Minimum Interference Routing: the Precomputation Perspective

Gábor Rétvari

High Speed Networks Laboratory, QoS IT Laboratory
Department of Telecommunications and Telematics
Budapest University of Technology and Economics
H-1117, Magyar Tudósok körútja 2., Budapest, HUNGARY
retvari@ttt-atm.ttt.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 transmission capacity offered by the network for other communicating parties. This is achieved by circumventing some critical bottleneck links. The main contribution of the paper is a method to assess the degree of link criticality facilitating efficient route precomputation even in the case, when up to date resource availability information is not immediately available.
Keywords: QoS routing, Traffic engineering, QoS management

1 Introduction

The deliberate, effective and adaptive selection of dynamic bandwidth-guaranteed paths in modern data networks has gained substantial research interest recently. Such assured-quality communication channels are essential for the introduction of various upcoming service classes. Obviously, any service that requires the transmission of loss and/or delay sensitive data (e.g., audio, video, etc.) 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 may use as few resources as possible, or it would be such that the overall network resource utilization is maximized.

In the broader context the task of selecting feasible and efficient bandwidth guaranteed paths turns out to be the fundamental problem faced by modern QoS routing. In view of the serious scalability concerns raised by the Integrated Services approach, where bandwidth-guaranteed routes must be picked one by one for individual micro-flows, today's prevailing QoS management architecture, Differentiated Services, focuses on the assurance of aggregate QoS to a whole bunch of micro-flows. In accordance with the wider scope of routing decisions,

the emerging *Traffic Engineering* discipline puts more emphasis on the criterion of network-global efficiency. In any case, the fundamental problem is to compute bandwidth driven paths for route requests arriving one at a time. Connections are fully characterized by their respective bandwidth demand. The set of potential traffic origin-sink pairs (sessions) is assumed to be known in advance, though, we do not presume any knowledge on the volume of traffic requested for a particular source-destination pair.

Nowadays, Internet routing is based on the Shortest-Path-First (SPF) discipline: some administrative costs are associated with network links and the path with the least aggregate cost is selected (if all costs equal to 1, we get to minimum-hop routing). The SPF algorithm proved to be easily implementable and deployable, thus it gained extreme popularity in the Internet community throughout the years. However, paths selected by the SPF algorithm are usually neither feasible nor efficient, thus SPF is deemed to be a major origin of congestion and unreliability in current Internet.

Several proposals exist to eliminate the shortcomings of the SPF algorithm. It is relatively easy to ensure path feasibility. The approach that protocol designers usually take is to augment a link state routing protocol to distribute resource availability information and use this dynamic data set to produce feasible paths. The widest-shortest-path (WSP) routing algorithm selects the path from the set of shortest paths, which offers the largest bottleneck bandwidth. The QoSPF routing protocol is a good example of the practical usefulness of the WSP algorithm [1]. The advantage of the WSP algorithm (and shortest-widest-path) is the ability to find a feasible path as long as such a path exists in the network. On the other hand, the signaling bandwidth consumed by frequent link state updates may substantially enlarge the protocol overhead. Therefore, it is a widely accepted phenomenon in QoS routing to limit the frequency of link state updates by means of some link state update triggering policy and precompute QoS routes no sooner than up to date routing information becomes available [2], [3]. As of the WSP algorithm, the selection rule that decides, which particular path from the set of feasible paths is to be picked is rather simplified. Hence, the WSP algorithm often causes various sorts of *interference* phenomena, such as alternate path blocking (an extensively utilized alternate path of a session starves another one). Minimum Interference Routing is the upcoming routing technology that aims to vigorously eliminate the disadvantages of traditional QoS routing protocols [4].

The basic idea of minimum interference routing is to select paths as to ensure that the chosen path blocks future requests to the least possible extent. This is achieved by maximizing some objective function, which describes the transmission capacity offered by the network for every single session. The full-fledged minimum interference routing problem is known to be *NP hard* [4]. However, there exists an algorithm, the so called Minimum Interference Routing Algorithm (MIRA), which is aimed to give a good approximate solution to the original problem. The key point of MIRA is the notion of *critical links*. A critical link fulfills the criterion that the admission of some traffic onto the link causes

the reduction of the available capacity of one or more sessions. Hence, a critical link is subject to interference. On the other hand, if a link is not critical, then further traffic can be placed to it without adversely influencing any sessions. The rationale behind minimum interference routing lies in the ability to distinguish links based on a well-established criticality criterion and circumvent strongly critical links in the course the path selection in an attempt to minimize interference. Several recent research results have pointed out the potential of MIRA to improve network utilization while efficiently preserving resources for future requests in the same time. MIRA and some of its derivatives can be found in [4], [5]. [6] and [7].

One inherent limitation of MIRA comes from the fact that in an operational network environment MIRA's original intention to rely on the on-demand path selection model implies several drawbacks. On-demand path selection may result unbearable initial communication setup delay and the implied computational stress on routing hardware may be intolerable. In addition, the underlying routing protocol engine introduces significant inaccuracy in the routing state, because it may not re-synchronize resource availability information between the arrival of subsequent routing requests. This inaccuracy invalidates the need to calculate a unique route for each and every route request. In [4] the authors study MIRA performance in the case, when link criticality is only calculated once for every n connection request to limit the extent of calculations needed to perform in on-demand fashion. They conclude that MIRA does not suffer significant performance degradation due to *criticality precomputation*. Nevertheless, we shall show that without some clever modifications MIRA performance falls well under that of the WSP algorithm in the presence of precomputation.

This paper generalizes the notion of MIRA in a new way. The key point of MIRA is the decision, whether or not a particular network link is critical. We shall show that by the cost of some additional computational complexity introduced it is even possible to assess the extent of link criticality and use that particular information for routing purposes. Our new approach immediately gives a good algorithm to solve the problem of Δ-*criticality* left open in [4]. We shall present a new routing algorithm: the least-critical-path-first (LCPF) routing algorithm, and by means of extensive simulations we show that the proposed algorithm performs route precomputation much more efficiently, than MIRA or WSP.

The rest of this paper is structured as follows. Section 2 describes the Minimum Interference Routing Algorithm. In Section 3 we show, how to obtain a good quantity on link criticality and in Section 4 we define the LCPF algorithm that makes heavy use of this quantity. Simulation studies are presented in Section 5 and finally, Section 6 concludes our work.

2 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 rationale behind MIRA comes from the recognition that if a network link acts as 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 more interference it will cause leading to inefficient routing in the long term. Thus, it is a plausible network-wide Traffic Engineering goal to minimize the interference along the selected path.

In order to better capture the notion of interference one has to invoke the elaborated toolset of network flow theory [8] [9]. Let $G(V, E, R)$ be a digraph. Let V be the set of nodes, E the set of edges and R the set of edge capacities.[1] We assume that G is such that there is only one (u, v) edge connecting any pair of nodes $u, v \in V$ (if there are more parallel edges, these can always be aggregated into one edge). We examine flow problems for source-destination pairs (s, d), all of which are members of a known set P. Such (s, d) pairs are called *sessions* for short. Let f be a *maximum flow* in G for some session (s, d). Then, $F_G(s, d)$ notes the value of the maximum flow $|f|$, $f(u, v)$ notes the flow traversing a particular edge $(u, v) \in E$, G_f notes the flow residual graph induced by f and C_{sd} is a set of edges, which belong to one or more *minimum cuts*.

A useful feature of maxflow theory is that the maxflow of a session defines an upper limit on the available transmission capacity offered to that session. By linear programming duality, associated with each maxflow there is a minimum cut. Edges belonging to the union set of the minimum cuts (C_{sd}) share the property that decreasing the capacity of the edge reduces the maxflow. Hence, we shall say that an (u, v) edge is *critical* for a session (s, d), if the edge is included in the minimum cut set for the session, i.e., $(u, v) \in C_{sd}$. Recall that 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 the link is critical for, and computes the minimum cost path to minimize overall interference.

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

1. **Critical link identification:** compute the maxflow and the critical link set C_{sd} for each $(s, d) \in P \setminus (a, b)$. This yields cost contribution factors κ (κ describes the contribution of session (s, d) to the cost of link (i, j)) in the form:

$$\kappa_{sd}^{(i,j)} = \frac{\partial MaxFlow(s,d)}{\partial R(i,j)} = \begin{cases} 1 \text{ if } (i,j) \in C_{sd} \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

2. **Cost assignment:** for each $(i, j) \in E$ compute the link cost:

$$w(i,j) = \sum_{(s,d):(i,j)\in C_{sd}} \alpha_{sd} = \sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd}\kappa_{sd}^{(i,j)} \ , \quad (2)$$

[1] For the sake of simplicity we assume that all edge capacities are integral (either integer valued or can be expressed as an integer multiple of some bandwidth quantum). The actual implementation of the LCPF algorithm does not rely on this assumption.

where α_{sd} represents the relative importance of the session (s, d). In order to force path feasibility, a link with inappropriate resources ($R < D$) is either omitted at the SPF computation, or its cost is set to infinity.

3. **Path selection:** compute the shortest weighted path over the cost set defined by w.

The computational complexity of MIRA is dominated by the $p - 1$ maxflow computations needed for critical link identification. Without Step 1, MIRA boils down to a simple SPF calculation, which raises the idea of criticality precomputation: CPU intensive critical link identification is performed offline, once for every n route request. Setting n means to trade-off between routing accuracy and implied computational complexity.

In a situation, where one decision on link criticality may impact the routing of several subsequent requests, it would be of extreme usefulness to somehow detect if there is a chance that a link turns to critical in the near future. In this case, the routing algorithm could prevent routing further traffic onto that link to avoid interference until criticality precomputation is performed again. This brings us to the definition of Δ-criticality: a Δ-critical link for a session is a link such that if the capacity of the link is decreased by Δ, the maxflow value for the session decreases (by any value). [4] concludes that an algorithm that effectively detects Δ-criticality has the potential to increase routing performance, though, the authors do not supply an exact algorithm, neither they give an advice how to set Δ for maximum sensitivity.

3 The Criticality Threshold

In the previous section we concluded that the identification of critical links is the most important and CPU intensive process of MIRA. In this step, MIRA solves the following decision question: given an edge and a session, is it the case that the edge is included in one or more minimum cuts for the session? Throughout this paper, we use simplified form of the link criticality conditions proposed in [4]. The purpose of the alternative formulation is not really a practical one, but rather to provide further insight into the nature of link criticality.

Theorem 1 (Formulation of link criticality). *For an edge (i, j) and a session (s, d): (i, j) is critical (i.e., $(i, j) \in C_{sd}$) if all the following conditions hold:*

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

Herein we do not present the proof of the theorem. In words, Theorem 1 states that a link (i, j) is critical for a session, if it carries nonzero flow from the session's maxflow and the endpoints of the link happen to reside in different cuts of the graph (i.e., there is no path from i to j in the residual graph G_f). 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.

Definition 1. *Given a non-negative number K, a session (s,d) and an edge (i,j), K is criticality threshold, if:*

$$\textit{Criticality: } 0 < R(i,j) \leq K \Rightarrow (i,j) \in C_{sd} \ ,$$

$$\textit{Non-criticality: } R(i,j) > K \Rightarrow (i,j) \notin C_{sd} \ .$$

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 (i,j) edge would render the edge critical. This is simply done by checking $R(i,j) - \Delta \leq K$. Observe that this gives an exact algorithm to detect Δ-criticality. Furthermore, the criticality threshold supplies a unique way to assess the criticality (or non-criticality) of any link. For example, a good measure would be $K/R(i,j)$ or $K - R(i,j)$.

The approach that we take to find the criticality threshold is to investigate the maxflow problem while changing the capacity of a particular link. For a graph $G(V,E,R)$, we define the *unconstrained graph w.r.t. link* (i,j) as a graph $G^U(V,E,R^U)$, in which we choose the capacity of (i,j) to infinity. The maxflow in G^U is called *unconstrained maxflow*. On the contrary, in the *constrained graph w.r.t. link* (i,j) we set $R^C(i,j) = 0$. The maxflow in G^C is called *constrained maxflow*. One can characterize the constrained, the original and the unconstrained maxflows as $F_{G^C} \leq F_G \leq F_{G^U}$.

From Theorem 1 we know that a link, which carries nonzero flow is critical if both its residual capacity is zero and all parallel augmenting paths are saturated in the flow residual graph. This sheds light on the rationale behind the definition of the unconstrained graph: instantiating some maxflow f in the unconstrained graph for which $f(i,j) > 0$ and setting $K = f(i,j)$ will generate a candidate for the criticality threshold. Such unconstrained maxflows constitute the so called feasible maxflow set $\mathcal{F}_G^{sd}$ w.r.t. link (i,j):

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

Recall that there are no $i \to j$ augmenting paths in the flow residual graph when (i,j) is critical. In other words, from a critical link no flow can be relocated to other flow paths, which would decrease the flow on (i,j). 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.

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

$$\psi^{sd}(i,j) = \inf_{f \in \mathcal{F}_G^{sd}} f(i,j) \ . \tag{4}$$

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. The following theorem proves that the choice $K = \psi^{sd}(i,j)$ indeed yields the criticality threshold:

Theorem 2. *For an edge (i,j) and a session (s,d) in graph G:*

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

And conversely:

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

Proof. First we prove that in case of $R(i,j) = \psi^{sd}(i,j) > 0$, all conditions in Theorem 1 hold, therefore (i,j) is critical. Let $G'(V,E,R')$ be a graph, such that $R'(i,j) = \psi^{sd}(i,j)$. Then, $F_{G^U}(s,d) = F_{G'}(s,d)$. Let f be a maxflow in G', such that $f(i,j) = \psi^{sd}(i,j)$, i.e., the flow on edge (i,j) is the least possible.

i) $f(i,j) = \psi^{sd}(i,j) > 0$
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 $R_f(i,j) = R(i,j) - f(i,j) = \psi^{sd}(i,j) - \psi^{sd}(i,j) = 0$. (b) is also true, because if a $i \to j$ alternative path happens to exist, then some flow can be shifted from (i,j) to the alternative path. This yields a flow f', for which $f'(i,j) < f(i,j)$. Observe that f' is a maxflow for the unconstrained graph too, therefore $f'(i,j) < f(i,j) = \psi^{sd}(i,j) = \inf_{f \in \mathcal{F}_G^{sd}} f(i,j)$ is a contradiction.

It is also true that reducing the capacity of a critical edge both reduces the maxflow and leaves the edge as critical. This proves (5). On the other hand, for any $G'(V,E,R') : R(i,j) > \psi^{sd}(i,j)$, the (i,j) link is not critical. This is because any flow that is a maxflow in the unconstrained graph G^U is also a maxflow in G' and there exists a maxflow f such that $f(i,j) = \psi^{sd}(i,j)$, which leaves non-zero residual capacity on (i,j). This proves (6). □

A naive way to find the criticality threshold would be to search through all feasible maxflows and find the one that commits the minimum flow to the selected edge. Though, this method would not be too effective. The following fundamental flow theory result gives an easy way to compute the criticality threshold.

Theorem 3. *For a graph G, an edge (i,j) and a session (s,d), the committed flow is the difference of the unconstrained and the constrained maxflow for (i,j):*

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

Proof. We need to show that $F_{G^U}(s,d) - F_{G^C}(s,d) = \inf_{f \in \mathcal{F}_G} f(i,j)$. It is easy to show that there exists a flow f in G^U, such that $f(i,j) = F_{G^U}(s,d) - F_{G^C}(s,d)$. Now, we need to show that this is indeed the infimum. Suppose that there exists a maxflow f' in G^U, such that $f'(i,j) < f(i,j)$. Invoke flow decomposition to eliminate all flow from the graph that traverses the (i,j) edge, furthermore, remove the (i,j) edge from the graph. Observe that the resultant graph is identical to the constrained graph with respect to (i,j), and such, the resultant f'' flow is a maxflow in G^C. Hence, $F_{G^C}(s,d) = |f''| = F_{G^U}(s,d) - f'(i,j) > F_{G^U}(s,d) - f(i,j) = F_{G^C}(s,d)$, which is a contradiction. This proves that $F_{G^U}(s,d) - F_{G^C}(s,d)$ is indeed the infimum. □

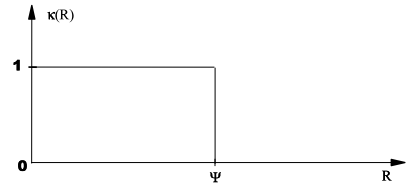

Fig. 1. Cost contribution profile in standard MIRA

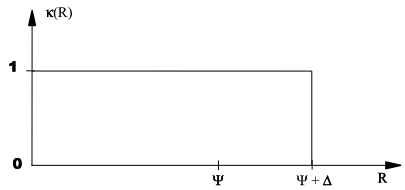

Fig. 2. Cost contribution profile for Δ-criticality

4 The Least-Critical-Path-First Routing Algorithm

So far we have seen that the cost of a link is composed of $\alpha_{sd}\kappa_{sd}^{(i,j)}$ per-session contributions. The notion of criticality threshold provides a comprehensible way to represent the cost contribution as the function of the link capacity. Such a $\kappa(R)$ graph is called cost contribution profile. The cost contribution profile for standard MIRA is depicted in Figure 1. The cost contribution profile used to detect Δ-criticality is shown in Figure 2. Note that MIRA does not compute the criticality threshold, instead, MIRA solves the decision question using an alternative of Theorem 1.

Nevertheless, the notion of criticality threshold elaborated in the previous section provides an impressing mean to not only solve the decision question, but to even calculate a quantity on the actual extent of link criticality. The basic idea of the least-critical-path-first (LCPF) routing algorithm is to determine the criticality threshold and use that sophisticated piece of information to compute bandwidth-guaranteed dynamic paths. In order to manifest the criticality of link (i,j) for session (s,d), we use the following intuitive cost contribution profile:

$$\kappa(R) = \frac{\psi_{sd}(i,j) + D}{R(i,j)} \ , \tag{8}$$

where the requested bandwidth is D units. Recall that the cost contribution profile should implement both path feasibility and efficiency. We chose the unit contribution at the point, where sending D flow to the link would practically 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 a request of D size 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 restriction of generality, we can assume that $\sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd} = 1$, thus, for the link cost we get:

$$w(i,j) = \sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd} \frac{\psi_{sd}(i,j) + D}{R(i,j)} = \frac{\Psi(i,j) + D}{R(i,j)} \ , \tag{9}$$

where $\Psi(i,j)$ is the so called *committed load*. In its simplest form the committed load is the average of the committed flows over the set of sessions ($\alpha_{sd} = \frac{1}{p-1}$). Otherwise, it represents the relative importance of sessions as well.

Hence, (9) yields the Least-Critical-Path-First-Routing Algorithm:

Least-Critical-Path-First Routing Algorithm (LCPF):

INPUT: A graph $G(V,E,R)$ with the set of link capacities R, a set of potential sessions P, an ingress node a and an egress node b between which a flow of D units have to be routed.

OUTPUT: The least critical feasible path between a and b.

ALGORITHM:

1. For all $(s,d) \in P$ 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 $\psi_{sd}(i,j) = F_{G^U}(s,d) - F_{G^C}(s,d)$.
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

Remarks:

Flow priorization: the algorithm facilitates for the network operator to represent his or her precedence of sessions by setting α factors accordingly. If the α factor is set to a high value for some important session, then the LCPF algorithm will do its best to circumvent the critical links of that session.

Complexity: according to [4] the computational complexity of MIRA is dominated by the set of $p-1$ maxflow computations. LCPF involves $2P|E|$ maxflow computations (2 for every session and every link), thus, its worst-case complexity is $|E|$ times as high as that of MIRA. For the first glance, the huge number of necessary maxflow computations is disappointing. However, it must be mentioned that in its current form, LCPF lends itself to various sorts of optimizations, which have not been exploited herein (for example, $\psi_{sd}(i,j) = 0$ for any link, such that $j = s$, or $i = d$, etc.). Therefore, we developed an optimized algorithm that provides an average running time comparable to that of standard MIRA, though, the detailed discussion of this algorithm is beyond the scope of this document. For now, it suffices to claim that the worst case complexity of LCPF is no worse than $|E|$ times of MIRA's complexity

Criticality precomputation: criticality precomputation implies that the CPU intensive criticality calculations are performed only after every nth connection request. As we shall see in the next section, simulation results obtained on large graph sets demonstrate that owing to the sophisticated "criticality-detection" LCPF significantly outperforms MIRA in the presence of criticality precomputation. Hence, reducing LCPF to a simplistic shortest path

computation in the average case and doing criticality computations offline makes the LCPF algorithm an overly effective and lightweight solution.

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, which probably yields a path consisting of feasible edges exclusively. We can say that LCPF applies "soft feasibility" on path selection. In the presence of criticality precomputation the accuracy of link state information is doubtful, so MIRA might blindly omit feasible links based on outdated link state information. Therefore soft feasibility transforms into better call acceptance in such cases.

We say that LCPF is a generalization of MIRA, as MIRA boils down to a special case of LCPF when used with the cost contribution profile depicted in Figure 1.

5 Simulation Results

Several research results [3], [10] demonstrate that in a realistic QoS routing environment some sort of precomputation is inevitable due to the inherent nature of the underlying protocol architecture. Therefore, in the course of the simulation studies presented in this section our main goal was to compare the routing performance of the LCPF algorithm to that of MIRA, WSP and SPF in the presence of route precomputation (for details on precomputed WSP, please refer to the Appendices of [1]).

In the scientific literature related to MIRA the so called KL graph (Figure 3) has slowly become the de facto simulation topology [7], [5], [4]. In the illustrative example, the capacity of the light links is 12K units and the dark links is 48K units and each link is bidirectional. In all the simulation experiments described in this paper, requests are uniformly distributed over all sessions and arrive randomly at the same average rate.

Preserving the available transmission capacity offered for communicating sessions is the key objective towards minimum interference routing. Figure 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 $n = 16$ and $n = 128$. The diagram implies that LCPF is, almost irrespective of the precomputation period, able to preserve the transmission potential of the S1-D1 session, thus it manifests minimum interference almost as effectively, as MIRA in this case.

Now we show that in the presence of criticality precomputation the deliberate criticality detection of LCPF transforms into better call acceptance. First, we experienced how many unit-sized, long-lived connections a particular algorithm is able to accommodate in the KL graph one after another until all sessions are

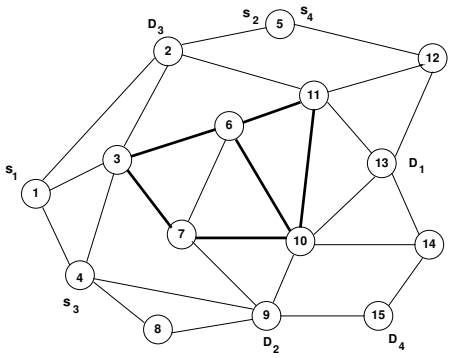

Fig. 3. The KL graph

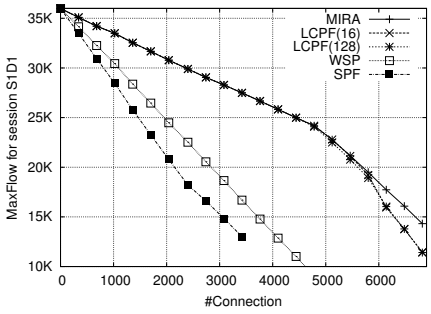

Fig. 4. MaxFlow for session S1-D1 after every connection is routed

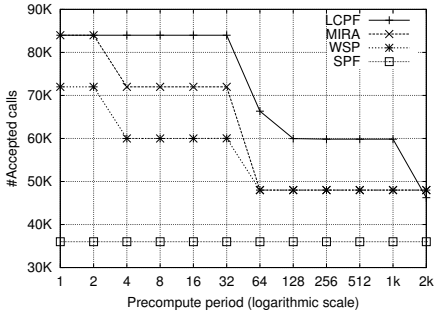

Fig. 5. Successfully routed connections as the function of the precomputation period

blocked. Figure 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, however, as the precomputation period increases MIRA (and WSP) suffers significant performance degradation. Nonetheless, regardless of the degree of applied precomputation, each algorithm outperforms 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 ($n = 1$), 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 Figure 6, which shows the average call blocking ratio (CBR) as the function of the Poisson request arrival intensity at a precomputation period of $n = 20$. The request size was uniformly distributed between 10 and 30 units. MIRA performs reasonably better than WSP for $n = 1$ [4]. However, when $n = 20$ MIRA produces non-zero call blocking even at a

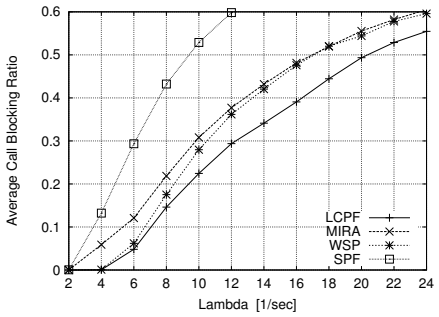

Fig. 6. Average CBR as the function of request arrival intensity

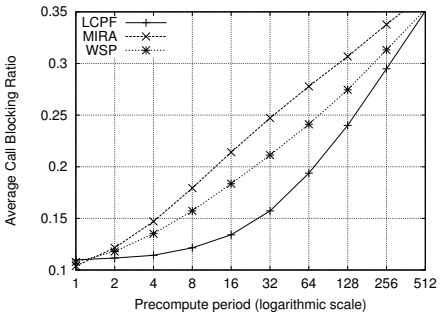

Fig. 7. Average CBR in 30 pseudo-random graphs as the function of n

request arrival intensity, where LCPF or WSP does not. LCPF performs better than any other algorithms at all request arrival rates.

One may argue that the performance benefits implied by LCPF (and MIRA) may be limited to the scope of the KL graph, and in other scenarios minimum interference routing may not prove so prosperous. Therefore, we carried out simulations on a set of 150 random graphs, which were obtained by relocating links in the KL graph. In all random graphs it was assured that there are at least two link-disjoint paths between any particular session source and destination. A random number of sessions ($3 \leq p \leq 7$) was located randomly in the graphs. 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 ($\sum_{i=1}^{p} B_i \frac{\lambda_i}{\mu_i} = const$). Figure 7 depicts the average call blocking ratio (precomputation period is again represented in a logarithmic scale). First, in the absence of criticality precomputation, MIRA and LCPF produces similar outstanding routing performance. In fact, MIRA performs slightly better, which seems to be a consequence of the "soft feasibility" approach of LCPF. However, as the interval between subsequent link state updates increases (as it would be the case in a realistic traffic engineering environment), and hence the effects of precomputation efficiency become dominant, MIRA loses precedence over WSP and the revenue of sophisticated criticality detection and soft feasibility of LCPF emerges (SPF, irrespective of n produces 0.48 average call blocking ratio). For $n > 2$, LCPF outperforms every other algorithms and preserves the same good efficiency at modest precomputation periods. Only at the extreme case of $n = 1024$, LCPF performance falls into the range of WSP and MIRA.

6 Conclusions

In this paper we investigated the impacts of precomputation on several QoS routing algorithms. By means of extensive simulation studies we have shown that the overly effective MIRA algorithm is not sufficient for route precomputation,

though, precomputation is an inevitable and straight consequence of the underlying protocol architecture. Therefore, we elaborated the notion of committed flow and criticality threshold and showed that these are fundamental properties in network flow theory. We exploited the potential of the criticality threshold to implement sophisticated proactive criticality detection in order to design the least-critical-path-first routing algorithm. We have shown that the LCPF algorithm manifests the minimum interference policy more deliberately than MIRA even in the case of large precomputation periods. Nevertheless, we concluded that the new routing algorithm is far more expensive in terms of offline computational requirements than MIRA, though, further work is necessary in this field.

Acknowledgement

This work has been done within the research cooperation framework between HSNLab at DTT-BUTE and Ericsson Research. The author is grateful to Miklós Boda (Ericsson), Tamás Henk and Tibor Cinkler (HSNLab) for their support.

References

1. R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions." IETF RFC 2676, 1999.
2. A. Orda and A. Sprintson, "QoS routing: the precomputation perspective," in *INFOCOM (1)*, pp. 128–136, 2000.
3. G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in *INFOCOM (2)*, pp. 680–688, 1999.
4. 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.
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. 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.
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. R. K. A. snd T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
9. M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. John Wiley & Sons, January 1990.
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.