HCASP: A Hop-Constrained Adaptive Shortest-Path Algorithm for Routing Bandwidth-Guaranteed Tunnels in MPLS Networks

Khaled M. F. Elsayed, *Senior Member, IEEE*
Department of Electronics and Communications Engineering, Cairo University, Egypt 12613
khaled@ieee.org

Abstract

We present an efficient low-cost algorithm for routing of MPLS bandwidth-guaranteed tunnels in general topology networks. The HCASP algorithm tries to achieve two objectives: the first is to limit the length of the chosen path for a certain tunnel so as not to largely exceed the length of the shortest-hop path between the ingress-egress pair; the second objective is to avoid over-loaded links at the time of tunnel establishment. Two popular schemes for routing of bandwidth-guaranteed tunnels in MPLS networks are the minimum interference routing algorithm (MIRA) and widest-shortest path routing (WSP). MIRA is known to provide excellent performance at the expense of solving a large number of maxflow problems each time a tunnel is routed. Using extensive simulation for general network topologies, we show that HCASP outperforms both MIRA and WSP for networks with a low degree of connectivity and large network diameter, whereas MIRA is better (but not significantly better than HCASP) for networks with high degree of connectivity and small network diameter. Moreover, HCASP can be applied in a distributed fashion using source routing, whereas MIRA is suitable for centralized implementation. Another advantage for HCASP is that it has a much lower computational complexity than MIRA.

1. Introduction

We consider the problem of establishing bandwidth guaranteed services which imply the dynamic set-up of a bandwidth guaranteed path between a network's egress-ingress router pair. Although, the establishment of bandwidth guaranteed paths are encountered in various situations (for example ATM, Frame relay, ..), we assume the context of multi-protocol label switching (MPLS) IP-based networks. In this case the bandwidth guaranteed paths are MPLS bandwidth guaranteed label switched paths (LSP) or tunnels. (We will use the terms LSP or tunnel interchangeably in the paper to essentially refer to the same thing.)

This problem is a special case of quality-of-service (QoS) routing with a constraint on the minimum bandwidth available on the path. The problem has been studied extensively and various schemes have been proposed to solve it. Some of the well-known schemes are those based on shortest-hop routing with a constraint on the path's minimum available bandwidth such as widest-shortest path (WSP) [1], shortest-widest path (SWP) [2], and others. An overview and comparative study has been reported in [3] which shows the superior performance of WSP and another algorithm based on setting link cost inversely proportional to remaining capacity on the link. The main problem of WSP and its variants is that they do not consider future LSP requests at all. This problem was addressed by minimum-interference routing.

Kar et al. [4] proposed the minimum interference routing algorithm (MIRA). MIRA attempts to reduce the possibility of selecting a route that "interferes" with future requests of other ingress-egress pairs. This is achieved on solving a maxflow problem for the other ingress-egress pairs in the network, identifying the critical links belonging to the mincut and assigning higher weights for these critical links. With these link weights, the Dijkstra algorithm, is applied on the graph. Due to the link weights being assigned in proportion to their criticality level, the critical links for other ingress-egress pairs are avoided and the overall network throughput eventually increases. It is shown in [4] that MIRA provides superior performance to WSP. However, the study is done for a single network topology with a small number of egress-ingress pairs. MIRA has been extended in [5] to the case of routing of restorable bandwidth guaranteed tunnels where a path is associated with a backup path.

A related scheme is MATE reported in [6]. Elwalid et al. propose a traffic-engineering scheme for MPLS networks that can be applied by measuring network state at ingress points. The scheme is applicable to MPLS LSPs with best-effort service requirements and the objective is to minimize the average delay in the network. Salvadori et al. [7] proposed a load balancing algorithm for MPLS networks.

We present an efficient low-cost algorithm for routing of MPLS bandwidth-guaranteed tunnels in general topology networks. The HCASP algorithm tries to achieve two objectives: the first is to limit the length of the chosen path for any LSP so as not to largely exceed the length of the shortest-hop path between the ingress-egress pair of the LSP; the second objective is to give preference to less loaded links at the time of LSP establishment. These two objectives help to reduce the unbalance in resource utilization in the network and therefore result in reduction of blocking probability of new LSPs..

2. Problem Definition and System Model

We consider a network with arbitrary topology represented by a connected undirected graph *G(V, E, C)* with a set of vertices *V* ={1, 2, …, N} representing the network nodes (i.e. the routers) and a set of edges *E*={1, 2, …, M}, representing the physical links connecting the nodes. Each edge $e \in E$ has a capacity c_e expressed in bits/sec. The set of values c_e comprise the set *C*. If there is an edge between node *i* and node *j* in *G*, it implies the existence two links in the network: one between *i* and *j* and the second between *j* and *i* with both links having the same capacity. Define $C_{\max} = \max\{c_e\}, e \in E$ and $C_{\min} = \min\{c_e\}, e \in E$.

We define the degree of connectivity of the graph as *DC=2M/N*. The value of *DC* gives an indication of how well connected the graph is. A full mesh will have *M=N(N-1)* links, and *DC* will be equal to *2(N-1)* while for a ring (a common example of low connected graphs), we have *DC* = 2. Furthermore, define h_{ij} as the length of the path with the minimum number of hops between nodes *i* and *j* both of which is in the set *V*. We define the graph diameter $GD = \max_{i,j \in V} h_{ij}$. The value of *GD* also provides an indication of how well connected the graph is: a full mesh will have a *GD* value of 1, while a ring will have a *GD* value of *N-1*. A subset of the nodes is assumed to be core routers where no external traffic is generated. The core routers only forward traffic of other nodes to their ultimate destination. The rest of the nodes are called ingress-egress routers where traffic is generated or delivered to the final destination. Let the set of ingress-egress nodes be denoted by V_{IE} whose granularity is given by $N_{IE} \le N$. We further assume that any node in the set V_{IE} may establish communication with any other node in V_{IE} (except itself). Therefore, the number of ingress-egress pairs in the network is equal to $L = N_{IE}(N_{IE} - 1)$. Let the set P be the set of all allowable ingress-egress pairs.

Requests to establish LSPs of guaranteed bandwidth arrive randomly to the set of ingress-egress routers. Each LSP *k* will be defined by the tuple (i_k, j_k, b_k) where i_k is the ingress router where the LSP originated, j_k is the egress router of the LSP and b_k is the amount of bandwidth requested by the LSP. We note that b_k must be less than or equal to the smallest link capacity in the network. In other words, we do not allow LSP bifurcation to occur. For a LSP such as *k* defined above, we note that a symmetric reservation is needed one from i_k to j_k and the second from j_k to i_k.

A link state routing protocol such as OSPF is used in the network to collect information about the links and the amount of reserved bandwidth on any link. Such information can be maintained using extensions of OSPF to support quality-of-service routing [8].

The problem is now defined as follows: select a path for an incoming LSP *k* that has enough available capacity to accommodate b_k such that the overall network blocking probability is minimized. Since the LSPs are symmetric, the routing algorithm we consider use the same path for the forward path from i_k to j_k and the reverse path from j_k to i_k. If the algorithm decides that a suitable route can be established, a signalling protocol such as RSVP [9] or LDP [10] is used to setup the path and reserve the bandwidth on each link and router on the path. The amount of the available bandwidth on all links on the chosen path is decremented by the amount b_k. The nodes in the network keep a record of the overall amount of reserved bandwidth on their links. Let r_e be the amount of bandwidth reserved on link $e \in E$, define the sets $R = \{r_e\}, e \in E$ and $U = \{u_e\}, e \in E$ where $u_e = r_e / c_e$ is the utilization of link $e \in E$.

3. The Hop-constrained adaptive shortest-path algorithm

For overview of the WSP and MIRA algorithms, we refer the reader to [4]. There are several problems with MIRA some of which have been identified in [11]. Some counter examples in which MIRA will perform poorly were identified, particularly for concentrator and distributor networks. This was attributed to the fact that MIRA focused exclusively on the interference on single ingress-egress pair not a on clusters of node. Also, the hop count is not considered (although [3] pointed out that hop count can be taken into account by using the Bellman-Ford algorithm instead of Dijkstra). (Our initial assessment of MIRA showed that indeed in some cases it could choose a path which is significantly longer than the shortest-hop path.

The main problem we further identify with MIRA and its derivatives are as follows. In all studies based on MIRA the number of ingress-egress pairs used is very limited. In a realistic setting, where in the limiting case all nodes in the network can be ingress-egress pairs, the amount of computations needed to solve the maxflow problem for all pairs in the network will render MIRA very expensive to deploy. This is also magnified by the nature of the algorithm which dedicates centralized execution at a route server. This leads to the second major drawback which is its inability to be applied in the destination-based distributed routing paradigm of the Internet or source routing. To reduce computation overhead, [3] suggests performing the link cost calculation not for every LSP request, but for after a number of LSP requests have been routed. This could be reasonable

for a small sized network, however, for a large network with a large number of ingress-egress pairs, it does not make sense at all to use the same link cost setting for ingress-egress pairs which might belong to significantly different locations within the network. (Actually, we have tested MIRA when the link cost is updated once every 50 and once every 100 LSPs have been routed and found that the performance really suffers and could make MIRA perform comparably to WSP with WSP having a much lower computational complexity.)

To avoid these problems, the main objectives sought in the design of HCASP are the following:

1- When selecting a path try to avoid bottleneck links. This is achieved by setting the link cost as function of its current amount of available capacity and utilization level.
2- Attempting to avoid link cost functions that change frequently to reduce the signalling overhead needed to send the link cost updates or those that change dramatically with small load increase as this may cause the network to become unstable (especially if used in conjunction with routing of best-effort traffic).
3- Another important feature is that for two links with the same utilization level, a link with higher capacity should be preferred over a link with a smaller capacity.
4- Avoid long-hop paths as much as possible. This is achieved by constraining the length of a selected paths to never be longer than shortest-hop distance between an ingress-egress pair plus a certain constant that is a function of the network degree of connectivity.
5- It should not have a high computational complexity and be based on Internet link-state routing protocols such as OSPF, can be applied using source routing or distributed routing and can be implemented using well-known path selection algorithms such as Bellman-Ford or Dijkstra's algorithms.

A key factor in achieving the above objectives is the choice of a cost function. We choose the cost link function as follows:

$$w(e) = \left\lfloor \frac{C_{\max}}{c_e} \frac{1}{1-u_e^x} \right\rfloor \quad (1)$$

where $C_{\max}$ is the maximum link capacity in the network, C_e is the link capacity of link *e*, and u_e is the utilization of link *e*. The factor $\frac{C_{\max}}{C_e}$ will give preference towards using higher capacity links, while the term $\frac{1}{1-u_e^x}$ will make highly utilized links (potential bottlenecks) less attractive to the path-selection algorithm. The exponent value could be any number $x \geq 1$. We choose $x = 4$ in our implementation. The choice of $x = 4$ is completely heuristic. We experimented using several values and noticed that using small values provides a cost function that changes abruptly with increasing link utilization and this leads to worse performance. Also, using large values of *x* would make $w(e)$ change rather slowly with the increase of the utilization and thus the scheme becomes less load-sensitive and the response to overload links can become somewhat late. The floor operation will serve to smooth the variation in the link cost due to small increases in the reservation levels on the link (which is very important for routing of best-effort traffic if simultaneously used in the network). This also keeps the link cost to an integer value which is the case for OSPF link cost setting. To remain compatible with OSPF, in the implementation we set the link cost by using $w(e) = \min(\left\lfloor \frac{C_{\max}}{c_e} \frac{1}{1-u_e^4} \right\rfloor, 65535)$, since OSPF uses 65535 as the maximum possible link capacity.

To satisfy the objective of not using long paths, we use an algorithm we call the hop-constrained Bellman-Ford (HCBF) algorithm. We set and setting an upper bound on the number of iterations equal to the desired constraint on the number of hops. The Bellman-Ford algorithm is based on dynamic programming and it starts by finding shortest one-hop path between any two nodes, then the shortest path with up to two hops, and so on. It stops if the cost of the shortest-path for all ingress-egress pairs does not change in two successive iterations or if it reaches the maximum possible distance between any two nodes which is *N-1*. The BF algorithm can easily be modified to limit paths to certain length by stopping iterations at a certain preset limit.

Setting the HCASP Hop Constraint: Consider an LSP described by (i, j, b) arriving to a network with graph representation *G(V, E, C)*. Let h_{ij} be the length of the shortest-hop path between *i* and *j*. HCASP then will not allow any path between *i* and *j* be longer than min(h_{ij} + *GD, N-1) where GD* is the graph diameter. This choice is based on two facts: Firstly, the setting of h_{ij} + *GD* allows flexibility in the path choice and not being too restrictive. If a path being selected between *i* and *j* finds the cost of the shortest-hop path high, then it is allowed to use intermediate nodes whose shortest-hop path does not currently have a high cost. But the length of any other shortest-hop path is constrained by GD. The choice of a maximum path length of h_{ij} + *GD* is thus logical and flexible. Secondly, it is known that in a connected graph with *N* nodes, the longest possible hop count between any two nodes is *N-1* which is also the maximum number of iterations needed by the Bellman-Ford algorithm, so taking the minimum of h_{ij} + *GD* and *N-1* insures a feasible upper bound for cases where the two nodes have a long shortest-hop path.

The above choice for the maximum allowable path length works well in practice as will be shown in the results section. This is particularly clear for high load situations when it becomes critical to limit the path length since long

paths will increase the overall network utilization by reserving bandwidth on all links selected for the path. We believe there could be better choices for the maximum allowable path length between two nodes. We could attain a superior performance by choosing a maximum path length which is h_{ij} + *EH(i, j, G, U), where EH* is the allowable number of extra hops which should optimally be a function of the graph topology *G* and the position of both *i* and *j* in *G*, and the current network loading state *U*. We believe, however, that the choice of a function for *EH* is a very complex process.

The source-routing HCASP algorithm is specified as follows:

INPUT: 1) A graph *G(V, E, C)* with current reserved bandwidth (*R*) on each link, the weight set *W*, the distances matrix $H=[h_{ij}]$ and graph diameter *GD*. (Note that each node calculates and stores H and GD at initialisation time.) *2) A LSP request specified by* (i, j, b).

Output: A path between *i* and *j with available capacity* $\geq b$ and new values for *R* and *W*

Algorithm:

- At ingress node *i*, eliminate all links from the graph with remaining capacity $< b$.
- At ingress node *i*, execute the HCBF algorithm to find shortest path between *i* and all other nodes in the networks including node *j* with the a maximum of min(*N-1*, h_{ij} + *GD)* iterations.
- If a feasible path is found, ingress node *i* sends a reservation request for all intermediate routers between *i* and *j* to commit the bandwidth *b* for the LSP in the forward and reverse directions. This is done in a node-by-node manner. Node *i* starts by sending the reservation to the next router in the path which forward it to the next router and so on.
- At any node receiving the reservation, if the requested bandwidth is still available, the amount of bandwidth is tentatively reserved and the reservation request is forwarded to the next router.
- If the reservation request reaches egress node *j*, it sends a commit request to all intermediate nodes back to ingress node *i*. The reserved bandwidth is then committed and all nodes send links state updates containing the new value of reserved bandwidth by flooding the network.

We should also note here that at the time a LSP is terminated, the reserved bandwidth over the LSP's path is released and link state updates are sent to reflect the new values of the link reservation.

4. Performance Evaluation

To test the performance of the HCASP algorithm against MIRA and WSP, we use discrete-event simulation with different network topologies and traffic loading For each simulation, we obtain and report the following performance metrics: 1) The LSP blocking probability for MIRA, HCASP, and WSP, 2) The absolute and relative difference in the blocking probability between MIRA and HCASP, 3) The percentage of LSPs routed along a path with 50% more hops than shortest-hop path. We also obtain the average shortest-hop path length between the ingress-egress pair of accepted LSPs and the average bandwidth of accepted LSPs. We use three network topologies with varying characteristics to be able to distinguish between the performance of the algorithms under different setting for network connectivity, number of ingress-egress pairs, network diameter and so on. The used network topologies with their parameters and the ingress-egress and core nodes are shown in *Figure 1*. We use two values for link capacities, 622 Mbps which are primarily used to connect core nodes or core nodes to certain critical ingress-egress nodes, and 155 Mbps links. The well-connected topology is the one used in [3], however, we increase the number of ingress-egress pairs to include all the nodes that are part of the four ingress/egress pairs used in [3].

We build the discrete-event simulation using the Simpack [12] library to manage the event list containing LSPs arrival and departure information. Each simulation is replicated at least 7 times to obtain confidence intervals and all simulation runs are done for at least 200,000 LSPs. For lower value of LSP arrival rate, we tend to increase the number of LSPs until we notice a LSP blocking event (up to an upper bound on number of LSPs generated to avoid infinite loops). We do not report the confidence intervals in the results, as they were very narrow except for blocking values less than 10^{-4}. We start from an empty network and we set a warm-up period of 10,000 LSPs where no statistics are collected. The LSPs arrive in accordance with a Poisson distribution with rate λ LSPs/unit time and last for an exponential duration with a mean of 1 unit. In all simulations, LSPs request a bandwidth *b* equally likely from the set {1, 2, …, 8} Mbps.

In the case when the overall LSP request arrival rate is divided equally among the ingress-egress pairs, we say that we have uniform traffic. For all experiments, we report the blocking probability, the difference in blocking probability between MIRA and HCASP, the relative difference in blocking probability between MIRA and HCASP (defined as the blocking probability difference divided by the blocking probability of MIRA), and the fraction of long paths (when an LSP is routed along a path that is at least 150% longer than shortest-hop path, we say it has a long path).

We start by comparing the performance of MIRA, HCASP, and WSP for the well-connected mesh network. The arrival rate is increased from 155 to 288 LSP

requests/sec. From *Figure 2*, we note that MIRA is the best performing algorithm followed closely by HCASP. We report in part (b) of *Figure 2*, the absolute difference in blocking probability between HCASP and MIRA. The maximum difference is 18% less blocking by MIRA at low loads. We also see in part (c) of the figure that indeed, MIRA may result in significantly longer paths than shortest-hop paths. For example, at R=150 LSPs/sec, 30% of the accepted LSPs are routed along a long, while HCASP causes only 6% of the LSPs to go through such long paths.

For the ring-like topologies and the low-connected mesh, we find that HCASP outperforms both WSP and MIRA as shown in *Figure 3* and *Figure 4*. In the few loading values when MIRA is better, the performance of the two schemes is quite close as in the well-connected mesh. For some points, WSP offers better performance. However, if we average over all loading values, we find that HCASP offers the best overall performance.

We have also made the same experiments but with MIRA updating the link cost periodically after each 50 or 100 LSPs are routed. We found that such delayed updates significantly impact the performance of MIRA. Using such a setup, HCASP outperforms MIRA in almost all loading and for all network topologies.

Table 1. Execution times for HCASP and MIRA

	Avg. Time in seconds for 10,000 LSPs establishment	Avg. LSPs/sec
Well-connected Mesh--HCASP	2.20	4,549.59
Well-connected Mesh--MIRA	69.29	144.33
Low connected Mesh--HCASP	3.27	3,054.37
Low connected Mesh--MIRA	108.53	92.14
Ringlike--HCASP	1.05	9,505.70
Ringlike--MIRA	137.22	72.88

We also report the execution time for routing 10,000 connections using HCASP and MIRA. We performed the experiments on an IBM NetVista with Pentium II 866 MHz, 128 Mbytes of RAM, and 256 Kbytes cache running RedHat Linux 6.2 and the source compiled with the GNU C/C++ compiler version 2.91.66. We report the recorded total time in Table 1 which shows that HCASP is orders of magnitude faster specially for the ringlike network where the number of ingress-egress pairs is large. Also, it is clear MIRA can not sustain a large throughput in connections/sec requests. If MIRA is implemented as suggested in [3] as a route server that makes all routing decisions, either a very fast machine is needed or the network is mainly deployed in an environment where long-lived connections are expected to dominate the service profile.

5. Conclusions

We presented HCASP as an efficient dynamic algorithm based on the popular shortest-path routing for path selection of bandwidth guaranteed tunnels in MPLS networks. The algorithm compares favorably with MIRA [3] which is one of the best known algorithms in this category. While MIRA aspires to prevent the occurrence of bottlenecks, HCASP is a reactive scheme that avoids selecting bottleneck links for new LSP requests. HCASP outperforms MIRA for low-connected networks and has much less computational complexity. Moreover, it can be used in mixed routing environment where both best-effort traffic and MPLS LSPs with guaranteed bandwidth requirement can both exploit the link weight function proposed for HCASP. Also, HCASP can be applied using distributed source routing.

Extension to routing of protected or restorable LSPs is a natural step. Also, integrated routing of IP/MPLS over a WDM network can be pursued. We are also investigating the feasibility of implementing HCASP in hop-by-hop distributed routing environment along the lines proposed by Wang and Nahrstedt in [13].

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi. Quality of service based routing: a performance perspective, Proceedings of SIGCOMM'98, Computer Communication Review, Volume 28, Number 4, October 1998.

[2] Z. Wang and J. Crowcroft, QoS routing for supporting multimedia applications, IEEE JSAC, Vol. 14, pp. 1228-1235, September 1996.

[3] Q. Ma and P. Steenkiste, Quality-of-service routing with performance guarantees, 4th International IFIP Workshop on Quality of Service, May 1997.

[4] K. Kar, M. Kodialam, and T. V. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications, IEEE JSAC, Vol. 18(12), pp. 2566-2579, Dec. 2000.

[5] M. Kodialam and T. V. Lakshman, Dynamic routing of restorable bandwidth guaranteed tunnels using aggregated network resource usage information, IEEE/ACM Transactions on Networking, vol. 11, no. 3, June 2003.

[6] A. Elwalid, C. Jin, S. Low, and Indra Widjaja, MATE: MPLS adaptive traffic engineering, IEEE INFOCOM 2001, pp. 1300-1309, April 2001.

[7] E. Salvadori, Elio and R. Battiti, A load balancing scheme for congestion control in MPLS networks, Technical Report DIT-02-95, Informatica/Telecomunicazioni, Università degli Studi di Trento, 2002.

[8] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, QoS extensions to OSPF, Work in Progress.

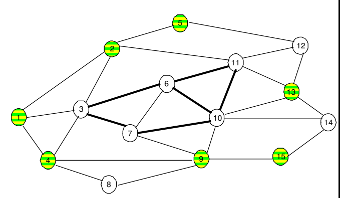
[9] R. Braden (Ed.), Resource ReSerVation Protocol (RSVP), Version 1 Functional Specification, Internet RFC 2205.

[10] B. Jamoussi (Ed.), Constraint-Based LSP Setup using LDP, Internet RFC 3212.

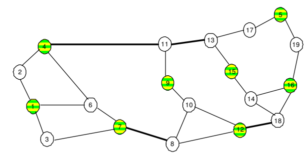
[11] B. Wang, X. Su, and C. Chen, A new bandwidth guaranteed routing algorithm for MPLS traffic engineering, IEEE ICC02, 2002.

[12] P. Fishwick, The Simpack ToolKit, available from http://www.cise.ufl.edu/~fishwick/simpack.html..

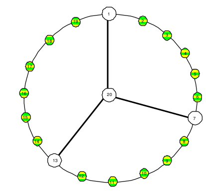
[13] J. Wang and K. Nahrstedt, Hop-by-hop routing algorithms for premium-class traffic in DiffServ networks, IEEE Infocom'02, April 2002.



(a) The well-connected mesh topology: N=15, M= 27, L= 42, GD = 4, DC = 3.6

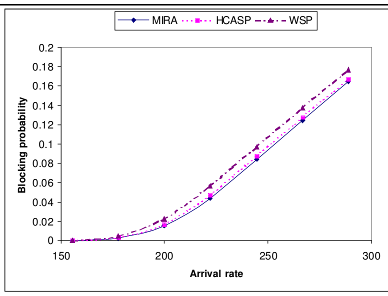


(b) The low-connected mesh topology: N=19, M= 25, L= 56, GD = 7, DC = 2.63

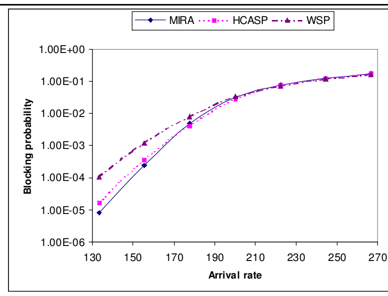


(c) The ring-like topology: N=20, M= 23, L= 240, GD = 8, DC = 2.3

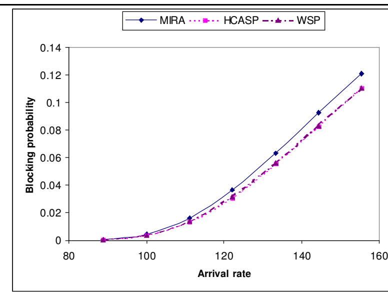
Figure 1. The three network topologies used in the simulation.



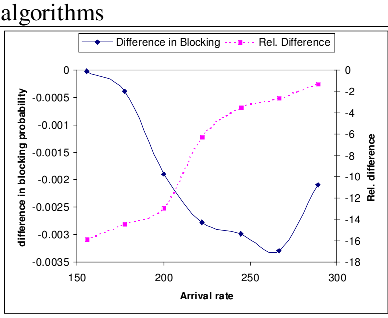
a) Blocking probability for the three algorithms



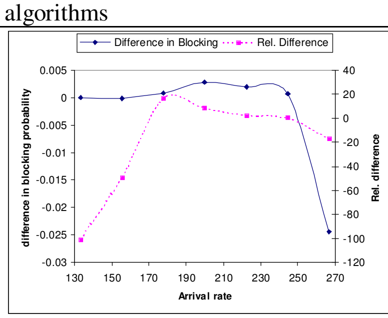
a) Blocking probability for the three algorithms



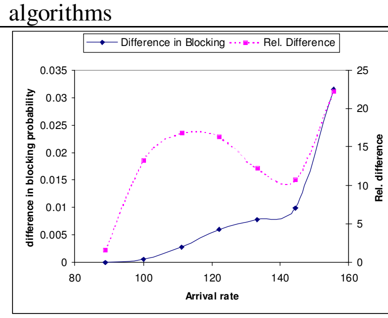
a) Blocking probability for the three algorithms



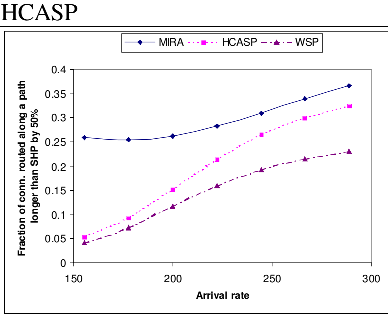
b) Difference and relative difference in blocking probability between MIRA and HCASP



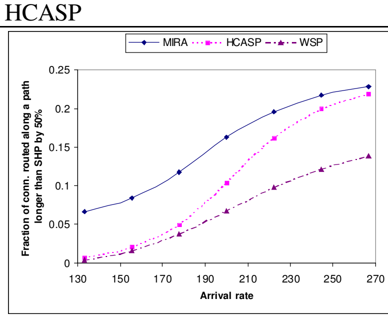
b) Difference and relative difference in blocking probability between MIRA and HCASP



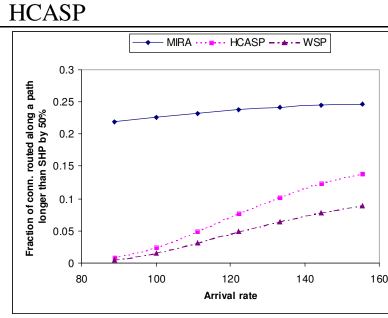
b) Difference and relative difference in blocking probability between MIRA and HCASP



c) Fraction of long-path connections for the three algorithms



c) Fraction of long-path connections for the three algorithms



c) Fraction of long-path connections for the three algorithms

Figure 2. Results for the well-connected mesh network

Figure 3. Results for the low-connected mesh network

Figure 4. Results for the ring-like network