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 traverse 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. This concept has been suggested in the past but this paper is the first to offer an in-depth investigation of it. We apply this concept to the routing problem of minimizing the maximum load in the network. We show that the resulting routing problem is a difficult (NP-Complete) problem and that it is also hard to approximate. However, we propose efficient algorithms for solving this problem both in the online and the offline contexts. Our results show that N-hub Shortest-Path Routing can increase the network utilization significantly even for $N = 1$. Hence, this routing paradigm should be considered as a powerful mechanism for the future datagram routing in the Internet.

I. INTRODUCTION

Routing in the Internet is based on the hop-by-hop shortest-path paradigm. The source of a packet specifies the address of the destination, and each router along the route forwards the packet to a neighbor located “closest” to the destination. Since usually the routing is 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 lay 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 therefore minimizes the bandwidth consumed by every packet, it does not guarantee full utilization of the network resources under high traffic loads. When the network is heavily loaded, some of the routers introduce an excessive delay while others are under-utilized. In some cases this non-optimized usage of the network resources may introduce not only excessive delays but also high packet loss rate.

A lot of study 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 a better delay-throughput behavior than the 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, it is desirable that such a scheme will be able to seamlessly inter-operate with network routers that continue to employ the shortest-path routing paradigm.

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 make routing decisions based on the accumulated information regarding 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. They find several loop-free paths in advance and distribute the load among them. However, due to their complexity, increased processing burden, and considerable deviation from the conventional shortest-path routing paradigm, none of these schemes have been adopted for the Internet. A major drawback of many proposed routing schemes is the necessity for those schemes to be deployed over the lion share of the routing domain in order to be effective.

This paper investigates the benefit of 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 discussed in Section II. N-hub routing allows the ingress router of a routing domain to determine one or more intermediate nodes (“hubs”) through which 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 a single hub – node $G$: packets are routed over 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 link $B-C$ or the link $C-D$ are heavily loaded whereas the links $B-G$, $G-H$ and $H-D$ are under-utilized. Finally, path-3 uses 2 hubs – $F$ and $B$: packets are routed 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 gets better control over the routing process, while the network routers continue to employ the static shortest-path paradigm for building their routing tables. Although this concept is not employed today in the Internet, we think that it is a powerful tool that should be investigated in the context of traffic engineering and QoS. To the best of our knowledge, this paper is the first to propose thorough theoretical and practical investigation of the potential of the concept of N-hub Shortest-Path Routing. In this paper we focus on applying this routing paradigm on the problem of minimizing the maximum load in the network.

The contribution of this paper is two-fold. On the theoretical side, we show that “N-hub Shortest-Path Routing” is an NP-Complete problem, and that it is also hard to approximate. Moreover, we show that in the online context, where the sequence of flows that have to be routed is not known in advance, the best competitive ratio that can be achieved for this problem is $O(\log |V|)$. On the practical side, we present efficient algorithms for the online context, and analyze their worst-case performance. Using simulations we show that the performance of these algorithms is very close to the optimal performance one may achieve when having full control over the routing of each flow, as in 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 in order to implement N-hub routing. In Section III we define the N-hub routing problem and show that this problem is NP-complete for every $N \geq 1$. In Section IV we propose 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

We are not aware of any work that addresses the computational complexity and the potential effectiveness of the concept of N-hub Shortest-Path routing, which is the core of this paper. However, several routing schemes that leverage, in one way or another, the concept of setting intermediate routers along a packet’s route were proposed in the past. In [5], this concept is identified as a flexible addressing and forwarding scheme. Ref. [6] proposes a new scalable routing architecture that takes advantage of this concept by allowing a packet to include a sequence of ‘locators’ through which it can traverse. Ref. [7] also proposes a routing architecture, referred to as “Internet Indirection Infrastructure” (I3), that is capable of routing a packet through several nodes in the network. Ref. [8] discusses an implementation of an overlay network that employs this concept in order to set routes that circumvents hot spots and link failures in the network, thereby increasing the reliability of the network.

The concept of N-hub Shortest-Path routing can be realized using several mechanisms. A straightforward way is to take advantage of the IPv4 Loose Source-Routing option [9]. However, this option, much like any other IPv4 option, is rarely used. The main reason for this is the heavy processing burden imposed on the general purpose CPU of the router when the IPv4 header contains any optional fields. Moreover, there are some notable security issues related to this option [10]. Ref. [11] noted that only 8% of the routers in the Internet are source-routing capable.

As opposed to IPv4, IPv6 [12] has a more “built-in” support for the 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 through which the packet should traverse before reaching the destination. Since all the headers of an IPv6 packet have a similar structure, the processing of a packet can always be performed by the dedicated hardware with no performance penalty.

Another way to implement N-hub routing in IPV4 is to use IP-in-IP encapsulation [13]. 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.

N-hub routing can also be implemented through the use of an overlay network [8]. An overlay network employs a higher level routing protocol implemented at the application layer. Such a routing protocol may use N-hub routing in order to achieve better control over the routes traversed by the packets of each flow.

Another powerful way to implement the N-hub routing paradigm is by employing MPLS [14]. This can be done by setting an MPLS tunnel along an explicit route. The explicit route can be implemented either by network layer encapsulation or by an LSP which is constructed through an appropriate LDP such as CR-LDP [15]. In the latter case, some or all of the hops included in the explicit route label request message are marked with the L bit set, indicating the ‘loose’ nature of the hops.

As already said, the main benefit from the ability to determine one or more intermediate nodes (hubs) for a route between a source-destination pair is achieving a better control

![Fig. 1. An example for a N-hub routing.](image-url)
over the load distribution in the network, without deviating much 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. Yet, the network has the capability to route a packet over less congested areas. Moreover, it can be employed in an effective way even if a small fraction of the network routers support it. This is because traffic can be diverted to less congested area without the support of the core routers.

It is well known that there is a trade-off between the simplicity of the traditional datagram (shortest-path) routing and the efficiency of virtual-circuit (strict) routing. However, both schemes can be viewed as private cases of N-hub routing: with \( N = 0 \) for shortest-path routing and \( N = \mid V \mid \) for virtual circuit routing. Hence, N-hub routing, where \( 0 \leq N \leq \mid V \mid \), 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 little deviation from the traditional shortest-path routing.

As mentioned, the ingress router of the routing domain, also known as the “edge router”, should be responsible for determining the intermediate router(s) through which the packets of each flow will traverse. To this end, the router may use information it acquires regarding the load distribution in the network via a link state routing protocol such as OSPF [16]. This is in accordance with the concept of Differentiated Service [17] (DiffServ) that lays most of the QoS-related overhead on the edge routers.

### 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. More specifically, our 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, of minimizing the maximum load imposed on a single link, was addressed in the past mainly in the context of the “Multicommodity Flow” problem [18], [19] and the Virtual Circuit Routing problem [20]–[22]. This objective does not always guarantee perfect load balancing and minimum average delay. However, it was shown in the past to yield good performance since the delay on a link grows exponentially with the load. Moreover, this objective is easier to analyze from a theoretical point of view.

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. Another possible objective is minimizing the variance of the loads on the network links. However, such an objective does not take into account the actual load on the links and therefore may 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 to the N-hub routing problem. Let \( G = (V, E) \) be a directed graph. Each edge, \( e \in E \), has a capacity \( u(e) \), where \( u : E \rightarrow \mathbb{R}^+ \). 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 \( T : \mathcal{F} \rightarrow \mathbb{R}^+ \). 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 \( h_f^1, h_f^2, \ldots, h_f^N \), where \( h_f^i \in V \), such that the packets of \( f \) are routed over \( s_f \rightarrow h_f^1 \rightarrow h_f^2 \rightarrow \ldots \rightarrow h_f^N \rightarrow d_f \), 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 \( \sum_{f \in \mathcal{F}} \frac{P_f T(f)}{u(e)} \), where \( P_f \) is the path chosen to route flow \( f \).

#### B. 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 weather 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 1:** 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 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:

\[
\begin{align*}
V_i^u &= \{ u_i^1, u_i^2, \pi_i^1, \pi_i^2, u_i^s, u_i^d \}, \\
E_i^u &= \{ (u_i^s, u_i^1), (u_i^1, \pi_i^1), (u_i^1, u_i^2), (\pi_i^1, \pi_i^2), (u_i^2, u_i^d) \}, \\
F_i^u &= \{ (u_i^s, u_i^d) \}
\end{align*}
\]

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

\[
\begin{align*}
V_j^c &= \{ c_j^1, c_j^2 \}, \\
E_j^c &= \{ (c_j^i, u_i^j), (u_i^j, c_j^i) \mid u_i \in c_j \} \cup \{ (c_j^i, \pi_j^i), (\pi_j^i, c_j^i) \mid \pi_j \in c_j \}, \\
F_j^c &= \{ (c_j^1, c_j^2) \}
\end{align*}
\]

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

\[
\begin{align*}
V &= \left( \bigcup_{i=1}^N V_i^u \right) \cup \left( \bigcup_{j=1}^L V_j^c \right), \\
E &= \left( \bigcup_{i=1}^N E_i^u \right) \cup \left( \bigcup_{j=1}^L E_j^c \right), \\
F &= \left( \bigcup_{i=1}^N F_i^u \right) \cup \left( \bigcup_{j=1}^L F_j^c \right), \\
T(f) &= \begin{cases} 
L & \text{if } f \in F_i^u, \ 1 \leq i \leq N, \\
1 & \text{if } f \in F_j^c, \ 1 \leq j \leq L
\end{cases}, \\
K &= L
\end{align*}
\]

As an example, Fig. 3 shows the graph of the corresponding 1-hub instance for the following SAT instance:

\[
C = \{ c_1, c_2 \}
\]  \hspace{1cm} (1)

where \( c_1 = \{ x, y \} \) and \( c_2 = \{ x, \bar{y}, z \} \).

The flows of the 1-hub instance are:

\[
\begin{align*}
F_1^u &= (x^s, x^d), \quad F_2^u = (y^s, y^d), \quad F_3^u = (z^s, z^d), \quad F_1^c = (c_1^1, c_1^2) \quad \text{and} \quad F_2^c = (c_2^1, c_2^2).
\end{align*}
\]

Their bandwidth demands are as follows: \( T(F_i^u) = 2 \) for \( i = 1, 2, 3 \) and \( T(F_j^c) = 1 \) for \( j = 1, 2 \).

It is easy to see that an instance for the 1-hub problem can be constructed in a 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 \( g : U \rightarrow \{ \text{TRUE}, \text{FALSE} \} \) be a truth assignment for \( C \). We now assign a hub for each \( f \in F \) as follows. For each \( f \in F_i^u \), where \( f = (u_i^s, u_i^d) \) \( 1 \leq i \leq N \), we assign the vertex \( u_i^1 \) as a hub if \( g(u_i) = \text{FALSE} \) and the vertex \( \pi_i^1 \) otherwise. For each \( f \in F_j^c \), where \( f = (c_j^1, c_j^2) \) \( 1 \leq j \leq L \), we assign the vertex \( z_i^1 \) as a hub, where \( z_i \in c_j \) and \( g(z_i) = \text{TRUE} \). Note that such a literal must exist since \( g \) satisfies \( c_j \). It can be easily seen that 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 \( f \in F_i^u \), \( f = (u_i^s, u_i^d) \) \( 1 \leq i \leq N \), can be routed in two ways: either through \( u_i^1 \) or through \( \pi_i^1 \). If this traffic is routed through \( u_i^1 \), we set \( g(u_i) = \text{FALSE} \). If this traffic is routed through \( \pi_i^1 \), we will set \( g(u_i) = \text{TRUE} \).
Obviously, each variable in $U$ is assigned one value. From the construction of $G$ follows that the traffic volume required for a flow $f \in F^c_j$, $f = (e^o_j, e^j)$, $1 \leq j \leq L$, must be routed through a vertex corresponding to a literal in $c_j$. From the construction of $T$ 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 $C$.

**Theorem 2:** 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 1 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.

**C. On the Approximation Hardness of N-hub**

It can be easily shown that there is no FPTAS [23] for N-hub unless $P = NP$. To this end, consider a restricted version of N-hub where all edge capacities are 1 and all bandwidth demands are also 1. This restricted version is still NP-complete, since its corresponding decision problem is the Edge-Disjoint Paths problem, which is known to be NP-Complete. The existence of an FPTAS implies the existence of a pseudo-polynomial algorithm. However, since Edge-Disjoint Paths is not a number problem, the existence of a pseudo-polynomial algorithm for this problem implies the existence of a polynomial-time algorithm, contradicting the fact that this problem is NP-Complete (assuming that $P \neq NP$). However, in what follows we show a stronger inapproximability result.

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

The following theorem is due to [23].

**Theorem 3:** Let $\Pi$ be an integer minimization problem. Suppose that the decision problem $\Pi_K$ is NP-hard for some constant $K$. Then, unless $\mathcal{P} = \mathcal{NP}$, there is no PTAS for $\Pi$ and there is no polynomial algorithm with an approximation ratio that is strictly less than $1 + 1/K$.

**Theorem 4:** Unless $\mathcal{P} = \mathcal{NP}$, N-hub does not admit a PTAS and cannot be approximated within $2 - \epsilon$ for $\epsilon > 0$.

**Proof:** 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 Theorem 2 the problem of deciding whether the optimum value of Integer N-hub is equal to 1 is also NP-hard. Applying Theorem 3 with $K = 1$ yields that Integer N-hub, and hence N-hub, does not admit a PTAS and cannot be approximated within $2 - \epsilon$ for any $\epsilon > 0$.

Appendix II 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 concentrate on the non-preemptive version, where once a flow is routed it cannot be rerouted. Furthermore, it is assumed that the flows continue for a very long time, hence we do not handle termination of flows. We present here three online approximation algorithms, that were originally developed for the “Unsplittable Multicommodity Flow” problem [23], and are modified to be suitable for the N-hub problem. We also present the competitive ratio for each of these algorithms. 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 [24] for more 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 $u : E \rightarrow \mathbb{R}^+$. Let $F \subseteq V \times V$ be an ordered set of flows between pairs of source and destination nodes. Each flow $f \in F$ has a traffic requirement $T(f)$, where $T : F \rightarrow \mathbb{R}^+$. Route every flow $f \in F$, 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, that allows to split the traffic of each flow 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 restricted. Hence, the Unsplittable Multicommodity Flow problem can be viewed as a $[\vert V \vert]$-hub routing problem.

**Theorem 5:** The best competitive ratio an online algorithm for N-hub may achieve has a lower bound of $\Omega(\log \vert V \vert)$.

**Proof:** In [20] 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 $\frac{\log \vert V \vert}{2}$. 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, we can show that no routing algorithm can do better or worse than a competitive ratio of $\Theta(\vert E \vert)$.

**Theorem 6:** For the online version of N-hub, when flow termination is allowed the competitive ratio any algorithm may achieve is $\Theta(\vert E \vert)$.

**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 $m_T^2$, where
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 $nT_e^2$. The optimal offline algorithm in this situation can maintain a maximum load of $\frac{T}{2}$ 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 a 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 an online algorithm may produce in the worst case is at most $\frac{nT}{C}$. The maximum load the optimal offline algorithm may produce is at least $\frac{nT}{2C}$. Hence, the worst competitive ratio that can be obtained is $O(|E|)$.

From Theorem 6 follows that not much can be done if we want to guarantee some competitive ratio when flow termination is considered. On the other hand, from Theorem 5 follows that when flow termination is not considered designing an algorithm that has a competitive ratio of $O(\log |V|)$ is a challenge. In what follows we present some online algorithms for the problem.

The algorithms we present have a similar 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
  \[ \sum_{e \in P} a \frac{T_f U_e}{\Lambda} - a \frac{L_e}{\Lambda}, \]
  where $a \in (1, 2)$ and $\Lambda$ is explained below.
- **Algorithm-2**: minimize
  \[ \text{MAX}_{e \in E} \left\{ \frac{L_e}{U_e} : e \notin P \right\} \]
  \[ \text{MAX}_{e \in P} \frac{T_f}{U_e}. \]
- **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 [20] (Algorithm-1) and in [21] (Algorithm-2 and Algorithm-3) for the Unsplittable Multicommodity Flow problem, and are adapted in this paper for the N-hub routing problem.

In Algorithm-1, $\Lambda$ is an estimate for the value of the optimal solution. A simple doubling technique is used in order to estimate its value. The algorithm starts with some initial estimate. If during the execution of the algorithm the maximum load exceeds $\Lambda$ by $\log(|V|)$, the approximation ratio of the algorithm, the estimate is doubled and the algorithm is re-invoked. The algorithm assigns to each edge a weight that increases exponentially in the load that would occur 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 weight of a route 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 by an exponential amount. 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, [20] uses the following auxiliary potential function:

\[ \Phi(j) = \sum_{e \in E} a \frac{e^{\gamma - j}}{\Lambda}, \]

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 $L_e^*(j)/\Lambda \leq 1$, $\gamma |E| \geq \sum_{e \in E}(\gamma - 1) a \frac{e^{\gamma j}}{\Lambda}$ 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. 2 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 route and the length of the shortest route that can be assigned to the flow. We now show that this competitive ratio is also valid for N-hub (when all the edges has equal capacities). In [21], 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 main point in 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 7 presented later shows how to adopt 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 since it is the least loaded one. 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 versus 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 an arbitrary one, as proposed in [21], but 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 similar considerations to those stated earlier for Algorithm-2, and other considerations which are not mentioned here due to lack of space, the same competitive ratio is guaranteed when the Algorithm-3 is used for N-hub. Once again, using Theorem 7 we can extend this competitive ratio to the case where edge capacities are not necessarily equal.

**Theorem 7:** 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 $\frac{\max \cdot C}{\min \cdot C}$ 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 $u : E \rightarrow \mathbb{R}^{+}$ be the edge capacity function. Let $OPT$ be the value of an optimal offline solution. Let $G'$ be another network with the same structure but with a different edge capacity function $u'$, such that for every edge $e \in E$, $u'(e) = u(e)/\alpha$. Let $OPT'$ be the value of an optimal offline solution for $G'$. We first prove that

$$OPT' = \alpha OPT$$

Assume that $OPT'/\alpha < OPT$. Let $S'$ be the solution corresponding to $OPT'$. Since the capacity of each edge in $G$ is $\alpha$ times larger than the corresponding edge in $G'$, applying the solution $S'$ to the original graph $G$ would yield a maximum load of $OPT'/\alpha$. 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. 3 we get that $OPT_{min} \leq \frac{u_{max}}{u_{min}}OPT$. Since $A_{min} \leq C \cdot OPT_{min}$ holds, we conclude that $A \leq C \cdot \frac{u_{max}}{u_{min}} \cdot OPT$.

**V. SIMULATION STUDY**

In this section we present simulations results for the routing algorithms discussed in the previous section. We have generated router-level networks with random capacity edges based on Waxman’s model [25] using the BRITE simulator [26]. We randomly chose a sequence of source-destination nodes. Each such a pair represents a flow to be routed in the network. The sequence of flows were generated using Zipf. A random network topology and a random sequence of flows form together one instance for the N-hub routing problem. Using an event-driven simulator we find for each instance the maximum load in the network using the following schemes:

1) The standard “Shortest-Path Routing” (SP) scheme used today in IP i.e. 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 $P$. An algorithm for OPT based on linear programming is presented in Appendix I. This scheme allows the traffic of a flow to be split over multiple routes. The importance of OPT is that the performance it achieves can be considered as a theoretical lower bound for N-hub.
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 [27].

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.

In Fig. 4 we show the performance of the first online algorithm (Algorithm-1) presented in Section IV. The figure depicts the performance of these three algorithms, as well as the performance of OPT and SP, 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 whole paper, is that the performance of 1-hub is very close to the performance of OPT, and that the improvement over SP is significant. Algorithm-1 reduces the maximum load in the network relative to SP by up to 73%. We have also simulated Algorithm-2 and Algorithm-3 with $N = 1$. However, the performance of these algorithms is slightly lower than the results of Algorithm-1. The difference was too small to be shown in the graph.

In order to 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 number of links increases. However, while the maximum load produced by the shortest-path routing decreases on the average only by 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 does not have a significant effect. 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 depending on the traffic conditions in the network. Note that the ability to use various routes for a single source-destination pair is especially important for networks with hot-spots.

Fig. 5 shows simulation results for a small network ($|V| = 10, |E| = 20$). The average number of routes between two nodes in networks of this size is of course smaller than in larger networks. One may expect that the maximum loads will be higher than in large networks, and that the relative difference between the performance of the N-hub routing scheme and the shortest-path routing will be reduced. However, it is interesting to note that the loads produced are actually smaller than in Fig. 4(a) and the relative difference between 1-hub and shortest-path is similar to that of Fig. 4(a). This is attributed to the fact that the average number of links a flow has to traverse increases as the network becomes larger. Hence, each flow consumes more network resources and the link load increases.

In Fig. 6 we look at the problem from a different angle. This figure depicts the maximum number of flows the network can accommodate under each of the routing algorithms 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 by routing it over the chosen route the maximum load in the network exceeds the maximum tolerated load. The simulation stops when the network is saturated. The network is assumed to be saturated when the probability of an arbitrary flow to be accepted is smaller than 0.01. Fig. 6 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 results for the previous type of simulations, the quality of their routing are still distinct. The higher number of flows accepted by Algorithm-1 and Algorithm-3 indicate their ability to better balance the load in the network, thereby achieving a higher throughput.

We conclude this section with simulation results of Algorithm-1 for 1-hub, 2-hub and 3-hub. The results are shown in Fig. 7 for the case where $|V| = 50$ and $|E| = 250$. The most important thing to notice is that the differences we found in the performance vs. $N$ are negligible. We therefore use a single curve for $N = 1$, $N = 2$ and $N = 3$.

**VI. CONCLUSIONS**

In this paper we studied the effectiveness of the N-hub Shortest-Path Routing concept in IP networks. We believe
we have shown that this concept offers an excellent compromise between the simplicity of shortest-path routing and the efficiency of virtual circuit (strict) 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 admit a PTAS and cannot be approximated within \(2 - \epsilon\) for \(\epsilon > 0\). However, we presented in an appendix 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 ratio, but are also less computational intensive.

We then used simulations in order 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 an hypothetical optimal algorithm that may split the traffic of the same flow among multiple routes.
- The effect of \(N\) on the performance of N-hub is very small. Hence, even the performance of 1-hub is very close to the optimal performance.
- Although the competitiveness ratio of an online algorithm is \(\Omega(\log |V|)\), all the 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 version with \(N = 1\), should be considered as a powerful mechanism for the future datagram routing in the Internet.

**Appendix I**

A linear program of 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 \(F\) 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 \(l^f_e\) represent the traffic load imposed on link \(e\) due to flow \(f\).

The linear program is as follows:

\[
\begin{align*}
\text{Minimize} & \quad L \\
\text{subject to} & \quad \sum_{e \in E^\text{in}_v} l^f_e - \sum_{e \in E^\text{out}_v} l^f_e = \begin{cases} 
T_f & \text{if } v = f^d \\
-T_f & \text{if } v = f^s \\
0 & \text{otherwise}
\end{cases} \\
& \quad \forall v \in V \text{ and } \forall f \in F \\
& \quad \sum_{e \in F} l^f_e \leq L \quad \forall e \in E.
\end{align*}
\]

where \(E^\text{in}_v\) and \(E^\text{out}_v\) are the sets of incoming and outgoing links of vertex \(v\) respectively, and

\(\mathbf{a})\sum_{e \in E^\text{in}_v} l^f_e - \sum_{e \in E^\text{out}_v} l^f_e = \begin{cases} 
T_f & \text{if } v = f^d \\
-T_f & \text{if } v = f^s \\
0 & \text{otherwise}
\end{cases} \\
& \quad \forall v \in V \text{ and } \forall f \in F \\
& \quad \sum_{e \in F} l^f_e \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**

A probabilistic approximation algorithm for the offline N-hub

In section III we saw that N-hub is NP-complete and it is also 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 [28]. This algorithm is then extended for the more general N-hub problem. The algorithm is a basis of an Asymptotic PTAS for N-hub. Before we begin we present the following result from [29] which will be of use latter on:
Theorem 8: 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 $t$, $0 < t < 1 - 1/\mu/n$, the following holds:

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

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:

$x_{i,f}$ – 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 available capacity offered by $e$.
- $z^*_{i,f}$ – For every flow $f = (s,d)$, node $i$ and link $e$, let $z^*_{i,f} = 1$ if $e$ is on the shortest-path from $s$ to $i$ or from $i$ to $d$.

The target function is:

$$\text{Minimize } L$$

subject to the following constraints:

(a) $\forall f \sum_i x_{i,f} = 1$
(b) $\forall e \sum_i x_{i,f} z_{i,f} T_f u_e \leq L$
(c) $\forall i, f \sum_i x_{i,f} \in \{0, 1\}$

The linear relaxation of the above program allows each variable $x_{i,f}$ 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 $\Gamma_f$ 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 $T_f$. Each hub $h \in \Gamma_f$ is associated with a weight that equals to that fraction of $T_f$. The sum of the weights for every $\Gamma_f$ 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 $\Gamma_f$ to 1 and rounding the weights of the other hubs to 0. In other words, the entire traffic of $f$ will be routed through the route defined by the selected hub from $\Gamma_f$. The selection of the hub is made randomly, with a probability that is equal to its weight. Note that this 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 9: Let $\epsilon$ 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, with a probability greater than $1 - \epsilon$ 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: (sketch) 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 on edge $e$ after the randomized rounding procedure - which we denote as $S_e$ - is a sum of the traffic loads of the flows that pass through it. This is actually a sum of random variables. We normalize these random variables by multiplying them with $\frac{u_{min}}{T_{max}}$. We denote the sum of the normalized random variables by $\hat{S}_e$. Note that $E(S_e) = L_e \frac{u_{min}}{T_{max}}$.

Applying Theorem 8 yields:

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

For $0 < t < 1 - \frac{L_e u_{min}}{nT_{max}}$. Choosing

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

where $\epsilon$ is a positive real smaller than 1, yields

$$\text{Prob} \left( \hat{S}_e \geq nt + L_e \frac{u_{min}}{T_{max}} \right) \leq \frac{\epsilon}{|E|}$$

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:

$$\text{Prob} \left( \hat{S} < nt + L_{opt} \frac{u_{min}}{T_{max}} \right) > 1 - \epsilon$$

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:

$$\text{Prob} \left( S < nt + L_{opt} \frac{u_{min}}{T_{max}} \right) = \text{Prob} \left( \hat{S} < nt + L_{opt} \frac{u_{min}}{T_{max}} \right) > 1 - \epsilon,$$

which concludes the proof.

The presented approximation algorithm and Theorem 9 are also applicable to the more general N-hub with the obvious modifications.

References


