A Precomputation Scheme for Minimum Interference Routing: the Least-Critical-Path-First Algorithm

Gábor Rétvári, József J. Bíró, Tibor Cinkler, Tamás Henk
High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
H-1117, Magyar Tudósok körútja 2., Budapest, Hungary
E-mail: {retvari, biro, cinkler, henk}@tmit.bme.hu

***Abstract*— This paper focuses on the selection of bandwidth-guaranteed channels for communication sessions that require it. The basic idea comes from Minimum Interference Routing: select a feasible path that puts the least possible restriction on the available transmission capacity of other communicating parties. This is achieved by circumventing some critical bottleneck links. The main contribution of the paper is a novel characterization of link criticality, the criticality threshold, which can be readily precomputed for routing dozens of subsequent calls. Based on this finding we define a generic precomputation framework for minimum interference routing, the Least-Critical-Path-First routing algorithm. We show by means of extensive simulations that efficient route precomputation is possible even in the case, when accurate resource availability information is not immediately available.**

***Index Terms*— Graph Theory, QoS routing, traffic engineering, route precomputation, network flows**

I. Introduction

The question of deliberate, effective and adaptive selection of dynamic bandwidth-guaranteed paths has gained substantial research interest recently. Obviously, any service that requires the transmission of loss and/or delay sensitive data over a packet switched public network infrastructure obligates the assurance of transmission quality guarantees on some prioritized traffic. This paper deals with the problem of finding a data path in a network for a traffic instance, which is both *feasible* and *efficient*. A path is feasible if it provides enough dedicated resources to satisfy pre-declared bandwidth demands. We say that a path is efficient, if it manifests some efficiency criteria of the network operator. For example, an efficient path would be such that the maximum link load is minimized. We assume that route requests arrive one by one and are fully characterized by their respective bandwidth demands. However, we do not presume any knowledge on the volume of present and future traffic demands.

Nowadays, the omnipresent Shortest-Path-First (SPF) routing algorithm has gained almost exclusive use in data networks. SPF manifests a powerful efficiency criterion, namely, the selected paths use as few resources as possible. Though, path feasibility is not guaranteed. Therefore, the ubiquitous use of SPF in today's Internet is deemed to be a major origin of its unreliability. Several proposals have come up to eliminate the shortcomings of SPF, e.g., widest-shortest-path routing (WSP) [1] and shortest-widest-path routing (SWP) [2]. Yet the most promising new technology seems to be Minimum Interference Routing [3]. The basic idea is to select bandwidth-guaranteed paths as to ensure that the chosen path blocks future requests to the least possible extent. Since the full-fledged minimum-interference routing problem is known to be NP-hard, a polynomial time approximate algorithm, the Minimum Interference Routing Algorithm (MIRA) was proposed.

The rationale behind MIRA lies in the incorporation of explicit knowledge on the traffic source-destination pairs (called sessions throughout this paper) to minimize the interference between them. Interference emerges when the traffic of some session is routed to a bottleneck link of some other session(s). MIRA searches these *critical links* and does its best to avoid using them in the course of path selection. Several recent research results have pointed out the potential of MIRA to improve routing performance while efficiently preserving resources for future requests at the same time. Various derivatives (e.g., [4], [5]), a treatment of both bandwidth and delay guarantees ([6]) and schemes for admission control ([7], [8]) have been defined lately.

MIRA was originally designed with the assumption that the routing algorithm possesses accurate and consistent information on the topology of the network and the availability of unreserved bandwidth at the interfaces. However, in an operational network environment the frequency of link state updates is basically a trade-off between the consistency of the network state information and the bandwidth consumption and processing overhead of routing protocol messages [9], [10], [11]. Since link criticality in fact reflects the network state that was valid the last time when a link state update occurred, hence the need to regenerate it for each and every connection setup request is invalidated. In [3] the authors study MIRA performance in the case, when link criticality is precomputed once, and used multiple times to route several subsequent requests. The resultant *criticality precomputation* scheme is not only more realistic, but also has the potential to decrease the prohibitive online computational complexity,

which is usually pointed out as a weakness of MIRA. In this paper, we shall show that without some clever modifications, the performance of MIRA falls well under that of the WSP algorithm when used with criticality precomputation.

The main result of this paper is the definition of a sophisticated measure of link criticality, the so called criticality threshold. Since the criticality threshold supplies a thorough characterization of link criticality, we can precompute it for routing dozens of subsequent calls. The calculation of this measure turns out to be computationally expensive, therefore, we show powerful optimization techniques to reduce the involved efforts to a tolerable level. Based on these findings we propose a new routing algorithm, the Least-Critical-Path-First (LCPF) algorithm, and by means of extensive simulations we show that LCPF performs route precomputation much more efficiently than either MIRA or WSP. We believe that our work is not only useful as the first really viable solution for precomputation in minimum interference routing but it also provides interesting further insights into this intriguing new routing technology.

The rest of this paper is structured as follows. Section II describes MIRA in more detail. In Section III we show, how to obtain a good quantity on link criticality and in Section IV we define the LCPF algorithm that makes heavy use of this quantity. In Section V, the practical usefulness of LCPF is evaluated and finally, Section VI concludes our work.

II. The MIRA Algorithm

Experience suggests that in order to achieve efficient routing it is not enough to simply pick a path from the set of feasible paths. One has to find a sufficient policy on how to deliberately select a feasible path that manifests some network-global traffic engineering goal and define a good algorithm that implements the policy. The main idea of MIRA is the recognition that if a network link acts as a bottleneck for a communication session, then admitting traffic of another session to that critical link will cause interference. The more traffic flows through critical links, the higher the interference. This leads to inefficient routing in the long term. Thus, it is a plausible network-wide traffic engineering goal to minimize the interference along the selected paths.

In order to better capture the notion of interference one has to invoke the elaborated toolset of network flow theory [12], [13]. Let $G(V, E, R)$ be a digraph. Let V be the set of nodes ($|V| = n$), E the set of edges ($|E| = m$) and R the set of edge capacities. Source-destination pairs (s, d), all of which are members of a known set P, are assumed to be known in advance. Such (s, d) pairs are called *sessions* for short. Let $p = |P|$. Let f be a *maximum flow* in G for some session (s, d). Then, $F_G(s, d)$ denotes the value of the maximum flow $|f|$, $f(u, v)$ denotes the flow traversing a particular edge $(u, v) \in E$, G_f denotes the flow residual graph induced by f and C_{sd} is a set of edges, which belong to one or more *minimum cuts*.

Minimum cuts have special role in MIRA. Consider some session (s, d) and the task is to select a path for some other session in a way as to minimize the interference along the selected path. If, after the path selection, the transmission capacity offered by the network for (s, d), that is, the maxflow of (s, d) remained intact, then the selected path does not interfere with (s, d). If, on the other hand, the maxflow $F_G(s, d)$ is decreased, then interference shows up. This is caused by routing traffic onto some critical bottleneck link, which has the property that decreasing the capacity of the link decreases the maximum flow of (s, d). As shall be pointed out later, this property just equals to the property that the link is included in the minimum cut set of (s, d). Henceforward, we shall say that an (u, v) edge is *critical* for some session (s, d), if the edge is included in the minimum cut set for the session, i.e., $(u, v) \in C_{sd}$. Any edge in C_{sd} is subject to interference. Therefore, for every link MIRA assigns an additive link weight, which is proportional to the number of sessions that the link is critical for, and computes the shortest weighted path over this weight set.

Hence, the path selection for a traffic instance in session $(a, b) \in P$ of demand D units involves the following basic steps in MIRA:

1) *Critical link identification:* for each $(s, d) \in P \setminus (a, b)$ compute the critical link set C_{sd}.
2) *Cost assignment:* map link criticality to additive link weights. The weight of link (i, j) is defined as:
$$w(i, j) = \sum_{(s,d):(i,j)\in C_{sd}} \alpha_{sd} \ , \qquad (1)$$
where α_{sd} represents the relative importance of session (s, d). In order to force path feasibility, any link (i, j) of inappropriate capacity ($R(i, j) < D$) is filtered out.
3) *Path selection:* find the shortest weighted path over the cost set defined by w.

In the vast majority of practical cases the computational complexity of MIRA is dominated by the $p - 1$ maxflow computations needed for critical link identification. Therefore, it is plausible to take this CPU intensive calculation in the background and only perform it when up-to-date network state information becomes available. Then, one decision on link criticality impacts routing of several subsequent requests. So, it would be of extreme usefulness to somehow detect whether there is a chance that a link turns to be critical in the near future. The notion of criticality threshold developed in the foregoing section serves just this purpose.

III. The Criticality Threshold

As the first step of route selection, MIRA solves the following decision question: given an edge and a session, is it the case that the edge is critical for the session, i.e., is it included in some minimum cuts for the session? Throughout this paper, we use the following form of the link criticality conditions:

Proposition 1: For an edge (i, j) and a session (s, d), (i, j) is critical for (s, d) if, for some maxflow f of (s, d):

i) $f(i, j) > 0$ and
ii) $F_{G_f}(i, j) = 0$

In this context, a link is critical for a session, if for some maxflow it carries nonzero flow and the endpoints of the link happen to reside in different cuts of the flow residual graph.

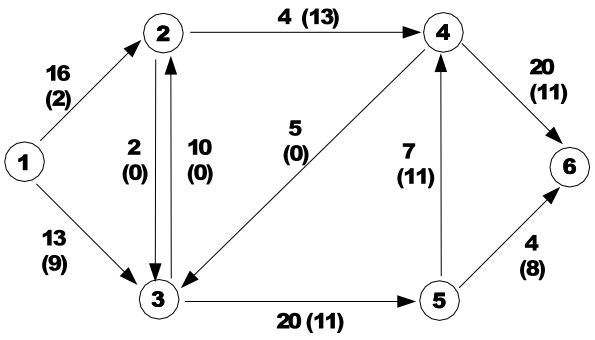

Fig. 1. Illustration of the criticality threshold

That is, there is no path from i to j in G_f. This also implies that $(i,j) \in C_{sd}$. The main contribution of this paper is the observation that there exists a well-defined threshold on the capacity of any link, the so called *criticality threshold*, such that if the capacity of the link falls beyond this threshold then the link turns to critical. This idea is captured in the following definition:

Definition 1: For a session (s,d) and an edge (i,j), the criticality threshold K is defined as:

i) $0 < R(i,j) \leq K \Rightarrow (i,j) \in C_{sd}$ *(criticality)*
ii) $R(i,j) > K \Rightarrow (i,j) \notin C_{sd}$ *(non-criticality)*

The significance of the criticality threshold is twofold. First, given a bandwidth demand of size Δ, the criticality threshold tells whether admitting the demand to any link (i,j) would render that link critical. This is simply done by checking $R(i,j) - \Delta \leq K$ (c.f. [3], Δ-criticality). Furthermore, the criticality threshold supplies an appealing way to assess the criticality (or non-criticality) of any link. For example, a good measure would be $K/R(i,j)$ or $K - R(i,j)$.

An illustration of the concept of the criticality threshold is provided in Fig. 1. One can easily check that if the link capacities are set as shown in the figure, then the maxflow for session (1,6) is 15 units and the minimum cut set, that is, the critical link set of MIRA, is constituted by links (2,4), (5,4) and (5,6). Now, consider for instance link (1,3). This link is not critical, however, lessening the capacity from 13 units to 9 units renders it critical. This is because, together with links (2,3) and (2,4), the link constitutes a minimum cut (again, of value 15). Further lessening the capacity of link (1,3) from 9 units to, say, 7 units would decrease the maxflow to 13 units. This phenomenon is captured in the criticality threshold, shown in parentheses in the figure. To put it in another way, one can safely route 4 units of traffic onto link (1,3) without risking that the maxflow of session (1,6) decreases. Additionally, both of links (2,4) and (5,6) are critical in our example, however, link (2,4) would still remain critical if its capacity was increased by as much as 9 units as opposed to link (5,6), for which this increase is only 4 units. This conveys the impression that link (2,4) is "more critical" in some sense than link (5,6).

The most plausible approach to find the criticality threshold is to observe the value of the maximum flow while decreasing the capacity of some network link from infinity to zero. First, set the capacity of the link to infinity. For a graph $G(V,E,R)$, the resultant graph $G^U(V,E,R^U) : R^U(i,j) = \infty$ is the so called *unconstrained graph* w.r.t. link (i,j). A maxflow in G^U is called unconstrained maxflow. On the contrary, in the *constrained graph* w.r.t. link (i,j) we set $R^C(i,j) = 0$. A maxflow in G^C is called constrained maxflow. One can characterize the constrained, the original and the unconstrained maxflows as $F_{G^C} \leq F_G \leq F_{G^U}$.

Consider the real-valued function, which describes the maxflow $F_G(s,d)$ as the function of the capacity $R(i,j)$ of some link (i,j). An immediate observation is that this function is non-negative and piece-wise linear on $R \in [0,\infty)$ and the partial derivative can be written as:

$$\frac{\partial F_G(s,d)}{\partial R(i,j)} = \begin{cases} 1 & \text{if } R(i,j) \leq K \\ 0 & \text{if } R(i,j) > K \end{cases} \tag{2}$$

The reasoning comes from Proposition 1: if (i,j) is critical, then for any (s,d) maxflow f, (i,j) is filled to capacity by a non-zero flow. Thus, decreasing $R(i,j)$ creates excess flow on (i,j) and this excess flow can not be routed around (i,j) since $F_{G_f}(i,j) = 0$. Therefore, the maxflow also decreases by the same amount. In contrary, if $R(i,j) > K$, that is, (i,j) is not critical, then either there is no flow on (i,j) (in which case its capacity can be reduced without affecting the maxflow) or there is non-zero available capacity from i to j onto which excess flow can be placed (or both). This yields that the derivative is zero in this case[1]. In some respect, (2) can be thought of as a simple mathematical reformulation of the definition of link criticality, that is, a link is critical for a session if decreasing the capacity of the link causes the reduction of the maxflow of the session.

Since the criticality threshold may be larger than the actual capacity of the link, it is more useful to examine maxflows in the unconstrained graph rather than the original graph. We define the *feasible maxflow set* w.r.t. link (i,j) and session (s,d) as the set of all unconstrained maxflows:

$$\mathcal{F}_{ij}^{sd} = \{f : f \text{ is a } (s,d) \text{ maxflow in } G^U\} \ . \tag{3}$$

Any flow f in the feasible maxflow set generates a candidate for the criticality threshold. What we need to do is to create a new graph G for each $f \in \mathcal{F}_{ij}^{sd}$ and set $R(i,j) = f(i,j)$. Then, obviously, the flow on (i,j) is strictly positive and the residual capacity is zero. If, furthermore, all augmenting paths from i to j are saturated in the flow residual graph, then all conditions in Proposition 1 are satisfied and (i,j) is critical in G. Since, as easily checked, (i,j) is not critical for any $R(i,j) > f(i,j)$ in this case, the current flow f generates the criticality threshold, that is, $K = f(i,j)$. In other words, criticality is achieved when no flow can be relocated from (i,j) to other flow paths, which would decrease the flow on the link. This implies that the flow generating criticality sends a *minimum* flow onto (i,j) in some sense. This idea is captured in the following definition.

[1]We note that the derivative is not defined at $R(i,j) = K$. Thus, (2) in fact gives the *lower derivative*. Nevertheless, for the sake of notational convenience, we shall use (2) in the generic sense in the foregoing developments.

Definition 2 (Committed flow): The committed flow for a session (s,d) and a link (i,j) is defined as:

$$\psi_{ij}^{sd} = \min\{f(i,j) : f \in \mathcal{F}_{ij}^{sd}\} \ . \tag{4}$$

The notion of committed flow insists that there are various ways to accommodate the maxflow in a graph, however, there is a certain amount of flow *committed* to a particular link in all cases[2]. Following, we show that the above intuitive reasoning is correct and the choice $K = \psi_{ij}^{sd}$ indeed yields the criticality threshold:

Theorem 1: For an edge (i,j) and a session (s,d) the criticality threshold is identical to the committed flow:

$$0 < R(i,j) \leq \psi_{ij}^{sd} \Rightarrow (i,j) \in C_{sd} \tag{5}$$

$$R(i,j) > \psi_{ij}^{sd} \Rightarrow (i,j) \notin C_{sd} \tag{6}$$

Proof: First we prove that in case of $R(i,j) = \psi_{ij}^{sd} > 0$, all conditions in Proposition 1 hold, therefore (i,j) is critical in this case. Let f be the unconstrained maxflow, which generated ψ_{ij}^{sd}, that is, $f \in \mathcal{F}_{ij}^{sd}$, such that $f(i,j) = \psi_{ij}^{sd}$. Clearly, one can always find such f. Furthermore, create a modified graph G out of G^U by decreasing the capacity of (i,j) from infinity to ψ_{ij}^{sd}, that is, create $G(V,E,R) : R(i,j) = \psi_{ij}^{sd}$. Then, f is feasible in G (i.e., it does not violate the link capacities and satisfies the flow conservation constraints), because it is feasible in G^U and, although filling up (i,j) to capacity, it is feasible on (i,j) as well. Therefore, f is also a maxflow for G. Now, consider the conditions in Proposition 1:

i) $f(i,j) > 0$ holds, since, by our assumption on f, $f(i,j) = \psi_{ij}^{sd} > 0$.

ii) We need to see that (a) $R_f(i,j) = 0$ and (b) there are no $i \to j$ feasible paths in the flow residual graph. (a) holds, because $R_f(i,j) = R(i,j) - \psi_{ij}^{sd} = 0$. (b) is also true, because if an $i \to j$ alternative path happens to exist in G_f, then some flow can be shifted from (i,j) to the alternative path, which yields a maxflow f'. But, as easily checked, f' is also an unconstrained maxflow. Accordingly, $f'(i,j) < f(i,j) = \psi_{ij}^{sd}$ would contradict our assumption that f is minimal.

It is also true that reducing the capacity of a critical link both reduces the maxflow and leaves it critical (c.f. (2)). This proves (5). On the other hand, for any $G(V,E,R) : R(i,j) > \psi_{ij}^{sd}$, (i,j) is not critical. This is because the above f remains to be a maxflow in G and f leaves non-zero residual capacity on (i,j), since $R(i,j) - \psi_{ij}^{sd} > 0$. Thus, (i,j) is not critical, which completes the proof of (6). ∎

A naive way to find the criticality threshold would be to search through all feasible unconstrained maxflows and find the one that commits the minimum flow to the selected link. Though, this method would not be too effective. Another approach would be to use linear programming *Parametric Analysis* [13], however, it would mandate the use of a full-fledged linear program solver. The following fundamental flow theory result gives an easy way to compute the criticality threshold:

[2]If there exists a direct (s,d) link in G, that is, $(s,d) \in E$, then the committed flow is infinite. This does not cause theoretical difficulties, however, special care must be taken when implementing the resultant algorithms.

Theorem 2: For a graph G, an edge (i,j) and a session (s,d), the committed flow for (i,j) is the difference of the unconstrained and the constrained maxflow:

$$\psi_{ij}^{sd} = F_{G^U}(s,d) - F_{G^C}(s,d) \ . \tag{7}$$

Proof: Consider a maxflow f in the constrained graph G^C. Put (i,j) back to G^C and increase its capacity in infinitesimally small steps. Then, the maxflow in the resultant graph G can be written as:

$$F_G(s,d) = F_{G^C}(s,d) + \int_0^{R(i,j)} \frac{\partial F_G(s,d)}{\partial R} \mathrm{d}R \ . \tag{8}$$

Applying (2) and using Theorem 1, (8) can be given for any $R(i,j) \geq 0$. Since $R(i,j) = \infty$ yields the unconstrained maxflow we have that $F_{G^U}(s,d) = F_{G^C}(s,d) + \psi_{ij}^{sd}$, which proves the theorem. ∎

Given a link and a session, one has to perform two maxflow computations to obtain the committed flow: one, when the capacity of the link is set to infinity and another one when its capacity is zero. The difference gives the threshold. The worst case complexity of this method is $O(n^2\sqrt{m})$ using the Goldberg-Tarjan highest-label preflow-push algorithm [12]. In the rest of this section we show that in general, it is enough to perform only *one* maxflow computation. Additionally, the average case complexity can further be reduced by detecting some special cases, when the committed flow is automatically zero.

Definition 3: Let G_f be a flow residual graph induced by a (s,d) maxflow f. Then, the *source set S* is the set of nodes which are reachable from the source in G_f and the *sink set T* is the set of nodes, from which the destination d is reachable [3].

Consider the following theorem:

Theorem 3: For some session (s,d) and some link $(i,j) \neq (s,d)$, the committed flow ψ_{ij}^{sd} can be computed as follows:

i) $\psi_{ij}^{sd} = 0$ immediately holds if either:
   1) $f(i,j) = 0$ or
   2) $i \notin S$ and $j \in S$ or
   3) $i \in T$ and $j \notin T$ or
   4) $i = d$ or
   5) $j = s$.

ii) If $i \in S$ and $j \in T$, then create a modified residual graph G'_f by removing all links $(u,v) : u \in T$ and $v \notin T$ and add a new (d,s) link of infinite capacity. Hence:

$$\psi_{ij}^{sd} = R(i,j) + F_{G'_f}(j,i) \tag{9}$$

iii) If $f(i,j) \geq F_{G_f}(i,j)$, then

$$\psi_{ij}^{sd} = f(i,j) - F_{G_f}(i,j) \tag{10}$$

Otherwise, $\psi_{ij}^{sd} = 0$.

Herein, we do not give a detailed proof of the Theorem. Most of the above statements are obvious consequences of our previous findings and some basic flow theory. Probably, only items *ii)* and *iii)* deserve more explanation. Clearly, one cannot increase the maxflow by increasing $R(i,j)$ unless there is a path from s to i and from j to d in G_f. This can only occur if $i \in S$ and $j \in T$. Additionally the maxflow from

j to i in G'_f of item *ii)* determines the upper limit of this augmentation. In turn, item *iii)* is a formal manifestation of our previous idea that (i, j) is not critical as long as flow can be shifted down from it.

Observe that only one maxflow computation per link is necessary to calculate the criticality threshold (and another one to obtain G_f, but this is common to all links). While this does not improve the theoretical worst case complexity, in practice this is a significant gain. Additionally, for a huge number of links in a realistic network one of the conditions in *i)* holds true, in which case the committed flow can be given by virtually zero computational effort. As our simulations suggest (see later), this reduces the average case complexity of the above method substantially.

IV. The Least-Critical-Path-First Routing Algorithm

In the previous section we introduced the notion of the criticality threshold and showed that it is not particularly hard to calculate this threshold. The importance of this finding is that the criticality threshold provides impressing means to characterize the actual extent of link criticality. Building on this observation, in this section we develop a new precomputation scheme for minimum interference routing, the Least-Critical-Path-First (LCPF) algorithm. The basic idea is to precompute the criticality threshold and use that sophisticated piece of information to select bandwidth-guaranteed paths for several subsequent connection requests.

In order to manifest the criticality of a link in the link weight that is used to calculate the least interfering path, MIRA applies a binary decision: a link is either critical or not. While this simple mapping efficiently transforms the momentary link criticality to link weights, it is not very well suited for precomputation. For the purposes of LCPF, we introduce a generic treatment of this mapping, the so called *cost contribution profile*. This profile describes the contribution of a session to the weight of a link as the function of the link capacity and criticality and potentially other components. Throughout this paper we use the following intuitive cost contribution profile for LCPF:

$$\kappa_{ij}^{sd} = \frac{\psi_{ij}^{sd} + D}{R(i,j)} \ , \tag{11}$$

where D units of bandwidth are requested from node a to node b. This cost contribution profile is called to implement both path feasibility and efficiency. We chose the unit contribution at the point, where sending D units of flow to the link would drive it right at the edge of criticality. This choice is called to represent effectiveness: any link with contribution less than 1 is able to tolerate the request without the risk of turning to critical. On the other hand, as the link capacity converges to zero, the contribution increases dramatically to force path feasibility. Without the loss of generality, we can assume that α_{sd} factors add up to 1, thus, for the link cost we get:

$$w(i,j) = \sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd}\kappa_{ij}^{sd} = \frac{\Psi(i,j) + D}{R(i,j)} \ , \tag{12}$$

The Least-Critical-Path-First routing algorithm (LCPF)

INPUT: A graph $G(V, E, R)$ with the set of link capacities R, a set of sessions P, a source node a and a destination node b between which a flow of D units has to be routed.
OUTPUT: The least critical path between a and b.
ALGORITHM:

1) For all $(s, d) \in P \setminus (a, b)$ and for all $(i, j) \in E$ compute the constrained maxflow $F_{G^C}(s, d)$ and the unconstrained maxflow $F_{G^U}(s, d)$. This yields the criticality threshold in the form $\psi_{ij}^{sd} = F_{G^U}(s, d) - F_{G^C}(s, d)$.
2) For all $(i, j) \in E$ compute the committed load $\Psi(i, j)$ and execute the cost assignment $w(i, j) = \frac{\Psi(i,j)+D}{R(i,j)}$.
3) Compute the shortest weighted path over the link weight set defined by w and route the demand of D units along that path.

Fig. 2. The Least-Critical-Path-First routing algorithm (LCPF)

where the so called *committed load* is defined as $\Psi(i, j) = \sum \alpha_{sd}\psi_{ij}^{sd}$ and the summation is over all (s, d) sessions in $P \setminus (a, b)$. In its simplest form the committed load is the average of the committed flows over the set of sessions. Otherwise, it reflects the relative importance of sessions through the α_{sd} factors. What remained is, similarly to MIRA, to calculate the shortest weighted path over w. This path is the least critical path from a to b (hence the name). A detailed description of the LCPF algorithm is given in Fig. 2.

Remarks:

Flow priorization: LCPF provides an easy way to reflect the precedence of sessions in the course of path selection. If for some important sessions the α_{sd} factor is set to a high value, then the critical links of that session will be given large link weight. This makes these links less probable to be selected during the shortest path computation.

Complexity: LCPF is a strictly polynomial algorithm. The computational complexity is dominated by the $2pm$ maxflow computations: 2 for every session and every link. This can be reduced to 1 maxflow computation in average using the method described in Theorem 3. The worst-case complexity is therefore $O(pn^2m^{3/2})$, which is m times as high as that of MIRA.

Criticality precomputation: in the context of this paper, real-time computations that must be accomplished upon the reception of every connection setup request are called *online* calculations. If these calculations take a long time, this may introduce unbearable latency in the process of connection setup in a highly dynamic routing environment. On the contrary, *offline* computations are performed in the background and can be scheduled without stringent real-time requirements. Criticality precomputation implies that the CPU intensive

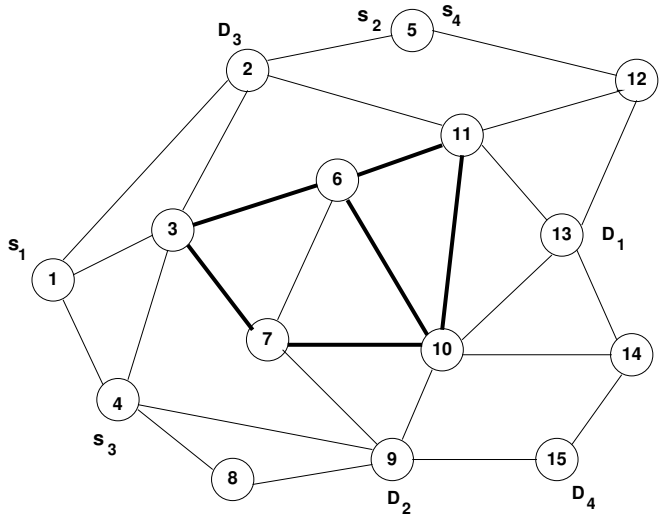

Fig. 3. The KL graph

criticality calculations (Step 1 in Fig. 2) are performed only after the arrival of every kth connection request. This converts the majority of the complexity of LCPF to offline complexity. Hence, the actual path selection for a connection setup request reduces to the calculation of w and one piece of shortest path computation in the average case.

Feasibility: one may argue that while MIRA returns a feasible path as long as such a path exists in the graph, LCPF does not. MIRA omits all links with inappropriate resources ($R < D$) before the path selection takes place. On the contrary, LCPF represents path feasibility in the contribution profile: LCPF orders high contribution to infeasible links probably yielding a path that consists of feasible edges exclusively. We can say that LCPF applies "soft feasibility" on path selection. However, since the accuracy of link state information can become doubtful, MIRA might blindly omit feasible links based on outdated knowledge on resource availability. In turn, soft feasibility may transform into better call acceptance in the presence of huge inaccuracy in network state.

Universality: it is important to emphasize that with LCPF, one is completely free to decide, which particular cost contribution profile to use. Alternative profiles may manifest different traffic engineering goals. Therefore, LCPF can be thought of as a generic framework for minimum interference routing. For example, MIRA boils down to a special case of LCPF, when used with the simple cost contribution profile:

$$\kappa_{ij}^{sd} = \begin{cases} 1 & \text{if } R(i,j) \leq \psi_{ij}^{sd} \\ 0 & \text{otherwise} \end{cases}$$

The investigation of alternative cost contribution profiles and their influence on the efficiency of routing is currently an open research problem.

V. Simulation Results

It is widely accepted that some sort of precomputation is inevitable in a realistic QoS routing environment. On the one hand, precomputation helps to reduce the online complexity of path selection. On the other hand, precomputation is a straight consequence of the nature of the underlying protocol architecture, which makes up-to-date network state information available only in discrete time intervals [9], [10], [11]. Accordingly, the frequency of link state updates induces a profound impact on the performance of minimum interference routing. First, as the network operator increases the frequency of link state updates in an attempt to reduce the implied protocol overhead, the accuracy of the routing algorithm's perception of network state becomes dubious. Secondly, it imposes a natural limitation on the frequency of link criticality computations. To manifest these effects in our simulations we defined the precomputation period k, which describes the number of subsequent calls that were processed without updating the routing algorithm's knowledge on network state and re-applying link criticality precomputation. In our simulation studies our main goal was to observe the effect of increasing precomputation periods on the performance of LCPF, MIRA, WSP[3] and SPF. Since currently there is no viable precomputation scheme for shortest-widest path routing, we omitted this algorithm in our simulations.

In the scientific literature related to MIRA the so called KL graph (Fig. 3) has slowly become the de facto simulation topology [3], [5], [7]. In the illustrative example, the capacity of the light links is 12K units and the thick links is 48K units and each link is bidirectional. In all the simulation experiments described in this paper, connection requests of uniformly distributed request size arrive and leave randomly, following an exponential-distribution.

Preserving the available transmission capacity of all sessions is the key objective of minimum interference routing. Fig. 4 shows that LCPF is indeed able to bring this policy into effect even in the presence of criticality precomputation. In the figure, the maxflow for session S1-D1 is depicted after setting up long-lived connections one by one in the KL graph. All requests are of equal size (10 units). Precomputation was not applied to MIRA and WSP, however, LCPF is run at precomputation period $k = 16$ and $k = 128$. The diagram implies that LCPF is, almost irrespective of the precomputation period, able to preserve the transmission potential of the S1-D1 session in this case.

Now we show that the deliberate criticality detection of LCPF transforms into better call acceptance in the presence of precomputation. First, we experienced how many unit-sized, long-lived connections a particular algorithm is able to accommodate in the KL graph until all sessions are blocked. Fig. 5 depicts the result of this experiment as the function of the precomputation period. Both MIRA and LCPF are able to fill up the network to the theoretical maximum (84000 units of traffic) when precomputation is not applied. Nonetheless, as the precomputation period increases MIRA (and WSP) suffers significant performance degradation but remains better than SPF (which is, by nature, not sensitive to the precomputation period). The graph also shows that both LCPF and MIRA are superior to WSP in the case of on-demand routing ($k = 1$),

[3]Note that WSP readily lends itself to precomputation (see the QoS extensions to OSPF routing protocol, QoSPF, [1, Appendices]).

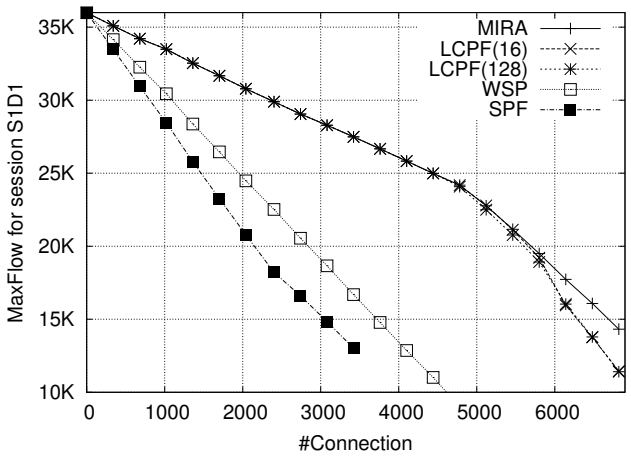

Fig. 4. Maxflow of session S1-D1 while gradually filling up the KL graph. LCPF is used with precomputation, for which the period is indicated in parentheses

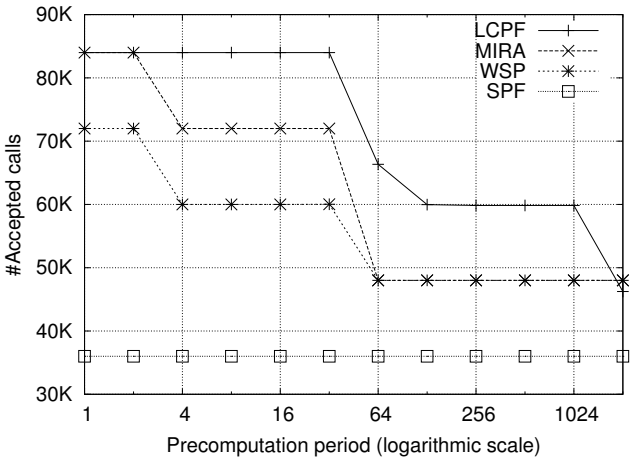

Fig. 5. Number of successfully routed connections as the function of the precomputation period

however, LCPF is able to retain the precedence as long as the precomputation period remains reasonable (take note of the logarithmic scale on the x axis).

Similar behavior can be deduced from Fig. 6, which shows the average call blocking ratio (CBR) as the function of the Poisson request arrival intensity at a precomputation period of $k = 20$. The request size was uniformly distributed between 10 and 30 units. Notably, MIRA produces non-zero call blocking even at a request arrival intensity, where WSP does not (for on-demand routing MIRA was reported to be superior to WSP [3]). Additionally, LCPF performs better than any other algorithms at all request arrival rates.

One may argue that the performance benefits implied by LCPF (and MIRA) are limited to the scope of the KL graph, and in other scenarios minimum interference routing may not prove so prosperous. Therefore, we conducted simulations on realistic random graphs generated by the BRITE tool [14] with the *router-level Waxman-model* ($\alpha = 0.15, \beta = 0.2$). Per-session Poisson request generation intensity (λ_i), exponential holding time (μ_i) and mean request size (B_i) is set as to assure that the average load is kept at a constant rate ($\sum_{i=1}^{p} B_i \frac{\lambda_i}{\mu_i} = const$). Fig. 7 depicts the average call blocking ratio as the function of increasing precomputation period averaged in 40 random graphs of 15 nodes, 45 bidirectional links and 4 sessions. Fig 8 gives the call blocking for graphs of 23 nodes, 44 links and 9 sessions. The main observations are as follows. First, when criticality precomputation is not applied WSP, MIRA and LCPF produce virtually zero blocking. In fact, MIRA generally performs slightly better than LCPF, which seems to be a consequence of the "soft feasibility" approach of LCPF. However, as the interval between subsequent link state updates and precomputations increases, MIRA loses precedence over WSP. This performance degradation may on the one hand be attributed to the inferior criticality concept implemented by MIRA. On the other hand, MIRA's undue dependency on the accuracy of link state information (recall that links deemed to be infeasible are filtered out before path selection takes place) also prove to be adverse as the consistency of network state information becomes more and more dubious. Nonetheless, with the increase of k the revenues of sophisticated criticality detection and soft feasibility of LCPF gradually emerge (SPF produces roughly 40% average call blocking ratio in both cases). For $k > 2$, LCPF outperforms all the other algorithms and preserves the same good efficiency at modest precomputation periods. Only at the extreme case of $k = 1024$, LCPF performance falls into the range of WSP and MIRA.

As the last phase of our simulation studies, we compared the online and offline complexity of WSP, MIRA and LCPF. We note that the processing requirements of these algorithms are strongly dependent on the actual topology of the network, the number of links and sessions and other factors. Furthermore, our implementation is far from being optimized and our home-grown profiler – yet, being of sub-millisecond accuracy – measures real-processing time instead of process virtual time. Therefore, the results are provided solely for informational purposes without the intention to be an exhaustive performance evaluation. We used a laptop equipped with a Mobile PIII 850 MHz processor for the experiments.

In the case of SPF and WSP, online computations basically mean to walk through the precomputed routing table, which is hardly measurable by our tools (for an in-depth evaluation of the complexity of precomputed WSP, see [15]). The same applies to precomputed MIRA and LCPF: even for the largest graphs the processing time needed to assemble the link weights and execute Dijkstra's algorithm (used for computing the shortest weighted path) remained well below the precision of our profiling tool. Thus, we decided to measure the average running time of the criticality computations in the case of MIRA and LCPF, and the calculation of the QoS routing table in the case of WSP. On the one hand, these parameters determine the offline complexity of the algorithms. Furthermore, these parameters immediately transform to online complexity, if one decides to ignore precomputation and precisely calculate a unique route for each connection setup request. The measurements were conducted in 15 sequences of increasing sized

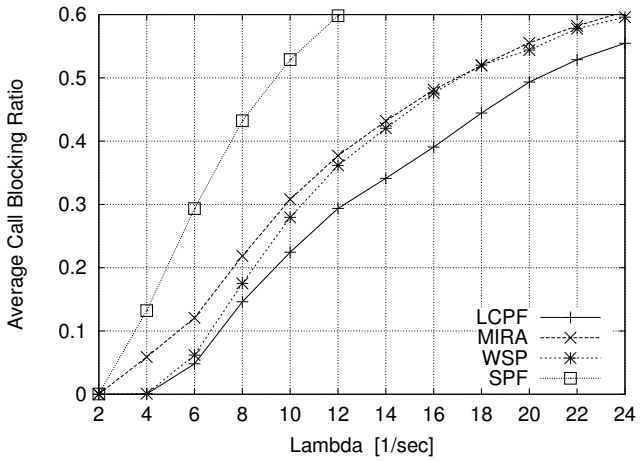

Fig. 6. Average CBR as the function of request arrival intensity. Precomputation period is 20

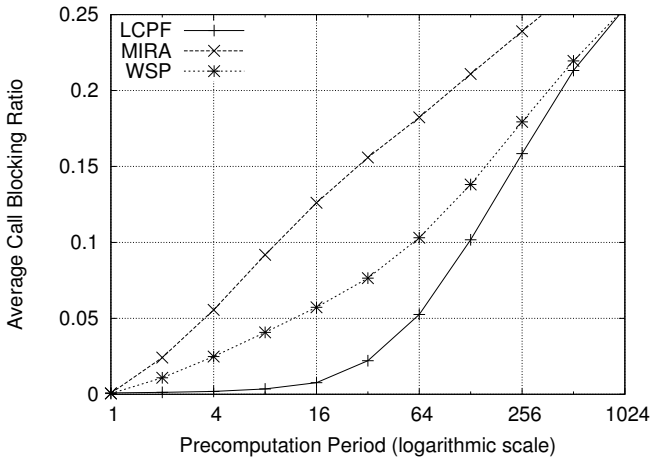

Fig. 7. Average CBR as the function of the precomputation period averaged in 40 random graphs of 15 nodes, 45 edges and 4 sessions

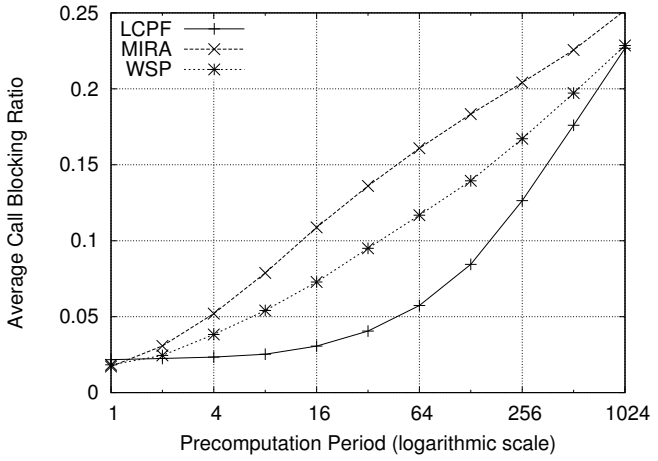

Fig. 8. Average CBR as the function of the precomputation period averaged in 40 random graphs of 23 nodes, 44 edges and 9 sessions

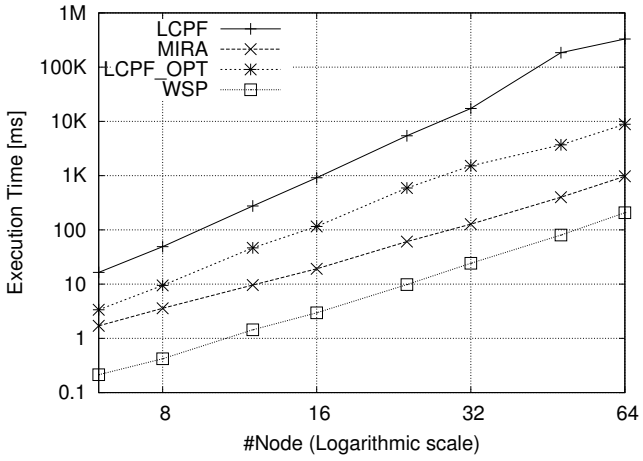

Fig. 9. Offline computational requirements of WSP, MIRA and LCPF as the function of the number of nodes in the random graphs

realistic random graphs. The number of edges was always twice the number of nodes with the number of sessions being the half. Fig. 9 shows the results (in milliseconds and in log-log space) as the function of the network size. For LCPF, we indicated the results with both the simple algorithm of Fig. 2 (LCPF) and the optimized form using Theorem 3 (LCPF_OPT).

First, WSP confirms its good record of being a lightweight algorithm: even for networks as large as 64 nodes it took less than 200 ms to compute the QoS routing table for all sessions. Second, in our specific examples, the running time of MIRA always remained under 1 second. As of LCPF, even for a small network of 16 nodes the precomputation lasts more than 1 second and the running time grows dramatically to more than 5 minutes in our largest sample configuration. This is hardly tolerable in a real network environment. However, the optimized algorithm reduces the running time to 20 ms in the 16 node graphs and to less than 10 seconds in the 64 node networks. As a rough estimation, in our experiments it took about 5 to 10 times more processing power to compute criticality with LCPF_OPT than with MIRA. Thus, when the precomputation period is increased over 10, LCPF_OPT turns to be more economical than MIRA, while at the same time, the routing performance remains on par even without up-to-date knowledge on resource availability.

VI. Concluding Remarks

Minimum interference routing was originally designed with the assumption that network state information is always up-to-date, accurate and consistent. Though, in a reality this is generally not the case, because the frequency of routing information re-synchronizations is tied up by practical limitations. Furthermore, the real-time requirements placed on path selection makes on-demand link criticality computations less appealing. In order to overcome these problems we developed a generic precomputation framework for minimum interference routing. The underlying theoretical foundations were provided by the notion of the criticality threshold, which not only gives a thorough characterization of criticality but also provides interesting new views in network flow theory. By extensive simulation studies we showed that the Least-Critical-Path-First routing algorithm exhibits unprecedented routing

performance owing to the sophisticated proactive criticality detection and soft feasibility. We found that LCPF successfully converts the online complexity of MIRA to an offline complexity of tolerable level while remaining more efficient even in the case of large precomputation periods.

REFERENCES

[1] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions." IETF RFC 2676, 1999.

[2] Z. Wang and J. Crowcroft, "Quality of Service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1228–1234, 1996.

[3] K. Kar, M. Kodialam, and T. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE Journal on Selected Areas in Communications*, vol. 18, December 2000.

[4] I. Iliadis and D. Bauer, "A new class of online minimum-interference routing algorithms," in *Networking 2002, Proceedings of Second the International IFIP-TC6 Networking Conference*, p. 959 ff., May 19-24 2002.

[5] K. Kar, M. Kodialam, and T. V. Lakshman, "MPLS traffic engineering using enhanced minimum interference routing: An approach based on lexicographic max-flow." Proceedings of Eighth International Workshop on Quality of Service (IWQoS), Pittsburgh, USA, June 2000.

[6] K. Gopalan, T. cker Chiueh, and Y.-J. Lin, "Load balancing routing with bandwidth-delay guarantees," *IEEE Communications Magazine*, vol. 42, pp. 108–113, June 2004.

[7] S. Suri, M. Waldvogel, D. Bauer, and P. R. Warkhede, "Profile-based routing and traffic engineering," *Computer Communications*, vol. 26, pp. 351–365, 2003.

[8] S.-W. Tan, S.-W. Lee, and B. Vaillaint, "Non-greedy minimum interference routing algorithm for bandwidth-guaranteed flows," *Computer Communications Journal*, vol. 25, pp. 1640–1652, November 2002.

[9] A. Orda and A. Sprintson, "QoS routing: the precomputation perspective," in *INFOCOM (1)*, pp. 128–136, 2000.

[10] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *SIGCOMM*, pp. 17–28, 1998.

[11] H. Alnuweiri, L. Wong, and T. Al-Khasib, "Performance of new link state advertisement mechanisms in routing protocols with traffic engineering extensions," *IEEE Communications Magazine*, vol. 42, pp. 151–162, May 2004.

[12] R. K. A. snd T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[13] M. S. Bazaraa, J. J. J., and H. D. Sherali, *Linear Programming and Network Flows*. John Wiley & Sons, January 1990.

[14] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal topology generation from a user's perspective," Tech. Rep. 2001-003, 1 2001.

[15] G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in *INFOCOM (2)*, pp. 680–688, 1999.