DYNAMIC-2 outperforms almost two times STATIC and BASE both in terms of in-profile and overall throughputs. This is again because BASE and STATIC use exclusively links 3 and 2, while DYNAMIC-2 uses the other two links as well.

3.5 Large Scale Example

In this section we consider a larger topology that closely resembles the T3 topology of the NSFNET backbone containing the IBM NSS nodes (see Figure 7). The major difference is that in order to limit the simulation time we assume 10 Mbps links, instead of 45 Mbps. We consider the following three scenarios.

In the first scenario we assume that load is uniformly distributed, i.e., any two users communicate with the same probability. Figure 8(a) shows the results for each scheme, results which are consistent with the ones obtained in the previous experiment. Due to the congestion control which reduces the number of dropped packets in the network, STATIC achieves higher throughput than BASE. On the other hand, the dynamic routing and load balancing are not effective in this case, since they tend to generate longer routes which leads to inefficient resource utilization. This is illustrated by the decrease of the overall and the in-profile throughputs under DYNAMIC-2 and DYNAMIC-3, respectively.

In the second scenario we assume unbalanced load. More precisely, we consider 11 users (covered by the shaded area in Figure 7(b)) which are nine times more active than the other, i.e., they send/receive nine times more traffic than the others.[7] Unlike the previous scenario, in terms of overall throughputs DYNAMIC-2 outperforms STATIC by almost 8 %, and BASE by almost 20 % (see Figure 8(b)). This is because DYNAMIC-2 is able to use some of the idle links from the un-shaded partition. However, as shown by the results for DYNAMIC-3, as the number of alternate paths increases both the overall and in-profile throughputs start to decrease.

In the last scenario we consider the partition of the network shown in Figure 7(c). For simplicity, we assume that only users in the same partition communicate between them. This scenario models a virtual private network (VPN) setting, where each partition corresponds to a VPN. Again, DYNAMIC-2 performs the best[8], since it is able to make

[7] This might model the real situation where the east coast is more active between 9 and 12 a.m. EST, than the west coast.

[8] The mean of the user overall throughput under DYNAMIC-2 is 15 %

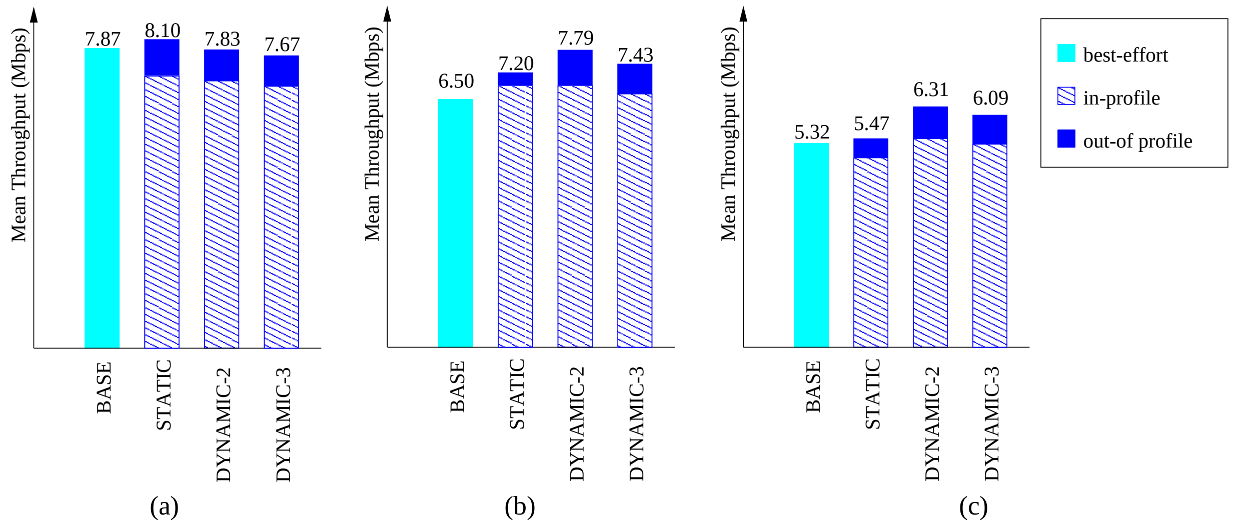


Figure 8. The throughputs when the load is balanced (Figure 7(a)), (b) unbalanced ((Figure 7(b)), and (c) when the network is virtually partitions (Figure 7(c)).

use of some links between partitions that otherwise would remain idle.

Finally, we note that across all simulations presented in this section the dropping rate for the marked packets was never larger than 0.3 %. At the same time the dropping rate for the unmarked packets was over 40 %.

3.6 Summary of Results

Although the experiments in this section are far from being exhaustive, we believe that they give a reasonable image of how our scheme performs. First, our scheme is effective in providing service differentiations at the user level. Specifically, the first two experiments show that users with similar communication patterns get service in proportion to their token rates. Second, at least for the topologies and the traffic model considered in these experiments, our scheme ensures that marked packets are delivered to the destination with high probability.

Consistent with other studies [19], these experiments show that performing dynamic routing and load balancing make little sense when the load is already balanced. In fact, doing dynamic routing and load balancing can actually hurt, since, as noted above, this will generate longer routes which may result in inefficient resource utilization. However, when the load is unbalanced, using DYNAMIC-k can significantly increase the utilization and achieve a higher degree of fairness.

Finally, we note that the in-profile dropping rate decreases as the the number of alternate paths increases. For example in the last experiment in the first two scenarios the dropping rate is no larger than 0.3 % under STATIC and 0 under DYNAMIC-2 and DYNAMIC-3, respectively, while in the last scenario the percentage decreases from 0.129 % for STATIC, to 0.101 % for DYNAMIC-2, and to 0.054 % for DYNAMIC-3.

larger than under STATIC, and 18 % larger than under BASE.

4 Discussion

In this paper, we have studied a differential service model, called *LIRA*, in which the service profile is specified in terms of resource tokens instead of absolute bandwidth. Since the exact bandwidth of marked bits that a customer can receive from such a service is not *known* a priori, a natural question to ask is why such a service model is interesting.

There are several reasons. First, we believe that the apriori specification of an absolute amount of bandwidth in the service profile, though desirable, is not essential. In particular, we believe that the essential aspects that distinguish Diffserv from Intserv are the followings: (a) the service profile is used for traffic aggregate much coarser than per flow traffic, and (b) the service profile is defined over a timescale larger than the duration of individual flows, i.e. service profile is rather *static*. Notice that the degree of traffic aggregation directly relates to the spatial granularity of the service profile. On the one hand, if each service profile is defined for only one destination, we have the smallest degree of traffic aggregation. If there are N possible egress nodes for a user, N independent service profiles need to be defined. Network provisioning is relatively easy as the entire traffic matrix between all egress and ingress nodes is known. However, if a user has a rather dynamic distribution of egress nodes for its traffic, i.e., the amount of traffic destined to each egress node varies significantly, and the number of possible egress nodes is large, such a scheme will significantly reduce the chance of statistical sharing. On the other hand, if each service profile is defined for all egress nodes, we have the largest degree of traffic aggregation. Only one service profile is needed for each user regardless the number of possible egress nodes. In addition to a smaller number of service profiles, such a service model also allows all the traffic from the same user, regardless of its destinations, to *statistically* share the same service profile. The flip side is that it makes it difficult to provision network resources. Since the traffic matrix is not known apriori, the best-case scenario is when the network traffic is evenly distributed, and the worst-case scenario is when all traffic goes to the same egress router.

Therefore, it is very difficult, if not impossible, to design service profiles that (1) are static, (2) support coarse spatial granularity, (3) are defined in terms of absolute bandwidth, and at the same time achieve (4) high service assurance and (5) high resource utilization. Since we feel that (1), (2), (4) and (5) are the most important for differential services, we decide to give up (3).

Fundamentally, we want a service profile that is static and egress node/path independent. However, to achieve high utilization, we need to explicitly address the fact that congestion is a local and dynamic phenomenon. Our solution is to have two levels of differentiation: (a) the user or service-profile level differentiation, which is based on resource token arrival rate. This is static and path independent; (b) the packet level differentiation, which is a simple priority between marked and unmarked packets and weighted fair share among marked packets. By dynamically setting the cost of each marked bit as a function of the congestion level of the path it traverses, we set up the linkage between the static/path-independent and the dynamic/path-dependent components of the service model.

A second reason for which our service model may be acceptable is that users may care more about the *differential* aspect of the service than the guaranteed bandwidth. For example, if user A pays twice as much as user B, user A would expect to have roughly twice as much traffic delivered as user B during congestion if they share same congested links, which is exactly what we accomplish in *LIRA*.

A third reason for which a fixed-resource-token-rate-variable-bandwidth service profile may be acceptable is that the user traffic is usually bursty over multiple time-scales[5, 23, 30]. Thus, there is a fundamental mismatch between an absolute bandwidth profile and the bursty nature of the traffic[9].

We do recognize the fact that it is desirable for both the user and the ISP to understand the relationship between the user's resource token rate and its expected capacity. This can be achieved by measuring the rate of marked bits given a fixed token rate. Both the user and the ISP can perform this measurement. In fact, this suggests two possible scenarios in which *LIRA* can be used to provide a differential service with an expected capacity defined in terms of absolute bandwidth. In the first scenario, the service is not transparent. Initially, the ISP will provide the user with the following relationship

$$expected\_capacity = f(token\_rate, traffic\_mix) \qquad (7)$$

based on its own prior measurement. The user will measure the expected capacity and then make adjustments by asking for an increase or a decrease in its resource token rate. In the second scenario, the service is transparent. Both the initial setting and the subsequent adjustments of the service profile in terms of number of token rate will be made by the ISP only.

Therefore, one way of thinking about our scheme is that it provides a flexible and efficient framework for implementing a variety of Assured Services. In addition, the dynamic link cost information and the statistics of the resource token bucket history provide good feedback both for individual applications to perform runtime adaptation, and for the user or the ISP to do proper accounting and provisioning.

5 Related Work

Our work is highly influenced by Clark and Wroclawski's Assured Service proposal [3, 4]. The key difference is that we define service profiles in units of resource tokens rather than absolute bandwidth. In addition, we propose a resource accounting scheme and an integrated set of algorithms to implement our service model.

Another related proposal is the User-Share Differentiation (USD) [29] scheme, which does *not* assume absolute bandwidth profiles either. In fact, with USD, a user is assigned a share rather than a token-bucket-based service profile. For *each* congested link in the network traversed by the user's traffic, the user shares the bandwidth with other users in proportion to its share. The service provided is equivalent to one in which *each* link in a network implements a weighted fair queueing scheduler where the weight is the user's share. With USD, there is little correlation between the share of a user and the aggregate throughput it will receive. For example, two users that are assigned the same share can see drastically different aggregate throughputs. A user that has traffic for many destinations (thus traverse many different paths) can potentially receive much higher aggregate throughput than a user that has traffic for only a few destinations.

There is a huge body of related work that addresses the resource allocation problem both for single and multiple resources. However, to the best of our knowledge, none of

[9] Some recent measurements show that the aggregate traffic over Internet backbone links are not very bursty. We note that this is not inconsistent with the observations that the aggregate traffic from a campus to the Internet exhibits long range dependency.

the existing proposals address the problem of allocating resources for traffic aggregate that has a large spatial granularity. In general, they are limited in scope to allocating resources along individual paths only. In addition, these schemes usually require each user to maintain per resource state, or/and each resource to maintain per user state. In the following, we discuss several of the more relevant schemes.

Waldspurger and Weihl have proposed a framework for resource management based on lottery tickets [27, 28]. Each client is associated a certain number of tickets which encapsulate its resource rights. The number of tickets a user receives is similar to the user's income rate in *LIRA*. This framework was shown to provide flexible management for various single resources, such as disk, memory and CPU. However, they do not give any algorithm(s) to coordinate tickets allocation among multiple resources.

Ferguson et al. proposed a flow control economy to allocate network resources such as links and buffers among competing virtual circuits (VCs) [7, 8]. In this model, each VC is endowed certain funds for buying resources. The VC's goal is to buy a minimum capacity on all links along its path, and use the extra money to minimize the average end-to-end delay. The resource prices are set so that the supply and the demand are balanced. It has been shown that such economy converges and the resulted allocations are pareto-optimal.

MacKie-Mason and Varian have proposed a model, called "smart markets", in which each packet carries a bid that represents how much the user is willing to pay for it [20]. At each congested link along a path a cutoff price is computed and only packets that have a higher bid are forwarded; the other packets are buffered. At the service level it is unclear how the priorities of individual packets translate into expected network throughput. In addition, in order to achieve high level of service assurance, a user needs to know the smallest bid along the path. No mechanisms are given to propagate this information to the users.

Awerbuch et al. [1] have proposed an on-line reservation algorithm to maximize the throughput in a network where the *duration* of each reservation is known in advance. The algorithm guarantees that the throughput is within $O(\log nT)$ factor of the throughput achieved by an optimal off-line algorithm, where n is the number of nodes and T is the maximum duration of a reservation. In the scheme, each link is associated with a cost that is a exponential function of its current utilization. Also, each connection is associated with a profit which is received only if the request is granted. The goal of the algorithm is then to maximize the overall profit. While this algorithm differs significantly form ours both in assumptions and goals, we note that the mechanisms used to implement *LIRA* can also be used to implement this scheme.

Kelly et. al [15, 16] have considered the problem of bandwidth allocation between competing streams with elastic traffic. In particular, they propose a mathematical model to analyze the stability and fairness of a class of rate controlled algorithms. In this model each user chose the charge per unit of time that it is willing to pay for a route. In turn the network computes the user rates according to a proportionate criterion. However, they only consider the model where resources are allocated on the basis of per virtual circuit.

To increase resource utilization, in this paper we propose performing dynamic routing and load balancing among the best k shortest paths between source and destination. In this context, one of the first dynamic routing algorithms, which uses the link delay as metric, was the ARPANET shortest path first [21]. Unfortunately, the sensitivity of this metric when the link utilization approaches unity resulted to relative poor performances. Various routing algorithms based on congestion control information were proposed elsewhere [12, 13]. The unique aspect of our algorithm is that it combines dynamic routing, congestion control and load balancing together. Also we alleviate the problem of system stability which plagued many of the previous dynamic routing algorithms by defining a more robust cost function and probabilistically binding a flow to a route. We also note that our link cost is similar to the one used in [19]. In particular, it can be shown that when all links have the same capacity, our link cost is within a constant factor of the cost of shortest-dist(P, 1) algorithm presented in [19]. It is worth noting that shortest-dist(P, 1) performed the best among all the algorithms studied in [19].

6 Summary

We study models and algorithms that support Assured Service with service profiles defined over large spatial granularities. We propose a service model in which the service-profile is defined in units of resource tokens rather than the absolute bandwidth, and an accounting scheme that dynamically determines the number of resource tokens charged for each in-profile packet. We present a set of algorithms that efficiently implement the service model. In particular, we introduce three techniques: (a) distributing path costs to all edge nodes by leveraging existing routing infrastructure; (b) binding a flow to a route (route-pinning) without maintaining per flow state; (c) multi-path routing and probabilistic binding of flows to paths to achieve load balancing. Simulation results are presented to demonstrate the effectiveness of the approach. To the best of our knowledge, this is the first complete scheme that explicitly addresses the issue of large spatial granularities.

While these techniques are developed in the context of supporting Assured Service, they may be useful in other contexts. For example, by combining the route-pinning technique with the SCED+ algorithm proposed in [6], guaranteed or premium service can be provided without the need for per flow management at core routers.

As future work, we plan to extend this work to support multiple ISP environments, multicast communication, and both sender and receiver-based charging schemes.

References

[1] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proceedings of FOCS'93*, October 1993.

[2] S. Blake. Some issues and applications of packet marking for differentiated services, July 1997. Internet Draft.

[3] D. Clark. Internet cost allocation and pricing. *Internet Economics, L. W. McKnight and J. P. Bailey (eds.)*, pages 215–252, 1997.

[4] D. Clark and J. Wroclawski. An approach to service allocation in the internet, July 1997. Internet Draft.

[5] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic evidence and possible causes. In *Proceedings of the ACM SIGMETRICS 96*, pages 160–169, Philadelphia, PA, May 1996.

[6] R. L. Cruz. Sced+: Efficient Management of Quality of Service Guarantees. In *Proceedings of INFOCOM'98*, pages 625–642, San Francisco, CA, 1998.

[7] D. F. Ferguson. *The Application of Microeconomics to the design of Resource Allocation and Control Algorithms in Distributed Systems*. PhD thesis, 1989.

[8] D. F. Ferguson, C. K. Nikolau, and Y. Yemini. An economy flow control in computer networks. In *Proceedings of INFOCOM'90*, 1990.

[9] S. Floyd and V. Jacobson. Random early detection for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, July 1993.

[10] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. In *Proceedings of ACM SIGCOMM'93*, pages 33–44, San Francisco, CA, September 1993.

[11] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless interdomain routing (cidr): An address assignment and aggregation strategy (rfc 1519), September 1993.

[12] D. W. Glazer and C. Tropper. A new metric for dynamic routing algorithms. *IEEE Transaction on Communication*, 38(3):360–367, March 1990.

[13] S. J. Golestani. *A Unified Theory of Flow Control and Routing in Communication Networks*. PhD thesis, Department of Computer Science, University of Virginia, 1980.

[14] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan. Erica switch algorithm: A complete description, August 1996. ATM Forum/96-1172.

[15] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecomunications*, 8:33–37, 1997.

[16] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 40, June 1998.

[17] K. Kilkki. Simple integrated media access (sima), June 1997. Internet Draft.

[18] L. Kleinrock. *Queueing Systems*. John Wiley and Sons, 1975.

[19] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. In *Proceedings of the ACM SIGCOMM'96*, pages 206–217, Palo Alto, CA, October 1996.

[20] J. MacKie-Mason and H. Varian. Pricing the interenet. *Public Access to the Internet, B. Kahin and J. Keller (ed.)*, 1994.

[21] J. M. McQuillan, I. Richer, and E. Rosen. The new routing algorithm for the arpanet. *IEEE Transaction on Communication*, 28(5):711–719, May 1980.

[22] K. Nichols, V. Jacobson, and L. Zhang. An approach to service allocation in the internet, November 1997. Internet Draft.

[23] V. Paxon and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[24] V. Paxon and S. Floyd. Why we don't know how to simulate the internet. In *Proceedings of the Winder Communication Conference*, December 1997.

[25] L. Roberts. Enhanced prca (proportionat rate control algorithm), August 1994. ATM Forum/94-0735R1.

[26] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service (rfc 2212), September 1997.

[27] C. A. Waldspurge. *Lottery and Stride Scheduling: Flexible Proportional -Share Resource Management*. PhD thesis, 1995.

[28] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of OSDI 94*, pages 1–12, November 1994.

[29] Z. Wang. User-share differentiation (usd) scalable bandwidth allocation for differentiated services, May 1998. Internet Draft.

[30] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. In *Proceedings of the ACM SIGCOMM'95*, pages 100–113, Boston, MA, August 1995.

[31] J. Wroclawski. Specification of the controlled-load network element service (rfc 2211), September 1997.

[32] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Communications Magazine*, 31(9):8–18, September 1993.