On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing

Reuven Cohen Gabi Nakibli
Dept. of Computer Sciences
Technion
Israel

Abstract— **In this paper we study the computational complexity and effectiveness of a concept we term "N-hub Shortest-Path Routing" in IP networks. N-hub Shortest-Path Routing allows the ingress node of a routing domain to determine up to N intermediate nodes ("hubs") through which a packet will pass before reaching its final destination. This facilitates better utilization of the network resources, while allowing the network routers to continue to employ the simple and well-known shortest-path routing paradigm. Although this concept has been proposed in the past, this paper is the first to investigate it in depth. We apply N-hub Shortest-Path Routing to the problem of minimizing the maximum load in the network. We show that the resulting routing problem is NP-complete and hard to approximate. However, we propose efficient algorithms for solving it both in the online and the offline contexts. Our results show that N-hub Shortest-Path Routing can increase network utilization significantly even for $N = 1$. Hence, this routing paradigm should be considered as a powerful mechanism for future datagram routing in the Internet.**

I. Introduction

Intra-AS routing in the Internet is based on the hop-by-hop shortest-path paradigm. The source of a packet specifies the destination address, and each router along the route forwards the packet to a neighbor located "closest" to the destination. Since the routing is usually static, i.e., the cost of a path is dependent on the network topologies rather than on the dynamics of the network traffic, a single route is used for every source-destination pair.

The shortest-path routing paradigm is known to be simple and efficient. It does not place a heavy processing burden on the routers and usually requires at most one entry per destination network in every router. However, while this scheme finds the shortest path for each pair of nodes and thus minimizes the bandwidth consumed by every packet, it does not guarantee full utilization of the network resources under high traffic loads. When the network load is not uniformly distributed, some of the routers introduce an excessive delay while others are underutilized. In some cases this non-optimized use of network resources may introduce not only excessive delays but also incur a high packet loss rate.

Much research has been conducted in a search for an alternative routing paradigm that would address this drawback of shortest-path routing. The sought paradigm should utilize the network resources more efficiently and minimize the probability of congestion, thereby achieving better delay-throughput behavior than traditional shortest-path routing. In addition, such a scheme should be practical in terms of the volume of control information exchanged by the routers, the memory requirement, the processing burden imposed by every packet, and so forth. Finally, such a scheme should interoperate seamlessly with network routers that continue to employ the shortest-path routing paradigm.

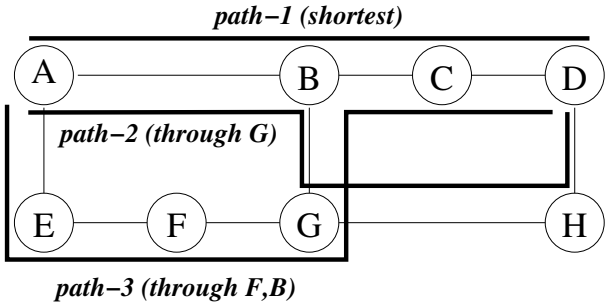


Fig. 1. An example of N-hub routing

Most of the routing schemes proposed in the past are able to employ more than one path between every source-destination pair. Generally, these schemes base their routing decisions on the load imposed on every network link. When a particular link, or an area, becomes congested, some of the routes are modified. Some routing schemes find an alternate data path only when the standard path is highly congested [1]. In [2]–[4], alternate routes are found for every source-destination pair even if the standard route is not heavily loaded. Several loop-free paths are found in advance and the load is distributed between them. However, due to the complexity of these schemes, their increased processing burden, and their considerable deviation from the conventional shortest-path routing paradigm, not one of them has been adopted for the Internet. A major drawback of many proposed routing schemes is that they must be deployed over the lion's share of the routing domain in order to be effective.

This paper investigates a routing scheme that takes advantage of a concept we refer to as "N-hub Shortest-Path Routing," or simply N-hub routing. This concept can be implemented using several existing IP mechanisms, as will be discussed in Section II. N-hub routing allows the ingress router of a routing domain to determine one or more intermediate nodes ("hubs") that a packet will traverse before reaching its final destination. Fig. 1 illustrates this concept. The figure shows three paths for a packet whose source and destination

nodes are A and D. The first path, path-1, is the shortest path. Path-2 uses node G as a single hub. Packets are routed first on the shortest path from A to G and then on the shortest path from G to D. Such a route is likely to improve the throughput if the links $B-C$ or $C-D$ are heavily loaded while the links $B-G$, $G-H$ and $H-D$ are underutilized. Path-3 uses 2 hubs: F and B. Packets are routed first on the shortest path from A to F, then on the shortest path from F to B, and finally on the shortest path from B to D. It is evident from the example above that N-hub routing is a generalization of shortest-path routing, because shortest-path routing is equivalent to N-hub routing with $N = 0$.

Using the concept of N-hub routing, the routing protocol gains better control over the routing process, while the network routers continue to employ the shortest-path paradigm for building their routing tables. Although this concept is not employed today in the Internet, we think it is a powerful tool that should be investigated in the context of traffic engineering and QoS.

It is important to note the practical benefits of N-hub Shortest-Path Routing over virtual-circuit routing. First, N-hub routing can be implemented in networks that usually do not employ virtual-circuit routing technologies (such as MPLS [5]). In particular, it can be implemented in sensor networks and ad hoc (mobile) networks. Second, when virtual-circuit routing is used, only one or two routes are usually established between every two routers. Therefore, it is not possible to react to changes in the traffic pattern before the time-consuming and labor-intensive building of new routes. In contrast, an N-hub route can be changed immediately according to changes in the link loads, without having to set up additional routes in advance. Third, N-hub routing imposes additional processing and memory burden on the hubs and the source edge routers only, while the other nodes employ regular shortest-path routing. In virtual-circuit routing this burden is imposed on all the nodes along the path. This is especially significant when each node has to maintain several thousands of explicit routes.

The ingress router of a routing domain should be responsible for determining the intermediate router(s) through which the packets of each flow will be routed. To this end, the router may use information it acquires regarding the load distribution in the network by means of a link-state flooding protocol like OSPF-TE [6]. For a typical case scenario for N-hub routing in an ISP AS, consider a DiffServ [7] domain, which supports the Expedited Forwarding (EF) Per Hop Behavior. When an edge router receives a packet of an EF flow (e.g., a Voice over IP flow), and N-hub routing is not supported, the router has no option but to forward the packet along the default (shortest) path or to drop it. With N-hub routing support, however, the edge router uses information about the load distribution in the entire domain, as can be obtained using OSPF-TE [6], in order to determine the hub(s) that define the least congested route. This list of hub(s) is added to the packet, and is also kept in the router's local flow table. When subsequent packets of the same flow are received by this router, it identifies them as belonging to the same flow, e.g., using the flow label of IPv6, and fetches from its table the list of hub(s) associated with this flow. Once every time-out period, the router checks if there is a better N-hub route that can be used by the considered flow.

To the best of our knowledge, this paper is the first to propose a thorough theoretical and practical investigation of N-hub Shortest-Path Routing. The contribution of this paper is fourfold. First, we define the N-hub shortest-path problem as an optimization problem, and show that from a computational complexity perspective, N-hub is "closer" to virtual circuit ($|V|$-hub) routing than to shortest path (0-hub) routing. This is because N-hub is NP-complete, and it has no polynomial approximation scheme (PTAS). Second, we develop a probabilistic approximation algorithm for the N-hub problem. Third, we show that online algorithms originally designed for multicommodity routing maintain their competitive ratio for N-hub routing. Fourth, we show that in practice, one hub for every flow is sufficient to obtain results that are almost equal to those obtained by optimal algorithms for the splittable multicommodity flow problem. These results are upper bounds for the results that can be obtained by optimal algorithms for virtual circuit routing.

The rest of this paper is organized as follows. In Section II we discuss related work and the various mechanisms that can be employed to implement N-hub routing. In Section III we define the N-hub routing problem and review its computational complexity. In Section IV we present several approximation algorithms for the online context. The competitive ratio of these algorithms is discussed, and one of them is shown to have the best competitive ratio that can be obtained for this problem. In Section V we present simulation results that show the potential effectiveness of N-hub routing in general, and the effectiveness of the various algorithms proposed in the paper. Finally, Section VI concludes the paper.

II. N-hub Shortest-Path Routing in IP Networks: Implementation and Related Work

N-hub Shortest-Path Routing can be implemented using several existing mechanisms. A straightforward way is to take advantage of the IPv4 Loose Source-Routing option [8]. When this option is used, the IP header is extended by a list of the addresses of the intermediate node(s) the packet must traverse. However, this option, much like any other IPv4 option, is rarely used, mainly because of the heavy processing burden imposed on the general purpose CPU of the router when an IPv4 header contains any optional field. Moreover, there are some notable security issues related to this option [9]. In [10], it is noted that only 8% of Internet routers are source-routing capable.

As opposed to IPv4, IPv6 [11] has a more "built-in" support for N-hub routing. The primary header of an IPv6 packet can be followed by flexible extension headers. These headers can, for example, indicate the IP addresses of the network routers the packet should traverse en route to its destination.

Another way to implement N-hub routing in IPv4 is to use IP-in-IP encapsulation [12]. In this case, an IP header indicating the final destination is encapsulated in the payload of another IP header. The latter header contains, in its destination address field, the IP address of an intermediate router. The total

number of headers is therefore equal to the number of hubs plus 1.

N-hub routing can also be implemented through an overlay network [13]. In an overlay network the source sends a packet to the first hub, while adding to its payload information that identifies the next hubs and the final destination. Each hub uses this information to route the packet to the next hub.

Another powerful way to implement the N-hub routing paradigm is to use MPLS [5]. MPLS is a virtual circuit technology that allows an MPLS ingress node to set up a tunnel over the shortest path or over an explicit path to an egress node. An explicit path contains a list of intermediate nodes. The route between two consecutive nodes in the list is either strict or loose. A loose route may contain other nodes. Therefore, N-hub shortest path routing can be viewed as a special case of the MPLS explicit route option. With respect to MPLS, our results imply that an explicit strict route need not be specified. Rather, it is sufficient for the ingress MPLS node to include a single loose node in the RSVP-TE Path message. If the tunnel should be established over the route whose maximum load is minimized, the routing algorithms we propose can be used.

We are not aware of any work that addresses the computational complexity and the potential effectiveness of N-hub Shortest-Path Routing, which is the core of this paper. Several routing schemes that are similar in one way or another to ours have been leveraged in other works, e.g., [13]–[15], but their focus is entirely different. In [16], [17], the authors present a multi-path routing scheme called "two-phase routing." In this scheme, traffic originating at a source node is routed over a set of routes in predetermined and static proportions. Each route is diverted from the source to an intermediate node before reaching the destination. This approach is shown to provide load balancing and bandwidth efficiency even with highly variable traffic. In [18] the authors explore the deployment of this routing scheme in optical networks, in order to increase routing resiliency. In [19] the authors study the throughput performance of that routing scheme. Our paper[1] investigates the effectiveness and computational complexity of the general form of two-phase routing. Furthermore, we consider non-static routing in which intermediate nodes are determined according to current traffic conditions, while addressing the online setting of the problem.

In [20], the authors investigate the effectiveness of selfish routing in Internet-like environments. Selfish routing allows the host to determine the path according to a criterion that maximizes its profit. This work specifically addresses a setting where sources choose N-hub routes in an overlay network. Their main conclusion is that selfish hosts can achieve results similar to those achieved by routing with full control. There are two notable differences between [20] and our work. First, in our model, the host chooses routes that do not necessarily maximize its profit. Second, [20] assumes absolute knowledge of flow demands that do not change over time, while we deal with the more practical online scenario where flow demands are not known in advance.

As already said, the main benefit gained from determining more intermediate nodes (hubs) for a route between a source-destination pair is better control over network load distribution, with little deviation from the traditional shortest-path routing paradigm. More specifically, the routers continue building their routing tables using the shortest-path information they acquire through a conventional routing protocol. However, the network is capable of routing a packet over less congested areas. Moreover, it can be employed effectively even if a small fraction of the network routers support it. This is because traffic can be diverted to less congested areas without the support of the core routers.

The trade-off between the simplicity of traditional datagram (shortest-path) routing and the efficiency of virtual-circuit routing is well known. However, both schemes can be viewed as special cases of N-hub routing: with $N = 0$ for shortest-path routing and $N = |V|$ for virtual circuit routing. Hence, N-hub routing, where $0 \leq N \leq |V|$, offers a compromise between these two extremes (see Fig. 2). As the number of allowed hubs grows, the number of possible routes between each source-destination pair increases, and the flexibility/efficiency of the routing scheme increases as well. However, we pay for the increased efficiency by sacrificing some of the inherent simplicity of shortest-path routing at each hub. In practice, as shown in Section V, the performance achieved with a single hub is very close to the optimal performance of virtual-circuit routing. Hence, 1-hub routing can be viewed as a routing protocol that offers the performance of virtual-circuit routing with only slight deviation from traditional shortest-path routing.

III. Problem Definition and Complexity

A. Problem Definition

In this paper we focus on applying the N-hub Shortest-Path Routing paradigm to a traffic engineering task. Our specific aim is to minimize the maximum load in the network. We deal with the routing problem of minimizing the maximum load imposed on a single link by determining up to N intermediate nodes through which the packets of each flow will be routed. Note that we do not assume any constraint regarding the criteria used for classifying packets to flows.

A similar objective – minimizing the maximum load imposed on a single link – was addressed in the past mainly in the context of the multicommodity flow problem [21], [22] and the Virtual Circuit Routing problem [23]–[25]. Maximizing the load on a single link does not always guarantee perfect load balancing and minimum average delay. However, it was shown in the past to yield good performance because the delay on a link grows exponentially with the load. Moreover, this objective is easier to analyze from a theoretical point of view. As a counter-example, consider the topology in Fig. 3 and suppose there are 3 flows as follows:

1) A flow from node A to node E, with a bandwidth demand of 1.
2) A flow from node A to node B, with a bandwidth demand of 2.

[1]An early version of our paper, published in Infocom 2004, predates Ref. [16]–[19]

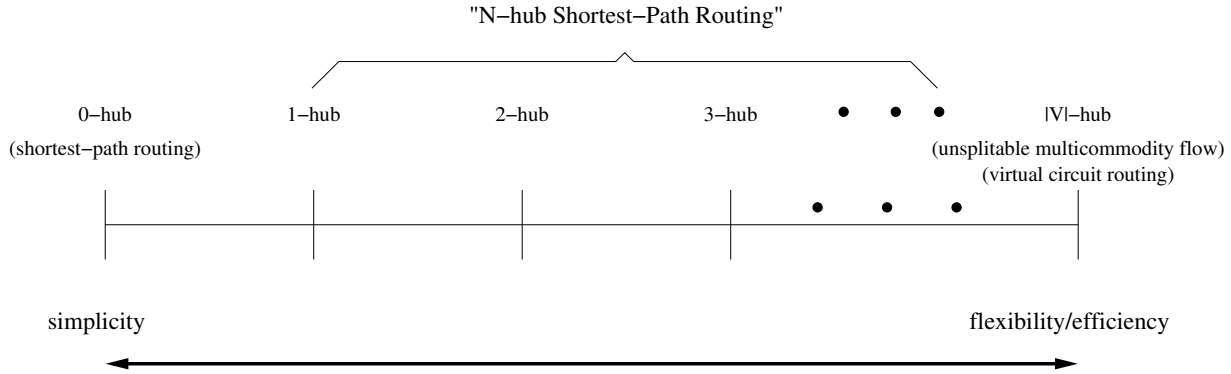


Fig. 2. N-hub routing as a compromise between efficiency and simplicity

3) A flow from node B to node E, with a bandwidth demand of 1.

An algorithm that minimizes the maximum load may produce a solution that routes flows 1 and 3 via node C. This solution yields a greater delay of the packets of flow 1 and flow 3 than a solution obtained by an algorithm that tries to minimize the average delay. The latter solution might route flow 1 through router C and flow 3 through router D.

One may consider the average load over all the edges in the graph as a better objective for minimizing the average delay of the packets. However, this objective is achieved with static shortest-path routing which, as mentioned above, is known to be inefficient for non-uniform traffic patterns in which some areas in the AS are more congested than others. Another possible objective is minimizing the variance of the loads on the network links. However, this objective does not take into account the actual load on the links. It may therefore yield very long and possibly non-simple routes in order to ensure that all the links will be equally utilized.

In our model the network is represented by a directed graph. The routers in the network are represented by the vertices of the graph and the links by the edges. The bandwidth of a link is represented by the capacity of the corresponding edge. The source and destination of each flow are represented by their edge routers. For every flow there is a traffic demand.

We now give a formal definition of the N-hub routing problem. Let $G = (V, E)$ be a directed graph. Each edge, $e \in E$, has a capacity $u(e)$, where $u : E \rightarrow \mathbf{R}^+$. Let $\mathcal{F} \subseteq V \times V$ be a set of flows between pairs of source and destination nodes. Each flow $f \in \mathcal{F}$ has a traffic requirement $T(f)$, where $T : \mathcal{F} \rightarrow \mathbf{R}^+$. Let s^f and d^f denote the source and destination of flow f respectively. For each flow $f \in \mathcal{F}$, find an ordered sequence of N hubs, denoted by $h_1^f, h_2^f, \ldots, h_N^f$, where $h_i^f \in V$, such that the packets of f are routed over $s^f \rightarrow h_1^f \rightarrow h_2^f \rightarrow \ldots \rightarrow h_N^f \rightarrow d^f$, where $a \rightarrow b$ denotes the shortest path from node a to node b on G, and the maximum relative load imposed on every edge in E is minimized. The relative load on edge e is defined as $\frac{\sum_{f \mid e \in P^f} T(f)}{u(e)}$, where P^f is the path chosen to route flow f.

In Appendix II we prove that the N-hub routing problem is NP-complete.

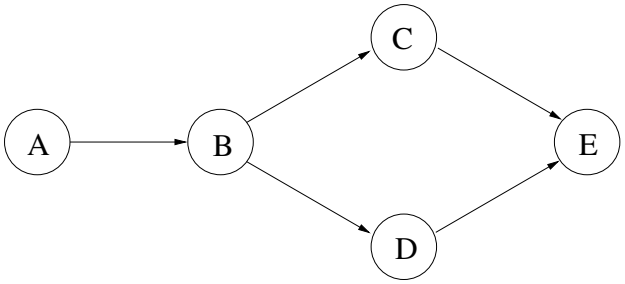


Fig. 3. An example of a network topology

B. On the Approximation Hardness of N-hub

One common way to get around an NP-complete problem is to develop a polynomial time algorithm that finds a near-optimal solution for the problem, namely an approximation algorithm. Usually, when the worst-case performance of an approximation algorithm is bounded, the average-case performance is very close to the optimum.

Algorithm A is an approximation algorithm for an optimization problem Π if for any input I it runs in polynomial time in the length of I and outputs a feasible solution $A(I)$ for the problem. In the context of N-hub, a feasible solution is a solution where the route between each source-destination pair traverses at most N hubs, while the route between two consecutive hubs is the shortest path between them. An algorithm A for a minimization problem (like N-hub, where we seek to minimize the maximum load) is said to have an approximation ratio of ρ, if for any input I, value(A(I))/value(OPT(I))$\leq \rho$.

An algorithm A for a minimization problem is an approximation scheme for Π if it takes as an input not only the instance I of the problem, but also a value $\epsilon > 0$ such that for any fixed ϵ, value(A(I))/value(OPT(I))$\leq 1 + \epsilon$. An approximation scheme A is said to be a polynomial time approximation scheme (PTAS) [26] if for each fixed ϵ there is a polynomial approximation algorithm derived from A with an approximation ratio of $1 + \epsilon$. If the running time of the approximation algorithm is also polynomial in the value of $1/\epsilon$, then A is said to be a fully polynomial approximation scheme (FPTAS) [26].

It can be easily shown that there is no FPTAS for N-hub unless $\mathcal{P}$=$\mathcal{NP}$. However, in what follows we show a stronger

inapproximability result.

Definition 1: Let Π be a minimization problem. The decision problem Π_K is the problem of deciding for a given instance I whether the optimum value of $\Pi_K(I) \leq K$.

Corollary 1: Unless $\mathcal{P}$=$\mathcal{NP}$, N-hub does not permit a PTAS and cannot be approximated within $2-\epsilon$ for $\epsilon > 0$.

Proof: Let Π be an integer minimization problem. Suppose that the decision problem Π_K is NP-hard for some constant K. Then, from [26] we know that unless $\mathcal{P}$=$\mathcal{NP}$, there is no PTAS for Π and there is no polynomial algorithm with an approximation ratio that is strictly less than $1 + 1/K$. Consider the *Integer N-hub* problem defined earlier. Obviously, in a feasible solution of this problem the maximum load has an integer value equal to 1 or more. However, by Corollary 3 (in Appendix II), the problem of deciding whether the optimum value of Integer N-hub is equal to 1 is also NP-hard. Hence, Integer N-hub, and subsequently N-hub, does not permit a PTAS and cannot be approximated within $2-\epsilon$ for any $\epsilon > 0$. ■

Appendix III presents an asymptotic PTAS for the problem. The algorithm gives an approximation factor that decreases as the congestion in the network increases.

IV. Online Approximation Algorithms

We now consider the more practical online version of N-hub, where routing decisions for the flows are performed one at a time without prior knowledge of future flows. We consider three online approximation algorithms, originally developed for the unsplittable multicommodity flow problem [26]. We slightly modify these algorithms in order to apply them to the N-hub Shortest-Path routing problem. We prove that their competitive ratios for the unsplittable multicommodity flow problem is the same as for the N-hub Shortest-Path routing problem. The competitive ratio of an online algorithm is defined as the worst case ratio, over all sequences of flows, between the value of the solution found by the algorithm and the value of the solution found by an optimal offline algorithm. See [27] for further details.

For the sake of completeness we give a formal definition of the unsplittable multicommodity flow problem. Let $G = (V, E)$ be a directed graph. Each edge, $e \in E$, has a capacity of $u(e)$, where $u : E \rightarrow \mathbf{R}^+$. Let $\mathcal{F} \subseteq V \times V$ be an ordered set of flows between pairs of source and destination nodes. Each flow $f \in \mathcal{F}$ has a traffic requirement $T(f)$, where $T : \mathcal{F} \rightarrow \mathbf{R}^+$. Route every flow $f \in \mathcal{F}$, in the order the flows are received, on a single arbitrary route in G, while minimizing the maximum relative load imposed on every edge. This problem, also known as Routing of Permanent Virtual Circuits, is NP-complete. The splittable version of this problem, which allows the traffic of each flow to be split over multiple routes, is known to be in $\mathcal{P}$.

The only difference between the N-hub problem and the unsplittable multicommodity flow problem is that in the former the set of possible routes for each source-destination pair is restricted while in the latter it is not. Hence, the unsplittable multicommodity flow problem can be viewed as a $|V|$-hub routing problem.

Corollary 2: The best competitive ratio that can be achieved by an online algorithm for N-hub has a lower bound of $\Omega(\log |V|)$.

Proof: In [23] this lower bound is proven for the unsplittable multicommodity flow problem. In this proof a specific network and a specific sequence of flows are considered. For this specific instance, the maximum load imposed on an edge by an offline algorithm is 1, whereas the maximum load imposed by an online algorithm is at least $\frac{\log |V|}{2}$. Since all the routes in the considered network have a length of at most three edges, each of them can be represented as a 1-hub route. Hence, this proof is also valid for the 1-hub problem, and for the general N-hub problem, as well. ■

However, if we consider a more practical variant of the online version, where termination of flows is permitted, i.e., the lifetime of each flow is finite, we can show that no routing algorithm can do better or worse than a competitive ratio of $\Theta(|E|)$.

Theorem 1: For the online version of N-hub, when flow termination is allowed, the competitive ratio that can be achieved by any algorithm is $\Theta(|E|)$.

Proof: Let us consider a directed graph that has a single source s, connected to a single target t via n directed edges, each with capacity C. We construct a sequence of n^2 flow requests, each with a traffic demand T. After all the n^2 flows are routed, the maximum load in the network is $m\frac{T}{C}$ where $m \geq n$. Let e be the edge with the maximum load. We now terminate all the flows that do not pass through e and some $(m-n)$ flows that do pass through e. The maximum load in the network now is $n\frac{T}{C}$. The optimal offline algorithm in this situation can maintain a maximum load of $\frac{T}{C}$ by routing each of the n remaining flows on a separate edge. Hence, the best competitive ratio a routing algorithm can achieve is at least $\Omega(|E|)$.

We now show that the worst competitive ratio any routing algorithm can achieve is $O(|E|)$. Consider a graph with $|E|$ edges, each with capacity C, and a sequence of n flow requests with traffic demand T. The maximum load that can be produced by an online algorithm in the worst case is $\frac{nT}{C}$. The maximum load that can be produced by the optimal offline algorithm is at least $\frac{nT}{|E|C}$. Hence, the worst competitive ratio that can be obtained is $O(|E|)$. Using Theorem 2, this result can be extended to networks with non-uniform edge capacities. ■

From Theorem 1 it follows that not much can be done if we want to guarantee some competitive ratio when flow termination is considered. However, from Corollary 2 it follows that when flow termination is not considered, it is possible to meet the challenge of designing an algorithm that has a competitive ratio of $O(\log |V|)$. In what follows we present some online algorithms for the problem.

These algorithms are similar in structure, as follows. Let f be a new flow to be routed. Let T_f be the bandwidth demand of f. Let L_e and U_e be the current load and capacity of link $e \in E$, respectively. From all feasible N-hub routes, the algorithm chooses the one that satisfies a given criterion as follows:

- Algorithm-1: minimize

$$\sum_{e \in P} a^{\frac{L_e + T_f/U_e}{\Lambda}} - a^{\frac{L_e}{\Lambda}},$$

where $a \in (1,2)$ and Λ is as explained below

- Algorithm-2: minimize

$$\text{MAX}_{e \in E} \begin{cases} L_e & e \notin P \\ L_e + \frac{T_f}{U_e} & e \in P \end{cases}$$

- Algorithm-3: minimize

$$\text{MAX}_{e \in P} L_e + \frac{T_f}{U_e}.$$

In all cases, P denotes a possible path for the considered flow.

These algorithms were presented in [23] (Algorithm-1) and in [24] (Algorithm-2 and Algorithm-3) for the unsplittable multicommodity flow problem, and are applied in this paper for the N-hub routing problem. We next show that the competitive ratios of the above algorithms for the N-hub problem are the same as for the unsplittable multicommodity flow problem, and that Algorithm-1 is the best online algorithm for the N-hub problem.

In Algorithm-1, Λ is an estimate for the value of the optimal solution. A simple doubling technique is used to estimate its value. The algorithm starts with some initial estimate. If, during execution, the maximum load exceeds Λ by $\log(|V|)$, the estimate is doubled and the algorithm is reinvoked. The algorithm assigns to each edge a weight that increases exponentially in the load that will be imposed if this edge is part of the route selected for the considered flow. The algorithm chooses from all possible routes for the considered flow the one with the minimum weight. A route's weight is the sum of the weights of all its edges. The intuition behind the exponential function weight is that as the load on an edge increases, the weight of the edge increases exponentially. Consequently, the algorithm prefers a long non-congested route over an exponentially shorter, but congested, route. The algorithm achieves a competitive ratio of $O(\log |V|)$ for the unsplittable multicommodity flow problem. To prove this, [23] uses the following auxiliary potential function:

$$\Phi(j) = \sum_{e \in E} a^{\frac{L_e(j)}{\Lambda}} (\gamma - L_e^*(j)/\Lambda), \tag{1}$$

where $L_e(j)$ and $L_e^*(j)$ are the load imposed on edge e by Algorithm-1 and by an optimal offline algorithm, respectively, after the first j flows are routed, and $a = 1 + 1/\gamma$. Function $\Phi(j)$ is non-increasing in j since the weight of the route chosen by the algorithm for every flow is not greater than the weight of the route chosen by an optimal offline algorithm. Since $\Phi(0) \leq \gamma|E|$ and $L_e^*(j)/\Lambda \leq 1$, $\gamma|E| \geq \sum_{e \in E} (\gamma - 1) a^{\frac{L_e(j)}{\Lambda}}$ holds, and the competitive ratio follows. For the N-hub problem, the weight of the route chosen by Algorithm-1 is still not greater than the weight of the route chosen by an optimal offline algorithm. This implies that the potential function in Eq. 1 is also non-increasing in j. Hence, the competitive ratio of $O(\log(|V|))$ holds for N-hub as well.

Algorithm-2 uses a simple greedy approach. It chooses a route such that the maximum load imposed on any edge is minimized after the flow is routed. When all edge capacities are equal, this algorithm has a competitive ratio of $O(\sqrt{D|E|})$, where D is the maximum ratio, over all flows, between the length of the longest and shortest routes that can be assigned to the flow. We now show that this competitive ratio is also valid for N-hub (when all the edges have equal capacities). In [24], where this competitive ratio is proven for the unsplittable multicommodity flow problem, the values of the loads are divided into levels. The load L_e on edge e is said to be in the i'th level if $i \cdot T_{max}/w \leq L_e \leq (i+1) \cdot T_{max}/w$, where T_{max} is the maximum bandwidth requirement and w is the capacity of the edges. The level of route P is the maximum level over all the edges in P. The crux of the proof is that when the maximum load in the network moves up to level i, then all the edges in the network, including the edges of the route chosen by the optimal offline algorithm, are at least in level $i-1$. Since this claim is also valid for N-hub, the competitive ratio is valid for N-hub as well. Theorem 2, which will be presented later in this section, shows how to adapt this competitive ratio to the general case where the edge capacities are not necessarily equal.

Algorithm-3 always chooses the route with the minimum load. The load of a route is defined as the maximum load over all the route's edges. The basic idea is to make the route selection criterion stricter than in Algorithm-2. To understand the difference between the two criteria, consider a network with two nodes connected by three edges with equal capacities. Suppose that the loads imposed on these edges by existing flows are 1, 4 and 6. Suppose also that the next flow to be routed has a bandwidth demand of 2. Algorithm-2 may route this flow either on the first edge or on the second edge, because in both cases the maximum load remains 6. In contrast, Algorithm-3 chooses the first edge because it is the least loaded. This implies that every route chosen by Algorithm-3 is also a valid choice for Algorithm-2, but not vice versa. In order to increase the attractiveness of Algorithm-2 over that of Algorithm-3, we have modified it in the following way. When Algorithm-2 finds several routes that do not increase the maximum load imposed on any edge, it does not choose one arbitrarily, as proposed in [24], but chooses the shortest one.

When Algorithm-3 is employed in networks with equal capacities, it has a competitive ratio of $O(d \log |V|)$, where d is the longest route that can be assigned to a flow. For reasons similar to those stated earlier for Algorithm-2, and others omitted here for lack of space, the same competitive ratio is guaranteed when Algorithm-3 is used for N-hub. Once again, we can use Theorem 2 to extend this competitive ratio to the case where edge capacities are not necessarily equal.

Theorem 2: Let $A(I)$ be an online algorithm for N-hub that achieves a competitive ratio of C in networks whose edges have the same capacity. Then, $A(I)$ achieves a competitive ratio of $\frac{u_{max}}{u_{min}} \cdot C$ in networks whose minimum edge capacity and maximum edge capacity are u_{min} and u_{max} respectively.

Proof: Let $G = (V, E)$ represent a network, and let $u : E \rightarrow \mathbf{R}^+$ be the edge capacity function. Let OPT be the value of an optimal offline solution. Let G' represent another network with the same structure but with a different

edge capacity function u', such that for every edge e $u'(e) = u(e)/\alpha$. Let OPT' be the value of an optimal offline solution for G'. We first prove that

$$OPT' = \alpha OPT. \quad (2)$$

Assume that $OPT'/\alpha < OPT$. Let S' be the solution corresponding to OPT'. Since the capacity of each edge in G is α times larger than the corresponding edge in G', applying the solution S' to the original graph G would yield a maximum load of OPT'/α. This maximum load is strictly lower than OPT, in contradiction to our assumption. A similar contradiction applies when $OPT'/\alpha > OPT$.

Let G_{min} be a graph similar to G whose edge capacities are equal to u_{min}. Let A_{min} and OPT_{min} be the values of the solutions found by the online algorithm and the optimal offline algorithm, respectively, for G_{min}. Since the edge capacities do not increase, $A \leq A_{min}$, where A is the value of a solution found by the online algorithm for G. Since the capacity of each edge in G_{min} is divided by a factor that is not greater than $\frac{u_{max}}{u_{min}}$, by Eq. 2 we get that $OPT_{min} \leq \frac{u_{max}}{u_{min}} OPT$. Since $A_{min} \leq C \cdot OPT_{min}$ holds, we conclude that $A \leq C \cdot \frac{u_{max}}{u_{min}} \cdot OPT$. ∎

We now discuss the time complexity of the three algorithms. Each algorithm has to review the entire set of N-hub routes before choosing one. There are $O(|V|^N)$ such routes. In a naive implementation, the algorithm metric is calculated independently for each route. Assuming that the maximum length of the shortest path between two nodes in the graph is D, the longest N-hub route is $D(N+1)$. Hence, each algorithm has to make $O(|V|^N D(N+1))$ metric calculations. When $N = 1$, and $D = O(|V|)$, the time complexity is $O(|V|^2)$. A faster approach for algorithms 1 and 3 is to precalculate the total metric for every possible shortest path between the graph nodes, using Dijkstra's algorithm, for example. In this case, the time complexity is $O(|V|^N(N+1) + |V|^3)$, which is smaller than the former time complexity for $|N| > 2$. The dominant operations in the metric calculation of Algorithm 1 are division and exponential computations for real values. In contrast, algorithms 2 and 3 require only division operations. Hence, Algorithm 1 has a higher time complexity and a longer expected running time.

V. Simulation Study

In this section we present simulation results for the routing algorithms discussed in the previous section. We generated router-level networks with random capacity edges, using Waxman's model [28] and the BRITE simulator [29]. We randomly chose a sequence of source-destination nodes. Each pair represents a flow to be routed in the network. The sequence of flows was generated using Zipf. A random network topology and a random sequence of flows form one instance of the N-hub routing problem. Using an event-driven simulator, we find for each instance the maximum load in the network under the following schemes:

1) The standard shortest-path routing scheme (SP) used today in IP. This is also known as minimum hop routing.
2) The hypothetical optimal routing (OPT) scheme. In this scheme we find a solution for the splittable multicommodity flow problem presented in Section IV. Recall that this version of the problem is in $\mathcal{P}$. An algorithm for OPT that is based on linear programming is presented in Appendix I. This scheme allows the traffic of a flow to be split over multiple routes. OPT's performance is a theoretical lower bound for N-hub, and therein lies its importance.
3) Algorithm-1, Algorithm-2 and Algorithm-3, as presented in Section IV.

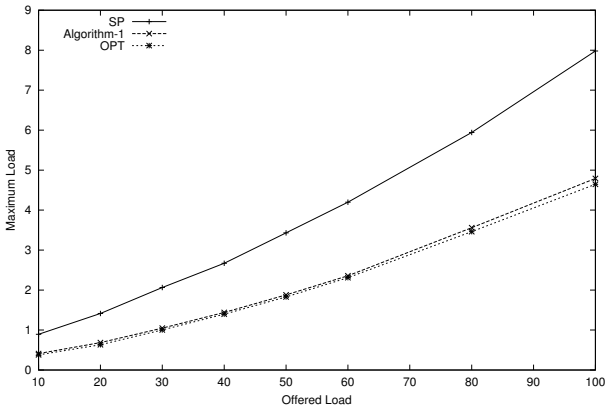
To solve the linear programs for OPT, we used the Lp_Solve software [30].

Throughout the simulation study, we assigned a random demand with a fixed average to each flow. Hence, there is a strong correlation between the number of flows the routing protocol has to handle and the load imposed on the network. We therefore use the number of flows as our offered load metric.

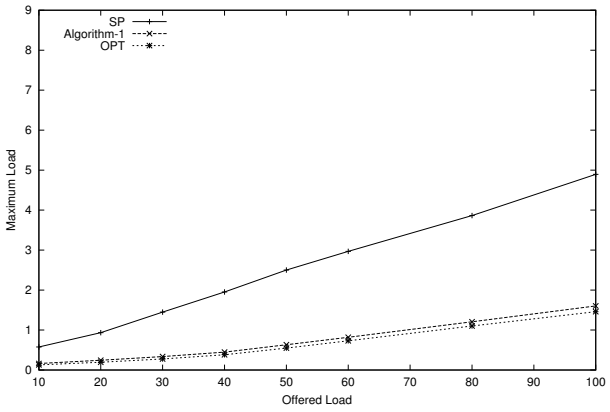
Figure 4 depicts simulation results of the routing schemes OPT, SP, and the first online algorithm (Algorithm-1) presented in Section IV. These simulations were carried out in a medium size backbone network (50 routers). Algorithm-1 is implemented with $N = 1$. The most important finding in these graphs, and probably in the research so far, is that *the performance of 1-hub is very close to that of OPT, and the improvement over SP is significant.* Algorithm-1 reduces the maximum load in the network by up to 73%. We also simulated Algorithm-2 and Algorithm-3 with $N = 1$. However, the performance of these algorithms is slightly lower than that of Algorithm-1. The inferior performances of Algorithm-2 and Algorithm-3 can be attributed to the fact that they do not take into account the length of the chosen routes. Longer routes impose, of course, greater load on the network.

We now compare the performance of the various algorithms in networks with different topologies. Fig. 4(a) shows simulation results for backbone networks with low link density ($|E|/|V| = 2$), whereas Fig. 4(b) shows the results for backbone networks with higher link density ($|E|/|V| = 5$). Note that as the link density increases, the number of routes between two nodes also increases. As expected, the maximum load produced by all the routing schemes decreases as the link density increases. However, while the maximum load produced by the shortest-path routing decreases on the average by only 25%, the maximum loads produced by the optimal routing scheme and by Algorithm-1 for 1-hub decrease by 65%. Since the shortest-path routing scheme uses only one path for a source-destination pair, the increase in the number of routes between two nodes is insignificant. In contrast, the optimal routing scheme and the 1-hub based routing algorithms can route different flows of a source-destination pair over different routes, in response to traffic conditions. Note that the ability to use various routes for a single source-destination pair is especially important for networks with hot-spots.

Figure 5 depicts simulation results for a small routing domain having $|V| = 10$ and $|E| = 20$. This routing domain has the same link density as the routing domain corresponding to Figure 4(a). This allows us to compare the performance of



(a) $|V| = 50,\ |E| = 100$



(b) $|V| = 50,\ |E| = 250$

Fig. 4. Performance of Algorithm-1 for 1-hub for different link densities

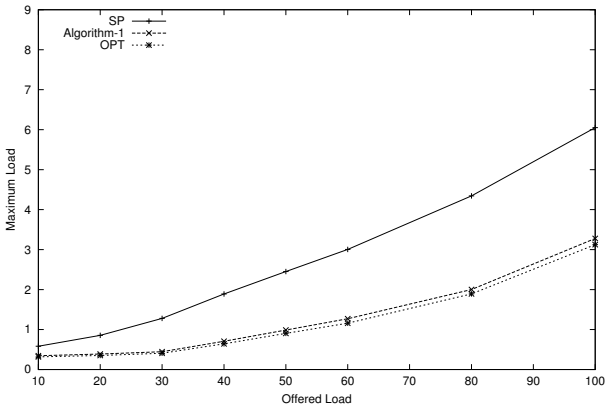


Fig. 5. Performance of the Algorithm-1 for 1-hub for small network ($|V| = 10,\ |E| = 20$)

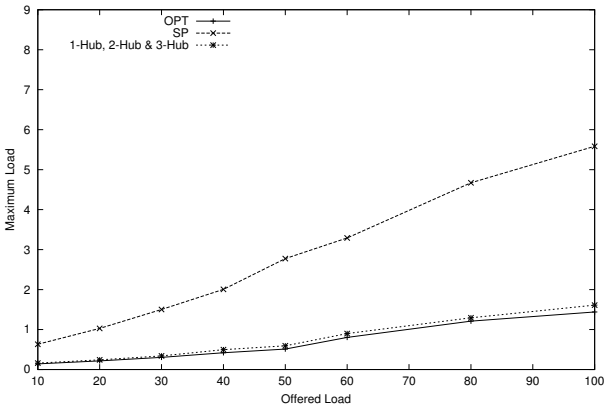
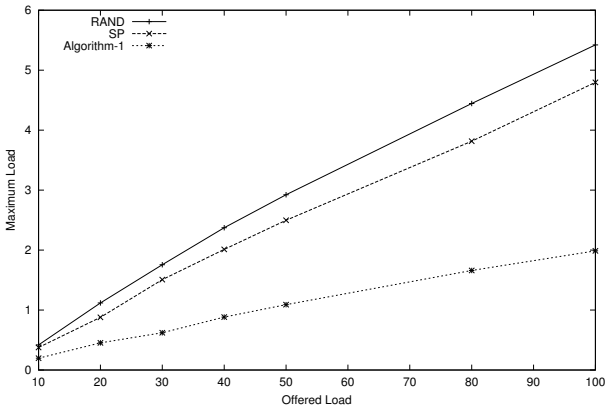


Fig. 6. Performance of Algorithm-1 for 1-hub, 2-hub and 3-hub ($|V| = 25$ and $|E| = 75$)

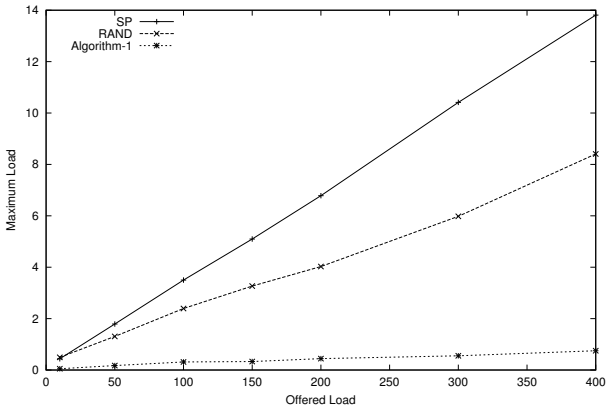
N-hub routing in routing domains with different numbers of routers. Note first that the increase in the number of routers has only a negligible effect on the difference in performance between the 1-hub and the optimal routing schemes. This is despite the fact that the number of unrestricted routes between two end nodes increases exponentially with the number of routers, whereas the number of 1-hub routes increases only linearly. One might expect the maximum loads produced by the various routing schemes to be higher in small routing domains than in larger ones, and the relative difference in the performances of the N-hub routing scheme and shortest-path routing to decrease. Interestingly, however, the maximum loads produced are actually smaller than in Figure 4(a) and the relative difference between 1-hub and shortest-path is similar to that of Figure 4(a). This is attributed to the fact that the average number of links a flow has to traverse decreases for smaller routing domains. Hence, each flow consumes fewer network resources, thereby reducing the maximum loads produced by the various routing schemes.

We now examine the performance of the N-hub routing scheme with different values for N, i.e., with different numbers of possible hubs. Figure 6 depicts simulation results of Algorithm-1 for the 1-hub, 2-hub and 3-hub schemes for a routing domain having $|V| = 50$ and $|E| = 250$. The most important finding is that the differences in performance for different values of N are negligible (less than 1%). We therefore use a single curve for $N = 1$, $N = 2$ and $N = 3$. This result is attributed to the flexibility of 1-hub routing. Adding flexibility by allowing more hubs to the routing process does not contribute to its effectiveness. However, for much larger routing domains, we expect a visible performance difference because the flexibility of 2-hub and 3-hub routing schemes increases polynomially (by powers of 2 and 3 respectively) with the number of routers, whereas the flexibility of a 1-hub scheme increases only linearly.

We wanted to investigate not only the specific algorithms proposed in the paper, but also the pure concept of N-hub routing. To this end, we tested the performance of 1-hub



(a) $|V| = 10$, $|E| = 30$



(b) $|V| = 200$, $|E| = 4000$

Fig. 7. Performance of a random algorithm for 1-hub for different network sizes

routing with a random algorithm (RAND). This algorithm does not take into account the aggregated load imposed on every network link or the load imposed by every connection. Rather, it selects a random hub for every new connection. We compared the performance of RAND with the performance of Shortest-Path (SP) and the performance of Algorithm-1 for different routing domain sizes. Figure 7(a) depicts simulation results for a small routing domain with 10 routers and 30 links. It is evident that RAND performs poorly in such domains, because the maximum load it imposes is even higher than the maximum load imposed by SP. This is attributed to the fact that like SP, RAND selects the routes without knowing the distribution of link loads. Since the routes selected by RAND are longer than those selected by SP, the bandwidth consumed by RAND is higher and the maximum load imposed in the routing domain increases. However, this is not the case for a large routing domain. Figure 7(b) depicts the performance of the various routing schemes for a domain with 200 nodes and 4000 links. The maximum load imposed by RAND is about 40% lower than the maximum load imposed by SP. This reduction is attributed to the fact that RAND is a symmetry-breaking procedure which better balances an offered load created with a Zipf distribution. This load reduction is not possible in a small routing domain in which the load is approximately uniform and there is no advantage in routing through distant hubs.

To validate our findings regarding the effectiveness of N-hub routing, we have also used for our simulations an actual ISP topology as mapped by the RocketFuel project [31]. The bandwidth for each link is determined according to [32]. Figure 8 depicts simulation results for the Exodus ISP from [31]. It is evident that the results are similar to those achieved using the Waxman model. Again, Alg-1 using 1-hub routing performs very close to the theoretical optimum and achieves 80% improvement over the results achieved by shortest-path routing. We found similar results when implementing other ISP topologies from [31]. The graphs are omitted due to lack of space.

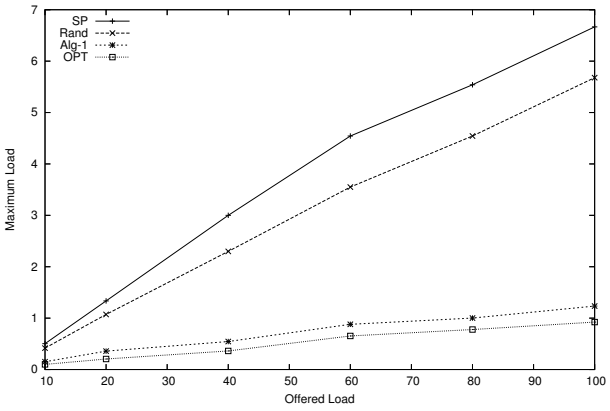


Fig. 8. Performance on an ISP (Exodus). $|V| = 80$, $|E| = 294$

We conclude this section by looking at the problem from a different angle. Figure 9 depicts the maximum number of flows the network can accommodate under each routing algorithm as a function of the maximum load that can be imposed on a single link. Instead of routing all the flows and finding the maximum load, we now determine the maximum number of flows that can be routed, subject to a maximum load constraint. A flow is rejected if routing it over the chosen route causes the maximum load in the network to exceed the maximum tolerated load. The simulation stops when the network is saturated. The network is assumed to be saturated when 100 consecutive flows are rejected. Fig. 9 depicts simulation results for networks with $|V| = 50$ and $|E| = 250$. We can see that the 1-hub version of Algorithm-1 achieves the best results: it can accommodate on the average 51% more flows than SP. Algorithm-3 achieves 48% improvement over SP, and Algorithm-2 achieves only 34% improvement. These results suggest that although the three routing algorithms produce similar maximum loads, as shown in the previous simulation, the difference in the quality of their routing is

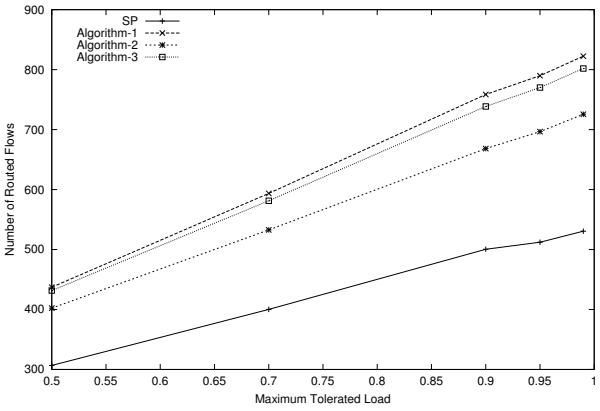


Fig. 9. No. of connections vs. max. tolerated load

distinct. The higher number of flows accepted by Algorithm-1 and Algorithm-3 indicates their ability to better balance the load in the network, thereby achieving a higher throughput.

VI. Conclusions

In this paper we studied the effectiveness of the N-hub Shortest-Path Routing concept in IP networks. We have demonstrated that this concept offers an excellent compromise between the simplicity of shortest-path routing and the efficiency of virtual circuit routing. We applied this concept to the problem of minimizing the maximum load in the network. We defined the corresponding optimization problem, and proved that it is NP-Complete even for $N = 1$. We also showed that it does not permit a PTAS and cannot be approximated within $2 - \epsilon$ for $\epsilon > 0$. However, we present in Appendix III a probabilistic asymptotic PTAS for the offline version of N-hub.

We have addressed the online version of N-hub, where the set of the input flows is not known in advance. We showed that the best competitive ratio an online N-hub algorithm may achieve is $\Omega(\log |V|)$. We then presented an online algorithm that achieves this lower bound, and two additional online algorithms that have less attractive competitive ratios, but are also less computationally intensive.

We then used simulations to study the practical effectiveness of N-hub routing in general, and of the specific algorithms presented in the paper. Our main findings are as follows:

- The performance of N-hub Shortest-Path Routing is very close to the performance of a hypothetical optimal algorithm that splits the traffic of the same flow among multiple routes.
- The N-hub Shortest-Path Routing scheme can produce much better quality routing than shortest-path routing, without the need to incorporate complicated logic into the routing process or even make the effort to learn the link load distribution throughout the routing domain.
- The effect of N on the performance of N-hub is very small. Hence, even the performance of 1-hub is very close to optimal.
- Although the competitive ratio of an online algorithm is $\Omega(\log |V|)$, all three online algorithms proposed in this paper perform very well in practice.

We therefore conclude that N-hub Shortest-Path Routing, and in particular the $N = 1$ version, should be considered as a powerful mechanism for future datagram routing in the Internet.

Appendix I
A linear program for the general routing problem

We describe a general routing problem, expressed in the form of a linear program, for the optimal routing scheme discussed in Section V. Let $G = (V, E)$ be a directed graph representing the network. Let $\mathcal{F}$ be a set of flows in the network. Let T_f denote the bandwidth demand of flow f, and let s^f and d^f denote the source and destination of flow f respectively. For every flow f and link e, let l_e^f represent the traffic load imposed on link e due to flow f.

The linear program is as follows:

$$\text{Minimize} \quad L$$

subject to the following constraints:

(a) $\sum_{e \in E_v^{in}} l_e^f - \sum_{e \in E_v^{out}} l_e^f = \begin{cases} T_f & \text{if } v = f^d \\ -T_f & \text{if } v = f^s \\ 0 & otherwise \end{cases}$

$\forall v \in V$ and $\forall f \in \mathcal{F}$

where E_v^{in} and E_v^{out} are the sets of incoming and outgoing links of vertex v respectively, and

(b) $\sum_{f \in \mathcal{F}} l_e^f \leq L \quad \forall e \in E.$

The first constraint ensures that the traffic flow is conserved in each vertex and it is routed from its source to its destination. The second constraint ensures that the load on each link does not exceed L.

Appendix II
An NP-Completeness Proof

The 1-hub routing problem is a special case of N-hub routing. In what follows we formulate the 1-hub problem with uniform capacities as a decision problem and prove that this problem is NP-complete. It is easy to see that if 1-hub with uniform capacities is NP-complete, then the more general N-hub problem with arbitrary capacities is NP-complete as well. An instance for the 1-hub problem is a directed graph $G = (V, E)$, a set $\mathcal{F} \subseteq V \times V$ of flows, a function T of bandwidth demand for each flow, and a positive real K. The question is whether there exists a hub $h^f \in V$ for each flow $f \in \mathcal{F}$ such that if the required traffic volume for f, namely $T(f)$, is routed over $s^f \rightarrow h^f \rightarrow d^f$, the total traffic routed through every link $e \in E$ does not exceed K.

Theorem 3: 1-hub is NP-complete.

Proof: It is easy to see that 1-hub $\in NP$. To prove that 1-hub is NP-complete we will show a reduction from SAT to 1-hub. Consider the following instance for SAT. Let $U = \{u_1, u_2, \ldots, u_N\}$ be a set of variables and $C = \{c_1, c_2, \ldots, c_L\}$ a set of clauses. A valid hub assignment for

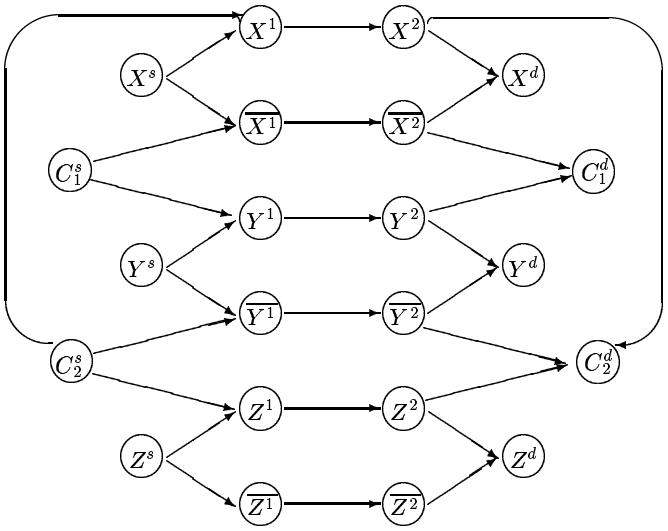


Fig. 10. An instance for the 1-hub problem constructed from the instance of SAT given in (3)

the 1-hub problem is an assignment that does not impose a traffic volume greater than K on any edge. We shall now transform the instance for SAT into an instance for the 1-hub problem such that a valid hub assignment for the 1-hub problem exists if and only if C is satisfiable.

For every variable $u_i \in U$ of SAT, $1 \leq i \leq N$, the following three sets are defined:

$$\begin{aligned} V_i^u = &\ \left\{u_i^1, u_i^2, \overline{u}_i^1, \overline{u}_i^2, u_i^s, u_i^d\right\} \\ E_i^u = &\ \left\{(u_i^s, u_i^1), (u_i^s, \overline{u}_i^1), (u_i^1, u_i^2), \right. \\ &\qquad \left. (\overline{u}_i^1, \overline{u}_i^2), (u_i^2, u_i^d), (\overline{u}_i^2, u_i^d)\right\} \\ \mathcal{F}_i^u = &\ \left\{(u_i^s, u_i^d)\right\}. \end{aligned}$$

For every clause $c_j \in C$ of SAT, $1 \leq j \leq L$, the following three sets are defined:

$$\begin{aligned} V_j^c &= \left\{c_j^s, c_j^d\right\} \\ E_j^c &= \left\{(c_j^s, u_i^1), (u_i^2, c_j^d) \mid u_i \in c_j\right\} \bigcup \\ &\qquad \left\{(c_j^s, \overline{u}_i^1), (\overline{u}_i^2, c_j^d) \mid \overline{u}_i \in c_j\right\} \\ \mathcal{F}_j^c &= \left\{(c_j^s, c_j^d)\right\}. \end{aligned}$$

An instance for the 1-hub problem is defined as follows:

$$\begin{aligned} V &= \left(\bigcup_{i=1}^N V_i^u\right) \bigcup \left(\bigcup_{j=1}^L V_j^c\right) \\ E &= \left(\bigcup_{i=1}^N E_i^u\right) \bigcup \left(\bigcup_{j=1}^L E_j^c\right) \\ \mathcal{F} &= \left(\bigcup_{i=1}^N \mathcal{F}_i^u\right) \bigcup \left(\bigcup_{j=1}^L \mathcal{F}_j^c\right) \\ T(f) &= \begin{cases} L & \text{if } f \in \mathcal{F}_i^u \ 1 \leq i \leq N \\ 1 & \text{if } f \in \mathcal{F}_j^c \ 1 \leq j \leq L \end{cases} \\ K &= L. \end{aligned}$$

As an example, Fig. 10 shows the graph of the corresponding 1-hub instance for the following SAT instance:

$$C = \{c_1, c_2\}, \tag{3}$$

$$\text{where } c_1 = \{\overline{x}, y\} \text{ and } c_2 = \{x, \overline{y}, z\}.$$

The flows of the 1-hub instance are: $\mathcal{F}_1^u = (x^s, x^d)$, $\mathcal{F}_2^u = (y^s, y^d)$, $\mathcal{F}_3^u = (z^s, z^d)$, $\mathcal{F}_1^c = (c_1^s, c_1^d)$, and $\mathcal{F}_2^c = (c_2^s, c_2^d)$. Their bandwidth demands are as follows: $T(\mathcal{F}_i^u) = 2$ for $i = 1, 2, 3$ and $T(\mathcal{F}_j^c) = 1$ for $j = 1, 2$.

It is easy to see that an instance for the 1-hub problem can be constructed in polynomial time.

We proceed by showing that a valid hub assignment for the 1-hub problem exists if and only if there exists a truth assignment that satisfies C in the corresponding SAT problem. Let us assume that C is satisfiable. Let $g : U \rightarrow \{\text{TRUE, FALSE}\}$ be a truth assignment for C. We now assign a hub for each $f \in \mathcal{F}$ as follows. For each $f \in \mathcal{F}_i^u$, where $f = (u_i^s, u_i^d)$ $1 \leq i \leq N$, we assign the vertex u_i^1 as a hub if $g(u_i) =$FALSE and the vertex $\overline{u}_i^1$ otherwise. For each $f \in \mathcal{F}_j^c$, where $f = (c_j^s, c_j^d)$ $1 \leq j \leq L$, we assign the vertex z_i^1 as a hub, where $z_i \in c_j$ and $g(z_i) =$TRUE. Note that such a literal must exist since g satisfies c_j. Clearly, the traffic volume between the pairs that correspond to the clauses is routed through the vertices corresponding to the literal whose value, as determined by g, is TRUE. Furthermore, the traffic volume of the pairs that correspond to the variables is routed through the vertices that correspond to the literals whose value is set to be FALSE. Hence, no edge in G has a load greater than L.

Let $h : \mathcal{F} \rightarrow V$ be a valid hub assignment in G. We now show how to construct from h an assignment g that satisfies C. From the way G is constructed, it follows that the traffic volume required for every flow $f \in \mathcal{F}_i^u$, $f = (u_i^s, u_i^d)$ $1 \leq i \leq N$ can be routed in two ways: either through u_i^1 or through $\overline{u}_i^1$. If this traffic is routed through u_i^1, we set $g(u_i) = FALSE$. If this traffic is routed through $\overline{u}_i^1$, we set $g(u_i) = TRUE$. Obviously, each variable in U is assigned one value. From the construction of G, it follows that the traffic volume required for a flow $f \in \mathcal{F}_j^c$, $f = (c_j^s, c_j^t)$ $1 \leq j \leq L$ must be routed through a vertex corresponding to a literal in c_j. From the construction of T, it follows that this literal is set by g to TRUE. Otherwise, h would not be a valid hub assignment. Hence, for every clause in C there is a literal whose value, as determined by g, is TRUE. Therefore, g satisfies C. ■

Corollary 3: N-hub is NP-complete. Moreover, a restricted version of N-hub referred to as *Integer N-hub*, where all the capacities are 1 and all the bandwidth demands are integers, is also NP-complete.

Proof: From the proof of Theorem 3 it follows that Integer 1-hub is NP-complete. Integer N-hub is NP-complete since it is a generalization of Integer 1-hub. Finally, N-hub is NP-complete since it is a generalization of Integer N-hub. ■

APPENDIX III
A PROBABILISTIC APPROXIMATION ALGORITHM FOR THE OFFLINE N-HUB PROBLEM

In section III we saw that N-hub is NP-complete and difficult to approximate. In this appendix we describe a probabilistic approximation algorithm for 1-hub. This algorithm is based upon an approximation technique presented in [33]. This algorithm is then extended for the more general N-hub problem. The algorithm is the basis of an Asymptotic PTAS for N-hub. We begin by describing the general technique we use in the algorithm and then present the algorithm itself.

A. The Approximation Technique

The idea is to formulate 1-hub as an integer program, and then to transform it into a rational linear program that can

be solved in polynomial time [34] by relaxing the integrality constraints of its variables. After the relaxed program is solved, the values of the relaxed variables are rounded either to 0 or to 1 in a randomized manner. Thus, with a certain probability, the value of the objective function, namely the maximum load in the network, is "close" to the optimum of the linear relaxation. It is therefore "close" to the optimum of the original integer programming problem. This concept was introduced in [33]. It is effective for problems whose objective function is an upper bound of sums of the problem's binary variables.

Let Π_I be an integer linear program and Π_R be its rational relaxation. Let the variables of the problem be $x_1, x_2, \ldots, x_n$. Note that in Π_I, $x_i \in \{0,1\}$ whereas in Π_R, $x_i \in [0,1]$. The basic algorithm, as presented in [33], consists of the following two phases:

1) Solve Π_R. Let the value assigned to every variable x_i be α_i, where $\alpha_i \in [0,1]$.
2) Set every variable x_i to 1 or 0 randomly, such that $Prob(x_i = 1) = \alpha_i$.

In some problems the constraints dictate that the variables should be partitioned into several sets, and the sum of the variables of each set must be 1. In these problems the variables in each set are still randomly rounded to 1, but in a mutually exclusive manner.

As mentioned above, this technique is suitable for problems whose objective function is an upper bound of the sums of its binary variables. Therefore, in order to approximate the objective function, an upper bound for these sums should be found. It was observed in [33] that the sum of the rounded variables is actually a sum of independent Bernoulli random variables, where each variable may be associated with a different probability. In order to find an upper bound for sums of this kind, [33] uses results from [35] and [36]. From these results the following is derived:

$$Prob(\Psi \geq m) < e^{-\frac{\beta^2 Np}{3}}$$

where Ψ is the sum of the independent Bernoulli variables, N is the number of the variables and $m = (1+\beta)Np$, where $0 < \beta \leq 1$ and $p = \frac{\sum_{k=1}^{N} p_k}{N}$ (p_k is the success probability for the kth Bernoulli variable).

This upper bound is applicable only for Bernoulli random variables and not for other random variables with a more general distribution. In the problem we consider the objective function is not necessarily an upper bound of sums of Bernoulli variables; it is actually an upper bound for sums of flow demands passing through the links. Hence, in the following we use a different probabilistic analysis.

B. The Approximation Algorithm

We now apply the approximation technique presented above to the 1-hub problem. We start by formulating 1-hub as an integer programming problem. For every flow f, and for every node i that can serve as a hub for the traffic of this flow, the following binary variable is defined:

x_{if} – A binary variable whose value is 1 if node i is assigned as a hub for the traffic of flow f and 0 otherwise.

Parameters:

- T_f – For every flow f, T_f indicates the traffic volume demanded by f.
- u_e – For every edge e, u_e indicates the capacity offered by e.
- z_{if}^e – For every flow $f = (s,d)$, node i and link e, let $z_{if}^e = 1$ if e is on the shortest path from s to i or from i to d and 0 otherwise.

The target function, Minimize L, is subject to the following constraints:

$$\begin{array}{ll} \text{(a)} & \forall f \quad \sum_i x_{if} = 1 \\ \text{(b)} & \forall e \quad \sum_{i,f} \frac{x_{if} \cdot z_{if}^e \cdot T_f}{u_e} \leq L \\ \text{(c)} & \forall i,\ f \quad x_{if} \in \{0,1\}. \end{array}$$

Constraint (a) ensures that exactly one node serves as a hub for f. Constraint (b) ensures that no edge will carry a relative traffic load greater than L. Constraint (c) ensures that the traffic of each flow is not split (i.e., it is routed on a single route).

The linear relaxation of the above program allows each variable x_{if} to be assigned any real value in $[0,1]$. This implies that we actually relax the requirement that for every flow there must be exactly one route that carries T_f (constraint (c)).

After obtaining an optimal solution for the relaxed linear program, we have for every flow $f = (s,d)$ a set of hubs Γ_f through which T_f is routed. Each hub $h \in \Gamma_f$ defines a route from s to d, that consists of the shortest paths from s to h and from h to d. Each such route carries a fraction of the traffic volume, T_f. Each hub $h \in \Gamma_f$ is associated with a weight equals to that fraction of T_f. The sum of the weights for every Γ_f is, of course, 1.

The next step is to convert the solution of the relaxed linear program into a solution of the original integer program by rounding the weight of one selected hub in every Γ_f to 1 and rounding the weights of the other hubs to 0. In other words, the entire traffic volume of f will be routed through the route defined by the selected hub from Γ_f. The hub is selected *randomly,* with a probability that is equal to its weight. Note that these random choices are made independently for each flow f. The following theorem shows that the presented approximation algorithm has an *absolute* performance factor of $O(\log(|E|))$. Namely, $|A(I) - OPT(I)| \leq O(\log(|E|))$.

Theorem 4: Let ϵ be a positive real such that $0 < \epsilon < 1$. Let L_{opt} be the optimum value of L obtained by the relaxed linear program. After a single hub is chosen for every f using the approximation algorithm, there is a probability greater than $1 - \epsilon$ that the load on each edge is upper bounded by:

$$L_{opt} + \left[\frac{n \ln(|E|/\epsilon)}{2}\right]^{1/2} \frac{T^{max}}{u^{min}},$$

where n is the number of flows. E is the set of links in the network, T^{max} is the maximum bandwidth demand of a flow, and u^{min} is the minimum capacity of an edge.

Proof: Consider an edge $e \in E$. Let L_e be the relative load imposed on e in the optimal solution as determined by the linear program. The load imposed on e by the approximation algorithm is a sum of n independent random variables, X_{ef} for $1 \leq f \leq n$. The value of X_{ef} indicates the contribution

of the traffic generated by f to the load imposed on e. Hence, the distribution of X_{ef} is as follows:

$$X_{ef} = \begin{cases} 0 & \text{with probability } 1 - p_{ef} \\ T_f/u_e & \text{with probability } p_{ef}, \end{cases}$$

where p_{ef} is the fraction of flow f routed over e according to the solution found by the linear program. Recall that the linear program is likely to split T_f between multiple routes. Some of these routes (or none of them) might include link e. Hence, p_{ef} is equal to the aggregated traffic of flow f carried by these routes. Namely, $p_{ef} = \sum_i x_{if} \cdot z^e_{if}$.

We know the following from [37]. Let $X_1, X_2, \ldots, X_n$ be independent random variables, where $0 \leq X_i \leq 1$. Let $\mu = E(S)$, where $S = X_1 + X_2 + \ldots + X_n$. Then, for every t, $0 < t < 1 - \mu/n$, the following holds:

$$Prob(S \geq nt + \mu) \leq e^{-2nt^2}.$$

Let S_e be a random variable such that $S_e = \sum_f X_{ef}$. Note that $E(S_e) = L_e$. In order to use the upper bound of [37], the random variables X_{ef} should take values in the range [0,1]. We therefore multiply X_{ef} by u^{min}/T^{max} for every f. Now, the load imposed by the algorithm on link e is a sum of random variables, $\hat{X}_{ef}$, of the following type:

$$\hat{X}_{ef} = \begin{cases} 0 & \text{with probability } 1 - p_{ef} \\ \frac{T_f}{u_e} \cdot \frac{u^{min}}{T^{max}} & \text{with probability } p_{ef}. \end{cases}$$

Let us denote this sum by $\hat{S}_e$. Note that $\hat{S}_e = S_e \frac{u^{min}}{T^{max}}$. Therefore, $E(\hat{S}_e) = L_e \frac{u^{min}}{T^{max}}$. This value does not exceed, of course, $L_{opt} \frac{u^{min}}{T^{max}}$.

Applying the upper bound of [37] mentioned above yields:

$$Prob\left(\hat{S}_e \geq nt + L_e \frac{u^{min}}{T^{max}}\right) \leq e^{-2nt^2}$$

for $0 < t < 1 - \frac{L_e u^{min}}{n T^{max}}$. Choosing

$$t = \left[\frac{\ln(|E|/\epsilon)}{2n}\right]^{1/2}$$

where ϵ is a positive real smaller than 1, yields

$$Prob\left(\hat{S}_e \geq nt + L_e \frac{u^{min}}{T^{max}}\right) \leq \frac{\epsilon}{|E|}. \tag{4}$$

Let $\hat{S} = \text{MAX}_{e \in E}\{\hat{S}_e\}$. Hence, $\hat{S}$ is the normalized maximal load imposed on any link according to the solution obtained by the approximation algorithm. Note that $L_{opt} \geq L_e$. From Eq. (4) we get:

$$Prob\left(\hat{S} < nt + L_{opt} \frac{u^{min}}{T^{max}}\right) > 1 - \epsilon. \tag{5}$$

We now return to the original problem with the original bandwidth demands. Let $S = \text{MAX}_{e \in E}\{S_e\}$. Hence, S is the non-normalized maximal load imposed on any link according to the solution obtained by the approximation algorithm. From (5) we get:

$$\begin{aligned} &Prob\left(S < nt \tfrac{T^{max}}{u^{min}} + L_{opt}\right) = \\ &\quad Prob\left(\hat{S} < nt + L_{opt} \tfrac{u^{min}}{T^{max}}\right) > 1 - \epsilon, \end{aligned} \tag{6}$$

which concludes the proof. ■

The presented approximation algorithm and Theorem 4 are also applicable to the more general N-hub with the obvious modifications. We consider an ordered N-tuple of nodes as a supernode. Instead of assigning to every flow a single node as its hub, we assign a supernode. There are $|V|^N$ supernodes in $G = (V, E)$. The formulation of N-hub as an integer programming problem is the same as 1-hub, except that x_{if} equals 1 if supernode i is assigned as a "hub" to flow f and that z^e_{if} equals to 1 if e is on the route defined by the end nodes of f and supernode i. The rest of the analysis is similar to the 1-hub analysis.

In some cases it would be desirable to guarantee with a high probability that the solution of the approximation algorithm will not exceed the optimal solution by a certain factor. Let us consider the case where this factor is 2. From Eq. 6 it follows that in order to ensure that the solution will not exceed the optimal solution by this factor, $nt \frac{T^{max}}{u^{min}} \leq L_{opt}$ must hold. This yields the following constraint on L_{opt}:

$$\left[\frac{n \ln(|E|/\epsilon)}{2}\right]^{1/2} \frac{T^{max}}{u^{min}} \leq L_{opt}.$$

Note that this does not impose a rigid upper bound but rather a probabilistic one. Furthermore, it should be noted that the approximation ratio of the above algorithm will be as small as we want it to be, provided that we increase the maximum load in the network. It represents, therefore, an asymptotic PTAS [26].

REFERENCES

[1] Z. Wang and J. Crowcroft, "Shortest path first with emergency exits," in *Proceedings of the ACM SIGCOMM*, September 1990, pp. 166–176.

[2] J. Chen, P. Druschel, and D. Subramanian, "An efficient multipath forwarding method," in *Proceedings of the IEEE INFOCOM*, San Francisco, March 1998, pp. 1418–1425.

[3] J. Garcia-Luna-Aceves, S. Vutukury, and W. Zaumen, "A practical approach to minimizing delays in Internet routing protocols," in *Proceedings of the IEEE ICC*, Vancouver, June 1999.

[4] S. Vutukury and J. Garcia-Luna-Aceves, "A simple approximation to minimum delay routing," in *Proceedings of the ACM SIGCOMM*, 1999.

[5] E. Rosen, "Multiprotocol label switching architecture (MPLS)," RFC 3031, January 2001.

[6] D. K. et. al., "Traffic engineering (TE) extensions to OSPF version 2," IETF RFC 3630, September 2003.

[7] S. B. et al., "An architecture for differentiated services," IETF RFC 2475, December 1998.

[8] J. Postel, "Internet protocol," IETF RFC 791, September 1981.

[9] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *Computer Communications Review*, vol. 9, no. 2, pp. 32–48, April 1989.

[10] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proceedings of the IEEE INFOCOM*, Tel Aviv, March 2000, pp. 1371–1380.

[11] S. Deering and R. Hinden, "Internet Protocol, version 6 (IPv6) specification," IETF RFC 2460, December 1998.

[12] C. Perkins, "IP encapsulation within IP," IETF RFC 2003, October 1996.

[13] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. 18th ACM Symposium on Operating Systems Principles*, October 2001.

[14] P. Francis and R. Gondivan, "Flexible routing and addressing for a next generation IP," in *Proceedings of the ACM SIGCOMM*, September 1994, pp. 116–125.

[15] I. Castineyra, N. Chiappa, and M. Steenstrup, "The Nimrod routing architecture," RFC 1992, August 1996.

[16] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient and robust routing of highly variable traffic," in *Third Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[17] R. Zhang-Shen and N. McKeown, "Designing a predictable internet backbone network," in *Third Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.
[18] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Preconfiguring IP-over-optical networks to handle router failures and unpredictable traffic," in *Proceedings of the IEEE INFOCOM*, March 2006.
[19] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Maximum throughput routing of traffic in the hose model," in *Proceedings of the IEEE INFOCOM*, March 2006.
[20] Y. Z. L. Qiu, Y. R. Yang and S. Shenker, "On selfish routing in Internet-like environments," in *Proceedings of the ACM SIGCOMM*, 2003.
[21] S. G. Kollopoulus and C. Stein, "Improved approximation algorithms for the unsplittable flow problems," in *Proceedings of FOCS*, 1997, pp. 426–435.
[22] N. G. Y. Dinitz and M. Goemans, "On the single-source unsplittable flow problem," *Combinatorica*, vol. 19, pp. 17–41, 1999.
[23] J. A. et al, "Online load balancing with applications to machine scheduling and virtual circuit routing," *Journal of the ACM*, vol. 44, no. 3, pp. 486–504, 1997.
[24] J. T. Havill and W. Mao, "Greedy online algorithms for routing permanent virtual circuits," *Networks*, vol. 34, pp. 136–153, 1999.
[25] W. Mao and R. Simha, "Routing and scheduling file transfers in packet-switched networks," *Journal of Computing and Information*, vol. 1, pp. 559–574, 1994.
[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Co., 1979.
[27] R. M. Karp, "Online algorithms versus offline algorithms: How much is it worth to know the future?" International Computer Science Institute, Technical Report TR-92-044, 1992.
[28] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *IEEE INFOCOM*, vol. 2. San Francisco, CA: IEEE, March 1996, pp. 594–602.
[29] I. M. A. Medina, A. Lakhina and J. Byers, "BRITE: An approach to universal topology generation," in *Proceedings of MASCOTS*, 2001.
[30] M. Berkelaar, "Lp_solve software," ftp.es.tue.nl/pub/lp_solve.
[31] R. M. N. Spring and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proceedings of the ACM SIGCOMM*, August 2002.
[32] D. W. R. Mahajan, N. Spring and T. Anderson, "Inferring link weights using end-to-end measurements," in *In ACM SIGCOMM Internet Measurement Workshop (IMW)*, November 2002.
[33] P. Raghavan and C. D. Thompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, pp. 365–374, 1987.
[34] L. Khachiyan, "A polynomial time algorithm for linear programming," *Docl. Akad. Nauk SSSR*, vol. 244, pp. 1093–1096, 1979.
[35] W. Hoeffding, "On the distribution of the number of successes of independent trials," *Annals of Math. Stat.*, vol. 27, pp. 713–721, 1956.
[36] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations," *Annals of Math. Stat.*, vol. 23, pp. 493–509, 1952.
[37] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *American Statistics Association Journal*, vol. 58, pp. 13–30, 1963.

PLACE PHOTO HERE

Reuven Cohen (M'93, SM'99) received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science from the Technion - Israel Institute of Technology, completing his Ph.D. studies in 1991. From 1991 to 1993, he was with the IBM T.J. Watson Research Center, working on protocols for high speed networks. Since 1993, he has been a professor in the Department of Computer Science at the Technion. He has also been a consultant for numerous companies, mainly in the context of protocols and architectures for broadband access networks. Dr. Cohen has served as an editor of the IEEE/ACM Transactions on Networking, and the ACM/Kluwer Journal on Wireless Networks (WINET). Dr. Cohen is a senior member of the IEEE and heads the Israeli chapter of the IEEE Communications Society.

PLACE PHOTO HERE

Gabi Nakibly (S'04) received the B. Sc. in Information Systems engineering (summa cum laude) and M. Sc. in Computer Science from the Technion - Israel Institute of Technology, Haifa, Israel, in 1999 and 2004, respectively. Since 2005, he has been a Ph.D. student in Computer Science Department in the Technion, working on QoS routing and traffic engineering.