INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Traffic Balance after Link Failures Using Few Weight Changes

Jing Wu[1], **Chengcheng Guo, Puliu Yan, Jianguo Zhou**

Lab for Internet and Information Technology, School of Electronic Information
Wuhan University,129 Luoyun Road, Wuhan, Hubei 430079, P.R. China

Abstract: Some failures in the network may lead to hot spot. The weight changes can be used to keep the traffic balance again after failures. But frequent weight changes may lead to disturbance in the network, so we should change link weights as few as possible. Local search algorithm can achieve that, but the computation is too complex. We present an OSPF/IS-IS weights tuning scheme named flow enumeration to reduce the computation complexity. The scheme is based on flow analysis and the main idea is to increase the weight of the link with maximum utilization and decrease the number of flows on the link. The simulations indicate that the computation time of our scheme is much lower than local search algorithm and the results of maximal link utilization are close.

Keywords: Traffic balance; Link weight; Link utilization; Traffic matrix; Local search; Flow enumeration

1. Introduction

The current approach to backbone design based on traffic engineering appears sufficient in most cases to keep the network load balanced. Sundar and Supratik [1] study data collected from 138 links on Sprint's IP backbone and found that the occurrence of sudden "hotspots", where the link utilization is as high as 90%, is mostly caused to link failures. However, at any given time, there is usually only one such "hotspot" in the network. Hence solutions are required to alleviate link overload after link failures.

In connectionless-oriented network, traffic is routed along shortest path to the destination, so the weights can be tuned to optimize the network resources again after some failures. But weight changes should be taken as few as possible, because even a single weight change is disruptive for a network. The weight change has to be flooded in the network. Every router re-computes the routing tables with several seconds after it receives the LSA update. Obviously, the more weight changes are flooded simultaneously, the more chaos are introduced in the network, and more packets are set back and forth between routers without the same view of network during the convergence.

Traffic balance of connectionless-oriented IP network based on link weight configuration is presented, when traffic engineering is developed originally. But traditional network management depends more on operators' experiences for weight configuration, so there is not much deep research on the field. Nevertheless, as recommended by Cisco [2], link weights are often just set inversely proportional to the capacities of the links, without taking any knowledge of the demand into account. Bernard Fortz et al [3] firstly present the research to optimize the weight setting based on the projected demands from AT&T WorldNet backbone. It shows that optimizing the weight settings for a given set of demands is NP-hard, so it resorted to a local search heuristic. The weight setting obtained from the algorithm could support a 50%-110% increase in the demands compared to the suggestion of Cisco. Bernard Fortz et al also use the local search heuristic for weight optimization in a changing world [4]. Its goal is also to minimize the maximal link utilization

[1] Corresponding author.
Email address: silence_wu@163.com

or minimize the sum of weighted link utilization through improving an input weight setting w_0 with as few weight changes as possible. Firstly, about 1000 single weight changes to w_0 are considered, corresponding to about five weight changes for each arc in our largest networks. As in [3], the number of weight changes is limited considered by applying random sampling to the neighborhood structure, as exploring the full neighborhood is too time consuming. Instead of selecting only the best weight change, 100 best weight changes are kept in a family F of "best weight settings." The process is iterated with F instead of w_0. It considers 1000 single weight changes for each weight setting in F and a new F is selected containing the 100 best of the old weight settings in F and the about 100000 new weight settings considered. After i iterations, including the start from w_0, the family consists of weight settings with up to i weight changes from w_0. The size of F corresponds to the breadth of the search. The simulation results indicate that for the largest networks considered, 10 iterations took about 1 1/2 hours on a single processor of a 194 MHz CPU SGI Challenge XL. The computation is too complex. Bernard and A. Nucc [5,6] then consider the possible link failures before original weight configuration. The worse case of link failure or the average effects of all possible link failures are considered to make the link weight configuration robust. The pre-configuration mechanism avoids the LSA and routing table update, but it is not very optimal for it can't cover all the failure scenarios and the computation is consuming. Alan Gous [7] describes the framework for automated optimization of routing metrics. It is indicated that the link weights should be tuned in a certain order to decrease the instability time of the network. But the tuning order is not discussed further.

Our work is motivated by traffic balance problem using few link changes after long-lived link failures. Which link weight should be changed and how to configure it are two major problems we should solve. One of the methods is to increase the weight of the link with maximal utilization, so some flows passing that link can be switched to other links. The operators can increase the weight of that link little by little until the traffic in the network is roughly balanced, but frequent tune is too consuming and we try to compute the increase of link weight only one time. In this paper, we present an algorithm named flow enumeration to solve that problem. In the new method, the weight increase of the link with maximal utilization is computed step by step, so that the flows are removed from the link with maximal utilization to other links one by one until the maximal link utilization no longer decreases. The flow enumeration algorithm can compute the weight increase very fast. The simulations reflect that the algorithm is much more efficiency than local search algorithm. It can also be used for tuning the weight configuration if traffic model changes greatly.

2. Problem Formulation

Above all, we define the following notations:

w_l: Weight of l^{th} link

Δw_l: Weight increase of l^{th} link

C_{ij}: Capacity of link $i \rightarrow j$

d_k: Traffic demand of k^{th} flow

s_k: Source of k^{th} flow

t_k: Destination of k^{th} flow

X_{ij}^{k} : Scaling of k^{th} flow passing through the link $i \rightarrow j$

a_{lk} : Scaling of k^{th} flow passing through the l^{th} link

E: Set of all the links

V: Set of all the nodes

K: Set of all the flows

The packets with the same input and output in the area belong to the same flow, i.e. flow is defined according to the source and destination of the packets.

If some links fail, the traffic will be redistributed. We are interested in increasing the weight of l^{th} link with maximal utilization after failures. If the weight of that link increases, the traffic on the link will be moved to other links, and the maximal utilization will be decreased. So we want to increase the link weight by Δw_l, so that the maximal link utilization will be minimized. The problem can be depicted in (1).

$$\begin{cases} \min \alpha \\ \sum_{k \in K} X_{ij}^{k}(\Delta w_l) d_k \le \alpha C_{ij} \qquad \forall (i,j) \in E \\ \Delta w_l > 0 \end{cases} \tag{1}$$

l is the sequence of the link with maximal utilization after failures. α is the maximal link utilization and the goal is to minimize α. X_{ij}^{k} could not be represented in analytic expression directly, so classical optimization method can not be used to solve the problem. The direct solution is to search for Δw_l from 1 with increase step of 1. But it is too complex, because every time Δw_l is changed, $|V|$ shortest path tree should be computed. The direct solution is too consuming.

3. Problem Solution

In fact, the main idea of flow enumeration algorithm is very simple. If some links fail, some links, for example l^{th} link may become hot spot. If the weight of l^{th} link is increased with Δw_l, some flows on l^{th} link are switched to some other links. From another viewpoint, if we want to transfer the k^{th} flows on l^{th} link to other links, the weight of the link needs some increase Δw_l^k, which is named least weight increase for k^{th} flow, so that the flows won't pass l^{th} link again. For every flow passing the l^{th} link after failures, the least weight increase is computed, and we can obtain the weight increase set $F = \{\Delta w_l^1, \cdots \Delta w_l^{|K|}\}$. So the solution is restricted in set *F*. If the weight increase in *F* is searched and sorted. The optimal solution can be obtained. We will try to depict the flow enumeration algorithm in detail.

Step1. New routing matrix computation.

If some link fail, some flows in the network must be affected and the routing paths are changed, i.e. the routing matrix $A=(a_{lk})$ is changed. The row number represents the link sequence and the column number represents the flow sequence. We must recomputed new routing matrix A' to obtain the possible hot spot after failure. If shortest path tree is computed with every $v \in V$ as root, the complexity is $O(|V|^2 \log |V|)$. In order to decrease the computation time, incremental SPT is used for new routing matrix computation based on original routing matrix [8]. The idea of incremental SPT computation is to resort the nodes below the failure links according to Dijkstra algorithm. The complexity depends on the number of affected nodes.

Step2. New link overload computation.

From the original routing matrix *A,* the set of flows Φ originally passing failure link is easy to be obtained. For every flow in *A*, $\tau \in \Phi$, the traffic of τ is subtracted from the load of the links contained in the original shortest path of τ and added into the load of the links contained in the new shortest path of τ. Hence we can get the new link loads and the link with maximal utilization u_l. If $u_l > u_{th}$, it reflects that the weight should be changed. u_{th} is the threshold and should be set according to the network state and topology. *l* is the sequence of the link with maximal utilization.

Step3. Flow enumeration.

For the l^{th} row in *A'*, the non-zero element corresponds to the flows passing the l^{th} link. The set of the flows passing the l^{th} link can be represented as $F=\{f_1,f_2,\ldots f_{|F|}\}$, the length of the shortest path corresponding to every flow is $w(f_m) \quad \forall f_m \in F$.

Step4. Shortest path computation without l^{th} link.

We exclude the l^{th} link from the topology, and compute the shortest path corresponding to every flow in the set *F* again. For every flow, the length of shortest path without l^{th} link can be represented as $w'(f_m) \quad \forall f_m \in F$. The incremental SPT algorithm can also be used here.

Step5. Least weight changes computation.

$$\Delta w(f_m) = w'(f_m) - w(f_m) + 1 \quad \forall f_m \in F \qquad (2)$$

The weight of l^{th} link must be increased with $\Delta w(f_m)$ at least to ensure that f_m won't pass l^{th} link. The elements in the set of weight changes are resorted, hence a sequence of weight changes $\Delta' = \{\Delta' w(f_i')\} \quad \Delta' w(f_i') \le \Delta' w(f_{i+1}')$ and corresponding flow sequence *F'* are obtained. Every time the weight of l^{th} link is increased with $\Delta' w(f_i')$, some flow will be removed from l^{th} link.

Step6. Computation of maximal link utilization for all possible weight changes.

For *i*=1 to |*F*|, the new weight $w_l' = w_l + \Delta' w(f_i')$ and the new routing for the flow f_i' are computed. Hence maximal link utilization $u_{\max}^i$ for all possible weight changes can be obtained.

Step7. Computation of minimization of maximal link utilization.

In the flow enumeration algorithm, the weight of the link with maximal link utilization after failures is increased to alleviate hot spot or congestion. The flows passing the link are enumerated and removed from the link and rerouted without the link one by one, and the corresponding maximal link utilization is computed step by step. The minimization of maximal utilization is selected. The algorithm reduces the search ranges of weight increase, so it is much efficiency.

4. Simulations

4.1 Methodology

4.1.1 Simulation topology

The Transit-stub topology generation tool [9] is used to generate random or transit-stub topology with 50, 100, 150 and 200 nodes respectively. The node degree is distributed in the interval [3.92 5.78] as depicted in Figure 1. The original link weights are distributed in the interval [1000 3000] uniformly.

4.1.2 Simulation traffic matrix

The gravity model [10] is always used for traffic matrix simulation. The gravity model can be represented as $T_{ij} \propto T_{i*} \bullet T_{*j}$, where T_{ij} is the traffic

from i to j, T_{i*} and T_{*j} denote the total traffic entering the network at i and exiting at j. For each node x, we pick two random numbers I_x, $O_x \in [0,1]$, which represents the traffic scaling entering and exiting the network at x. Another parameter T denotes the throughput of the network, and the traffic entering at i and exiting at j can be represented as $T_{ij}=T \bullet I_x \bullet O_x$. Similar methodology is also used in reference [1][4]. It is supposed that all the link bandwidth in the network is 10G. The table below represents the value of T(Mb) and corresponding maximal link utilization.

The value of T in transit-stub topology is much smaller than random topology. Because the transit-stub is a kind of hierachical topology, the links between different layers easily become the bottlenecks. The threshold of link weight tuning is u_{th}=0.8.

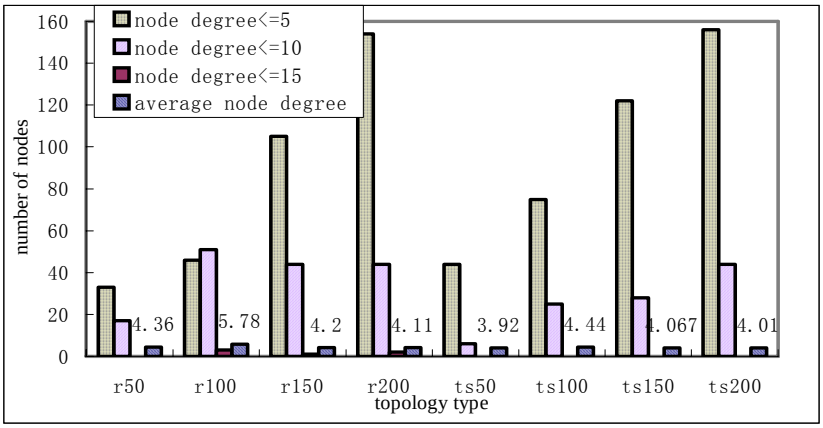


Figure 1. Node degree distribution of simulation topology

Table 1. Value of T in the simulation

Topology type	r50	r100	r150	r200	ts50	ts100	ts150	ts200
T(Mb)	300	80	120	30	40	10	30	5
Maximal utilization(%)	86.6	71.0	69.7	76.8	76.9	57.7	75.3	76.8

4.2 Simulation results and analysis

4.2.1 Single link weight change

Figure 2 denotes the comparison of maximal utilization after weight changes according to flow enumeration algorithm and local search algorithm respectively. The neighborhood scaling b is set 0.5, 1, 1.5, 2 respectively in local search algorithm. The searching range is the original weight scaled by b. M=1, i.e. the weight of at most one link is changed. Every simulation is repeated five times and average is considered as the result for a single link failure, two link failures and single node failure respectively. The failure elements are selected randomly, but the same elements are selected in every group simulation. Several conclusions can be obtained from Figure 2.

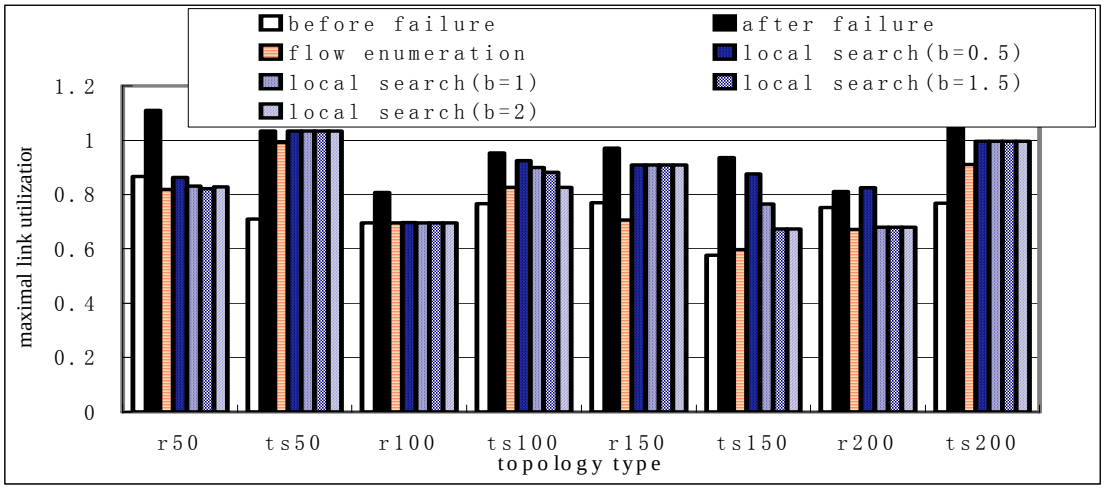
(1) The bigger the neighborhood scaling b of local search algorithm is, the more optimal the result is, because the searching range is extended.

(2) For (b)ts50 and (c)ts100，the hot spot can't be avoided no matter through flow enumeration algorithm or local search algorithm. Because the hop spot is on the 'key' link, i.e. the flows on the link won't be switched no matter how the weight is changed. The cases occur in transit-stub topology more often than in random topology, because the transit-stub is hierarchy topology and the links between different layers are easy to become the bottleneck.

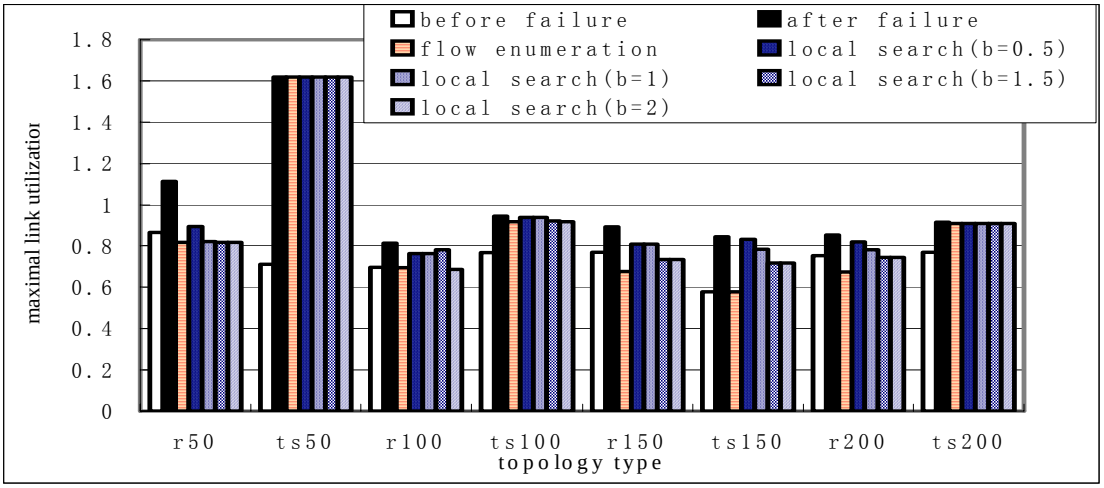
(3) For (a)r150, (a)ts150, (a)ts200, (b)r150, (b)ts150, (b)r200，the results of flow enumeration algorithm are much better than local search algorithm. Because the weight must be changed so much that the increase exceeds 2wi. So the increase exceeds the searching range of local search algorithm.

(4) For(c)r50，the local search algorithm is better than flow enumeration algorithm. The essence of the flow enumeration algorithm is to remove the flows from the link with maximal utilization in some order. So before a flow is switched, some other flows may have been switched before. It is not a good idea if

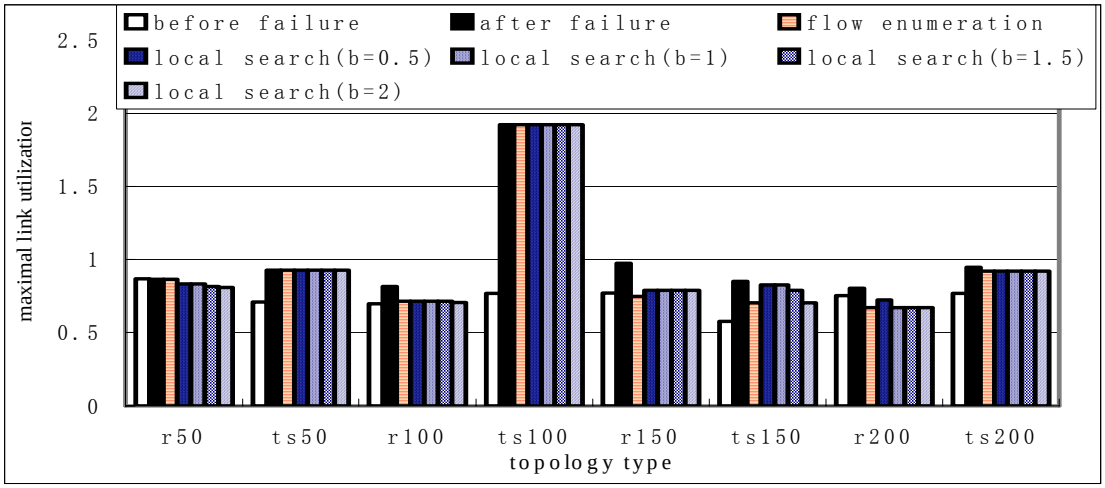
the flows are switched to other links with very large utilization. But local search algorithm changes the weights randomly, so the flows maybe switched to some links with low utilization.



(a) Maximal link utilization through weight changes after single link failure



(b) Maximal link utilization through weight changes after two link failures



(c) Maximal link utilization through weight changes after sin link failures

Figure 2. Maximal link utilization through single weight change after failures

Table 2. Computation time of flow enumeration and local search algorithms for single weight change (seconds)

	r50	ts50	r100	ts100	r150	ts150	r200	ts200
FE SLF	0.1	0.2	0.5	1.5	1.8	12	30	66
LS b=0.5 SLF	23	25	123	164	384	354	1164	1090
LS b=1 SLF	24	25	124	164	386	356	1163	1091
LS b=1.5 SLF	24	25	124	164	385	356	1163	1090
LS b=2 SLF	24	25	124	164	386	356	1163	1090
FE MLF	0.1	0.2	0.5	1.6	1.8	12	33	86
LS b=0.5 MLF	22.9	25	134	165	384	358	1163	1085
LS b=1 MLF	23.0	24	134	171	386	356	1164.	1083
LS b=1.5 MLF	23	24	133	169	387	357	1164	1084
LS b=2 MLF	23	24	129	169	385	356	1165	1084
FE SNF	0.1	0.2	0.4	1.6	1.7	7.7	38	64
LS b=0.5 SNF	23	24	120	165	386	312	1161	1169
LS b=1 SNF	23	24	120	164	383	315	1162	1169
LS b=1.5 SNF	23	24	120	167	386	314	1162	1170
LS b=2 SNF	23	24	120	167	383	314	1160	1170

(5) In other cases, the effect of flow enumeration algorithm is similar to local search algorithm, but the flow enumeration algorithm is much more efficiency. The table below reflects the computation time of two algorithms. The computation time is the average of five same simulations and the process is Pentium(R) 4 CPU 2.8GHz. FE, LS, SLF, MLF and SNF respectively denote flow enumeration algorithm, local search algorithm, single link failure, multiple link failures and single node failure.

Table 2 indicates that flow enumeration algorithm needs much less time than local search algorithm. Because |V| shortest path trees construction are needed in local search algorithm every time the link changes are searched. On the contrary, flow enumeration algorithm computes the weight increase from the routing changes of flows.

Figure 2 indicates that the single link weight change may still make the maximal utilization smaller than the threshold. So we can consider changing weights of multiple links. For local search algorithm, M iterations are needed for getting optimization solution. For flow enumeration algorithm, the weights of M links may be tuned. Figure 3 denotes the comparison of maximal utilization after multiple weight changes according to flow enumeration algorithm and local search algorithm respectively. The neighborhood scaling b is set 2 in local search algorithm. M=5, i.e. at most five link weights are changed. Every simulation is repeated five times and average is considered as the result respectively. The failure elements are selected randomly.

4.2.2 Multiple link weight changes

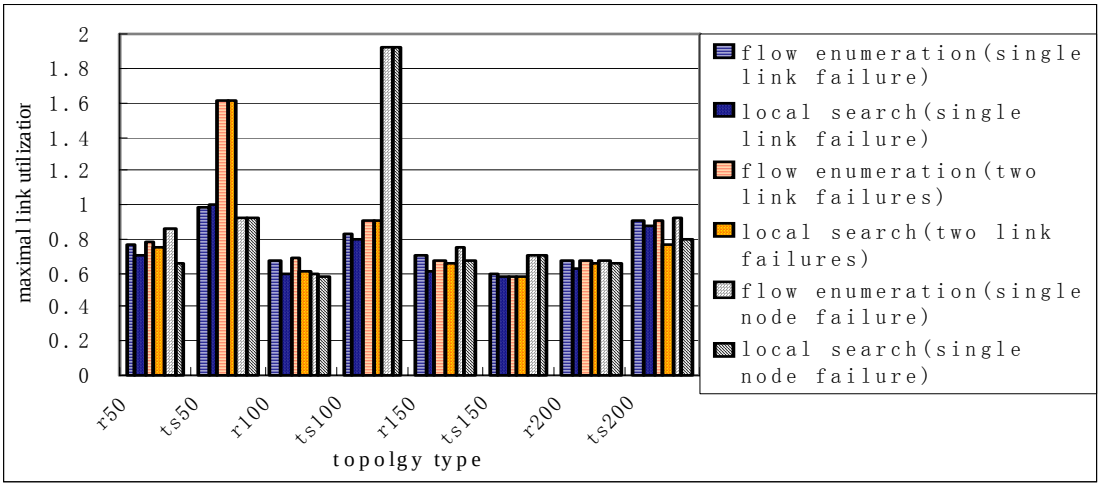


Figure 3. Maximal link utilization through multiple weight changes after failures

Table3 Computation time of flow enumeration and local search algorithms for multiple weight changes (seconds)

		r50	ts50	r100	ts100	r150	ts150	r200	ts200
FE	SLF	0.2	0.3	1.3	1.6	1.8	11.5	29.6	65.7
LS	SLF	9164	9922	51245	78840	167033	187690	218723	259214
FE	MLF	0.3	0.3	0.7	4.6	1.9	12.0	33.3	86.3
LS	MLF	9176	9924	50812	78900	168812	193443	227224	269763
FE	SNF	0.2	0.3	1.6	1.6	1.7	7.7	37.9	124.3
LS	SNF	9595	9655	50985	78913	162945	180745	230803	269047

Figure 3 indicates that the effect of local search algorithm is a little better than flow enumeration algorithm in some cases. It attributes to two reasons.

(1) The flow enumeration algorithm stops if the maximal utilization is smaller than the threshold. But local search algorithm won't stop until the M iterations are finished.

(2) If the link with maximal utilization of this iterative step is same as last step, the flow enumeration algorithm can't increase the weight of the link to decrease the maximal utilization any more. But local search algorithm can decrease the maximum further through changing the weights of other links.

But the computation time of local search algorithm increases 100 times while the computation time of flow enumeration algorithm is within 100 seconds as depicted in the Table 3.

5 Conclusion

If the network capacity is not redundant enough, network failures always lead to some links overloaded. The explicit routing can be used in MPLS network to avoid that problem, but pure IP network only depends on network management to make the traffic balance. Changing link weight is one of those effective mechanisms. But frequent weight changes may cause instability of network, so link weights should be changed as few as possible. Though local search algorithm can change the link weight as few as possible, it is too complex. The flow enumeration algorithm is presented in the paper for weight changes. Its main idea is to increase the weight of the l^{th} link with maximal utilization after failures, so that for the flows on the l^{th} link originally, the shortest paths won't include the l^{th} link any more after weight changes. The flow enumeration algorithm shrinks the solution searching range of local search algorithm, so it is more efficiency. The simulations validate that the time of single weight change computation is always within 100 seconds, but the complex of local search algorithm increases fast with the number of links whose weights are changed increases.

Flow enumeration algorithm is very efficiency, but if the network topology is not connected well, only weight changes can't solve the problem. So for connectionless-oriented, both link weight configuration and topology update are important for network management operators to keep the traffic balance. The work related to the reliable network topology will be studied further.

Acknowledgments

This work was supported by the National Nature Science Foundation of China under Grant No. 60502028.

References

[1] Sundar Iyer, Supratik Bhattacharyya, Nina Taft, Christophe Diot: "An approach to alleviate link overload as observed on an IP backbone", In: *Proc. of INFOCOM*2003, Vol. 1, 30 March-3 April 2003, pp: 406 - 416

[2] Cisco. (1997) Configuring OSPF. [Online]. http://www.cisco.com/uni-verc/cc/td/doc/product/software/ios113ed/113ed_cr/np1_c/1cospf.htm.

[3] Bernard Fortz, Mikkel Thorup: "Internet Traffic Engineering by Optimizing OSPF Weights", In: *Proc. of INFOCOM* 2000, 26-30 March 2000, vol.2, pp. 519 – 528

[4] Bernard Fortz, Mikkel Thorup: "Optimizing OSPF/IS-IS Weights in a Changing World", *IEEE Journal on selected areas in communications*, Vol. 20, No. 4, May 2002

[5] Bernard Fortz, Mikkel Thorup: "Robust optimization of OSPF IS-IS weights", In: *Proc. of INOC* 2003, pp: 225-230, October 2003.

[6] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, C. Diot: "IGP Link Weight Assignment for Transient Link Failures", *SPRINT ATL Technical Report TR02-ATL-071000.*http://cambridgeweb.cambridge.intel-research.net/people/pub/cdiot/TR02-ATL-071000.pdf

[7] Alan Gous, Arash Afrakhteh: "Traffic Engineering through automated optimization of routing metrics",

In: *Proc. of Trans-European Research and Education Networking Association Networking Conference*, June 2004

[8] J. M. McQuillan, I. Richer, and E. C. Rosen: "An overview of the new routing algorithm for the Arpanet", In: *Proc. of 6th symposium on Data communications*, pp: 63-68, ACM Press, 1979

[9] Calvert K, Doar M, Zegura E.: "Modeling Internet topoloty", *IEEE Communication Magazine*, 1997, Vol.35, No.6, pp:160-163

[10] Yin Zhang, Matthew Roughan, Nick Duffield, Albert Greenberg: "Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads", In: *Proc .of SIGMETRICS'03*, June 10-14, 2003, San Diego, California, USA