Scheme for Alternative Packet Overflow Routing (SAPOR)

Alexander A. Kist and Richard J. Harris
RMIT University Melbourne
BOX 2476V, Victoria 3001, Australia
Telephone: (+) 61 (3) 9925-5218, Fax: (+) 61 (3) 9925-3748
Email: {kist,richard}@catt.rmit.edu.au

***Abstract*— Shortest path routing schemes like Open Shortest Path First (OSPF) have shortcomings when networks are highly loaded. Traffic engineering of IP networks is required to avoid this problem. Current efforts suggest the optimisation of OSPF weights to balance the network load more evenly. Also more advanced technologies like Multiprotocol Label Switching (MPLS) are proposed. One major problem of dynamic routing efforts that are using OSPF is the fact that many traffic flows are influenced by single weight changes. The Scheme for Alternative Packet Overflow Routing (SAPOR) which is introduced in this paper, realises a methodology that can remember the routing of packets for the duration of a micro flow. This allows the rerouting of overflow traffic. In this case, well known concepts and methodologies from conventional circuit switched teletraffic engineering can be adapted for IP networks.**

I. Introduction

IP networks are a major technology for the transport of data. More than half of the transported traffic is already IP based. The core backbone network delivers the IP packets to their destination. Core routers and the interconnecting bearers form these networks. Most topologies allow many alternative paths in such backbone networks. Shortest path routing, in particular OSPF [1], is widely used to select the appropriate paths for origin destination pairs. Over a wide range of operating conditions, OSPF provides optimal solutions, in particular, in Internet Service Provider (ISP) grade Internet environments. The principal problem of shortest path routing, which many efforts try to solve, is that by definition all traffic is routed on the shortest path. If the load of the network increases, the shortest path links will be highly loaded, whereas other alternative network resources are potentially unused.

On the other hand, many current initiatives work on the migration of telephony and other carrier grade services to all IP next generation transport networks (e.g. 3GPP [2]). In such a context, the Internet philosophy of “best effort” service and “over provisioning” are not satisfactory any more. Even current QoS ensuring methodologies like DiffServ and IntServ do not address all problems. In particular, traffic-engineering problems are not solved by these QoS technologies. But the major selling points of carrier grade services are guaranteed service levels. Traffic engineering efforts are necessary to provide carrier grade services using IP network bearers. Mutiprotocol Label Switching (MPLS) [3] is one possible approach, to engineer the current networks and allow arbitrary routes, although, MPLS requires the migration of whole network regions to MPLS. Recent research also targets QoS routing and MPLS (e.g. Ying-Dar Lin [4]). Other research efforts directly target the optimisation of OSPF protocol weights to adapt the cost metric for given traffic demand matrices. This includes the Equal Cost Multi Path (ECMP) effort which is now part of OSPF Version 2. ECMP splits the traffic between paths with equal cost. Fortz and Thorup [5] use Tabu search techniques to optimise OSPF weights. Murphy et al. [6] use a linear programming approach and the fast solver CPLEX to generate optimised OSPF weights. Other approaches by Harmatos [7] use a heuristic and by Ye et al. [8] use online simulation to get to similar results.

One major disadvantage of all OSPF rerouting techniques is that all have an impact on existing traffic flows. For example, if weights of links are changed, all existing flows will be rerouted. It can introduce major traffic shifts and instabilities in the network. This is one of the major arguments in [5] “why weight changes are bad”. A second argument stresses the importance that the network operators are in charge. This paper introduces a Scheme for Alternative Packet Overflow Routing (SAPOR). In this case, only new flows are redirected by this scheme, the routing of existing flows remains the same. This is in particular interesting, since a few long living flows carry a large fraction of the traffic and a growing number of real-time services would profit from persistent packet routing. In principle, this scheme can be combined with any of the above-mentioned efforts. It extends existing methods with the possibility of dynamic routing without the negative side effects of weight changes. SAPOR is rather an enabling technology than a new optimisation effort.

The principal concept of overflow routing is not new and widely known by the teletraffic engineering community. Conventional circuit switched network operators have used these methods for years. Dynamic non-hierarchical routing (DNHR) [9] which uses different path sets for different times of the day, for voice carriers, was initially developed by AT&T. Other examples of these efforts include works on Dynamically Controlled Routing (DCR) [10], Dynamic Alternative Routing (DAR) [11] and State- and Time-Dependent Routing STR [12]. Gerald Ash’s book [9] presents a comprehensive discussion of this topic.

The discussion in this paper will use the notion of traffic

flows. A flow is the aggregate of a large number of packets that are directed from the same origin node to the same destination node in the observed network. These flows can be measured in bytes per second. A micro flow is the aggregate of packets between the same IP packet endpoints, i.e. the same source and destination IP addresses. To further diversify flows, port and protocol numbers can additionally be used to define micro flows. The remainder of this paper is organised as follows: The next section introduces the concept of SAPOR and Section III describes the different functional components in more detail. Section V discusses issues like the performance and the requirements of SAPOR. The paper concludes with a discussion of further work.

II. Concept

A router on the network layer has to route all incoming packets on the appropriate links of the paths to the destination node. The SAPOR scheme is located in routers. Figure 1 depicts the concept of SAPOR, i.e. the way outgoing links are chosen for incoming packets. Compared to a conventional system, SAPOR is located in between the router function that switches the packets on the links and the routing table that is generated by the shortest path algorithm. The main functional groups of SAPOR are the *Hash Function*, the *Token System* and the *Routing Tables*. The hash function consists of two parts: the actual function (2) and the hash space (3). The token system consists of the token buffers (6), the token scheduler (5) and the token list (4) which is equivalent to the hash space. The routing tables (7) are a selection of different tables. The incoming packets are accumulated (1) and routed on the emanating arcs (8). The dotted lines indicate the token flows, the dash dotted lines indicate requests to the routing tables and the solid line shows the virtual packet flow in this scheme. Every outgoing link has an associated token buffer. To distinguish the different links and buffers in the discussion, colours are used to indicate their affiliation.

This paragraph describes the operation of SAPOR. The incoming buffer receives new packets (1). As a first step, the hash space is calculated for a packet of the origin and destination address by a hash function (2). In the second step, a token buffer is selected (6). This selection is firstly based on the primary routing table (7). If the table indicates an outgoing link (e.g. red) and an available (red) token (6), the token is assigned to the hash space (3) and the number of tokens in the token buffer is reduced. In the case where the space already has an assigned token, this step is skipped. The current packet and all subsequent packets are routed on the link with the same colour (red). If the token buffer is empty, the secondary routing table is considered and the same procedure is used for the second table. If new packets arrive that are mapped on the same hash space that already holds a token, the packets are sent on the outgoing link with the appropriate colour. Tokens have a finite time to live. If no new packets arrive within the specified time, these tokens are selected and returned to the appropriate token buffer by the token scheduler (5). If a packet arrives before the time has expired the timer is reset.

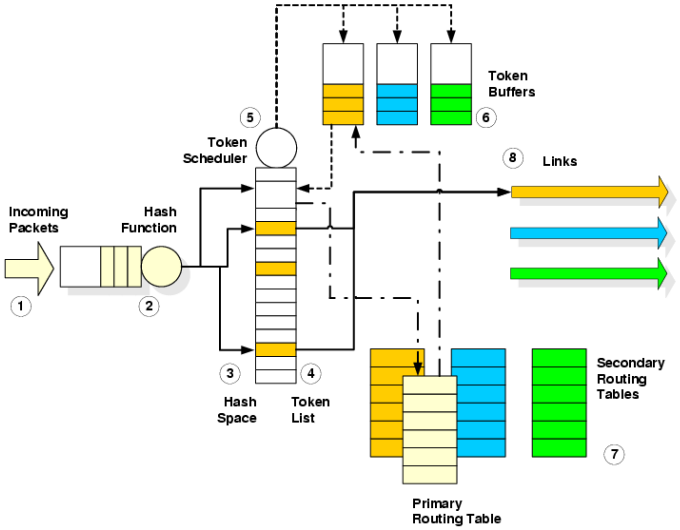

Fig. 1. SAPOR Scheme

III. Building Blocks

SAPOR uses three major building blocks: The hash function, the token system and the routing tables. This section discusses the functional blocks in more detail.

A. Hash Function

The purpose of the hash function is to separate single micro flows on the basis of flow specific parameters like the source IP address, the destination IP address and other parameters. A wide range of possible hash functions exist. The work by Cao et al. [13] discusses hashing based schemes for Internet load balancing and investigates the performance of different methods. Most of their results can be applied in this context as well. The important attributes of a hashing scheme for SAPOR can be summarised as follows: The results have to be evenly distributed over the hash space and the hash space should be large enough that no frequent overlapping occurs. For example, a 16 bit hash function yields 65536 hash spaces. If link specific hash functions are used (See the alternative Section V-F) the overlapping is less critical and smaller hash spaces can be used. For the purpose of SAPOR, the XOR folding of the source and destination address is sufficient. Equation (1) shows this calculation.

$$H(.) = \Big(S_1 \oplus S_2 \oplus S_3 \oplus S_4 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4\Big) \bmod N \quad (1)$$

The ith bytes of the source and the destination IP address are denoted by S_i and D_i respectively. N limits the size of the hash space. [13] indicates a sufficient spreading and this scheme is simple to implement. Alternative hash functions can also be used.

If the mapping of the hash space changes, disruptions can occur since the mapping has to be reorganised. The concept of robust hashing was introduced in [14] to combat this setback.

However, this problem does not occur in this context, since the token mapping is persistent, even when new links are added or they disappear. For further discussion of failure modes see Section V-B.

B. Token System

The token system in combination with the hash function is a major part of SAPOR. The number of available tokens defines the size of the aggregated flows that are allowed on outgoing links. This behaviour is similar to the token buffers that are used for the leaky bucket scheme (e.g. [15]). The number of tokens depends on the (statistical) properties of the flow aggregates and the interval in which the token list is updated. A token buffer is associated with a link and it has a limited number of tokens. Every time a flow is transported on this link a token is assigned to the flow. If no tokens are available no more flows can be routed on this link. Tokens have a (token) time to live (TTTL). If a flow transmits no more packets and the TTTL has expired the tokens are returned to the appropriate buffer.

1) Token Buffers: The token buffers can be simply implemented by a counter that indicates the number of tokens in a buffer. If a token is removed, the number decreases; if a token is returned, the number increases. Under normal operating conditions, this number is positive, but if the bucket size changes during operation, the bucket count can be negative. No tokens are available if the token count is less than one.

2) Number of Tokens: The major parameter that specifies a token buffer is the number of available tokens ν. It defines the maximum number of flows on the associated link. The calculation that is shown in this section uses the assumption that many micro flows exist and the measure of average flow size is a valid approximation. This requires, in particular, that no micro flow dominates over others, i.e. no single flow peaks the flow aggregates. The average micro flow size is denoted by $\overline{f}$ and is measured in bytes per seconds. The average duration of a flow is denoted by $\bar{t}$ and measured in seconds. The flow arrival rate is denoted by ϑ and measured per second. The number of active flows A can be calculated by Equation (2).

$$A = \vartheta \cdot \bar{t} \tag{2}$$

Equation (3) shows the calculation of the number of required tokens ν_l for link l. The first tokens are determined by the number of flows on the link, the second sum calculates the number of tokens that are required due to the TTTL of the tokens and the delayed reset of τ.

$$\nu_l = A_l + \delta \cdot \tau \cdot \frac{C_l}{\sum_{i=1}^{l_{max}} Ci} \tag{3}$$

where l_{max} is the number of outgoing links and $\delta\nu$ is the average flow change rate measured in new flows per sec. The second product is necessary to scale the change rate to the single emanating links. The possible number of flows A_{max} depends on the capacity C_l, the average flow size $\overline{f}$ and the utilisation u_l. A_{max} is depicted in Equation (4).

$$A_{max} = \frac{C_l \cdot u_l}{\overline{f}} \tag{4}$$

For example, an average flow size of $102.4 bytes/sec$, an average duration of a flow of 30 seconds, an average flow arrival rate of $207.8 Flows/sec$, a token update every 10 seconds and a utilisation of 60% yields 8144 tokens for one single outgoing link (Equation (5)).

$$\nu_l = 204.8 \cdot 30 + 200\ 1/sec \cdot 10\ sec = 8144 \tag{5}$$

The required link capacity can be calculated using Equation (6).

$$C_l = \frac{6144 \cdot 204.8\ bytes/sec}{0.6} = 1\ Mbyte/sec \tag{6}$$

3) Token Scheduler: The token scheduler is the function that determines "expired tokens", clears the token list and returns them to the token buffers. The token scheduler is the most expensive function of the SAPOR scheme, but it is not frequently used. It requires two major sub functions: The traversing of the token list and the implementation of the TTTL. The discussion in this section gives possible implementations, although there are many different ways for how the same functionality could be achieved.

The token list can be implemented by using a combined array-linked-list data structure. This avoids the requirement that all spaces have to be traversed for the price of additional memory. Every time a storage space is assigned, the "next pointer" of the last added space, is pointed to the current space. The current "next pointer" is set to a dummy end node. A global pointer remembers the end of the list space. During the iteration-step, the array-list is traversed in the sequence of the linked list. The iteration-step also remembers the last space it visited. In this way it is possible to delete a space by connecting the "next pointer" of the previous space to the space that is indicated by the "next pointer" of the current space. Other implementations are also possible.

The TTTL can be implemented with a simple counter variable. Every token list item has one such variable associated with it. Zero indicates that the space is empty, i.e. no flows are using this space. Empty spaces are skipped by the iteration operation. If the number is positive it is increased every time it is traversed by the iteration function. If a packet arrives during the update interval the number is reset by this event. The iteration function is executed every T_c seconds. If n reaches the maximum count n_{max} the counter is set to zero and the token is returned to the appropriate token buffer. n_{max} is a positive integer that is larger than one. In this case, the minimum time before a token is deleted is $(n_{max} - 1) \cdot T_c$ and the maximum time before a token is returned is $n_{max} \cdot T_c$, where T_c is the time interval between two iterations. This difference is due to the resolution which is defined by the number of count steps $0 \ldots n_{max}$ for the timer. Larger n_{max} yield smaller differences between the minimum

and the maximum values. The mean time τ before a buffer is delated is therefore defined by Equation (7).

$$\tau = \frac{2 \cdot n - 1}{2} \cdot T_c \tag{7}$$

It is obvious that for large n the fraction approximates n.

4) Token Update Interval: Since the token scheduler is the most expensive function, the token update interval T_c should be as long as possible. On the other hand, long intervals require more tokens, a larger token list and the traversing of more active tokens in the token list. The number should be chosen on the basis of the evaluation of the given constraints. For the discussions in this paper an arbitrary number of 10 seconds has been chosen.

IV. Routing Tables

The actual routing function of SAPOR uses a number of possible alternative routing tables. They are denoted by primary, secondary, tertiary, ... n-ary table etc. These tables can be generated by any means that are appropriate and useful for the network configuration. SAPOR uses one primary routing table per node and subsequent routing tables can be link specific. If all routing tables are the same and built by a standard routing algorithm (e.g. OSPF), the SAPOR scheme behaves in exactly the same way as the original OSPF implementation. In general, the mechanisms of the legacy systems should be used for the primary routing table, for example, the original shortest path routing tables. The routing decisions for the n-ary routing tables and therefore paths can be based on any factors, e.g. administrative decisions by human network operators. In the case of SAPOR, the administrator can stay in charge of any (overflow) routing decision. A simple automatic way of generating these secondary routing tables is to remove the outgoing link in the shortest path calculation and recalculate the shortest path tree without the first link for the relevant secondary routing table. The same principle can be applied for all subsequent emanating links.

Secondary routing tables could also define a complete redundant link system that is only used if the primary system is overloaded or fails. As mentioned before in this paper, existing teletraffic engineering methodologies can be applied and used. The engineering efforts are reflected in the selection of primary, secondary, etc. paths. Note that the path choices are arbitrary but it has to be ensured that the primary and all n-ary paths do not build loops. This is particularly important since this scheme is node-local.

V. Remarks

This section discusses some further issues that concern the SAPOR scheme, in particular, requirements and performance issues.

A. Speed of Changes

If the routing tables in the SAPOR scheme are changed, the traffic flows do not change instantly, they are smoothly shifted onto new links. The change speed is defined by the number of new arrivals per second and therefore departures per second. Equation (8) shows the duration of a change δt.

$$\delta t = \vartheta \cdot \delta\nu \tag{8}$$

Where ϑ is the flow arrival rate in flows per second and $\delta\nu$ is the number of existing flows that have to be rerouted. For example, if $\vartheta = 200 Flows/sec$ and $\delta\nu = 1000$ existing flows have to be rerouted this will take 5 seconds.

B. Network Topology Changes

In the case of network topology changes, e.g. connection or nodes disappear or become operational, the scheme has to adapt to these changes. When new connections come online, SAPOR will simply shift flows onto new capacities as they become available. This will occur according to the speed of change philosophy outlined in the previous section.

If nodes are no longer reachable, the speed of change might not be fast enough, since packets will be still routed on obsolete links. The failure mode for adjacent nodes is fairly simple. If an adjacent link is failing, the corresponding token buffer is emptied and all tokens in the token list that belong to this link are flushed. In this way all existing flows will be assigned to new links.

If "non adjacent" links or nodes disappear, there are several possible actions. Firstly, no specific action is taken and the flows rearrange with the speed of change. Flows that are routed to nodes that have lost their connectivity are disrupted. Secondly, the token list is completely flushed if major routing table changes occur. In this case flows that are routed by the primary routing table end up on the same links as before, the overflow routing however may be disrupted. Thirdly, only tokens are flushed that belong to links that are the root to changed network parts. These can be identified by comparison of the different routing tables.

Lastly, only tokens of links are flushed that have lost their connectivity. To identify changes in connectivity a simple method can be used. Routing tables in nodes are written as vectors, where every dimension identifies one emanating link. The elements consist of a collection of nodes n_{xyz} that are reachable via this link. Equation (9) depicts an example of a vector PT_1 which reflects a primary routing table.

$$RT_1 = \begin{pmatrix} \{n_a, n_b\} \\ \{n_c, n_d, n_e\} \\ \{n_f\} \end{pmatrix} \tag{9}$$

All other routing tables can be written in the same way as PT_2, PT_3 etc. The connectivity vector PT can be calculated by using the logical "or" function on the vector elements. Equation (10) shows this calculation.

$$RT = RT_1 \vee RT_2 \vee \cdots \vee RT_n \tag{10}$$

The elements of RT in one dimension identify nodes that can be reached via the link that represents this dimension. To find links that have lost their connectivity, the RT values

before and after the table changes have to be compared. This calculation is depicted in Equation (11).

$$L = RT_t \wedge \overline{PT_{t+\delta t}} \tag{11}$$

The logical "and" operator results in differences in connectivity that existed in the first vector but not in the second vector. All links in vector L that are not empty have changed connectivity and have to be flushed.

The fourth option is the least disruptive, but it also requires the most effort. Further research has to identify the right approach for specific networks and operating conditions. For instance, the topology in the core part of backbone networks will change far less frequently than the topology in access parts of IP networks and the problems will therefore require different strategies.

C. Requirements and Performance

1) Complexity: The complexity of SAPOR is twofold. All operations that are required during packet routing are of the order of one: $O(1)$ i.e. the hash function, the check if a token is assigned, the lookup of the routing table, the assignment of a new token and the changes in the token buffer, therefore, it is possible that all packets are routed in $O(1)$ time. If a token is already assigned to the buffer less $O(1)$ steps are required. The second operation is more complex, and therefore more expensive, but it is less frequently required. To clear and update the token list, all spaces have to be traversed. The complexity of this process is therefore $O(l)$ where l indicates the size of the token list. l is equivalent to the hash space. Using more memory and intelligent data structures can reduce the number of spaces that have to be traversed.

2) Memory: The memory requirements for the routing tables are equivalent to the requirements for tables of legacy systems. The token buffers are simple counters and require only several bytes per buffer. The token list is of size l and requires therefore l bytes. However, a more advanced token list requires $k \cdot l$ bytes.

D. Efficiency

As previously mentioned, SAPOR is an enabling technology and not a routing methodology in the principal sense, therefore, actual performance improvements or comparisons with existing technologies are not possible. The efficiency in terms of improved utilisation etc. depends solely on the methodologies that are used to generate the alternative routing tables, although SAPOR enables the use of these new methods. Future research will have to focus on adaptation and development of efficient routing methodologies for the SAPOR scheme.

E. Statistical Effects

All calculations in this paper are undertaken with the mean values of the appropriate parameters. In practice these simplifications will be sufficient, since many hundreds of flows are observed. If SAPOR is used in very specific environments, e.g. where many long-term flows exist, some of the parameters might require adaptation to these networks.

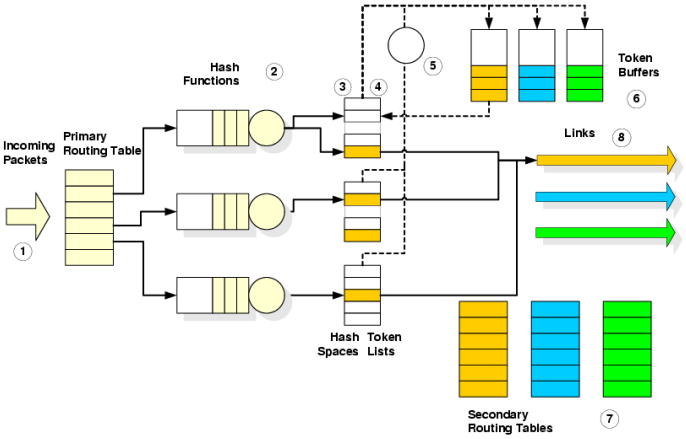

Fig. 2. Alternative SAPOR Scheme

F. Modified Scheme

Figure 2 depicts a simple modification of the original SAPOR scheme. Most of the function blocks are the same as previously described. The major difference is the location of the primary routing table. It is situated before the hash function. The modified scheme also uses separate hash spaces for all emanating links. The principal functionality of both schemes is the same. The latter requires more separate building blocks and can use shorter hash spaces, whereas the former uses fewer building blocks and requires one larger hash space. The modified scheme has the disadvantage that it requires a lookup of the primary routing table every time a packet has to be routed. The complexity is therefore slightly higher than that of the original scheme.

G. Implementation

One minor disadvantage of SAPOR compared to pure management efforts is that SAPOR has to be implemented in existing routers, although it uses methods that are already implemented in existing routers. Simple changes to software modules can implement SAPOR. It has to be noted that the migration is a simple process, since SAPOR is a local router function and can easily coexist with legacy systems. In fact, the nodes can be updated one by one. Improvements are already enabled with the first migrated node.

VI. Further Study

Recent years have seen major efforts to develop QoS routing schemes (e.g. Ying-Da Lin et al. [4]). The version of SAPOR that was discussed in this paper does not directly target different traffic classes, constraint based routing and QoS routing. Future schemes can support different QoS technologies. Decisions within the scheme can be further separated and the routing can be based on the type of service, for example.

Secondly, as indicated above, there are a wide range of methodologies and schemes available from circuit switched networks. Further research efforts will have to focus on the investigation of the applicability of these schemes for packet switched networks that use SAPOR.

Thirdly, research efforts will have to focus on the further development and the testing of SAPOR with existing and new routing methodologies. Then, more extensive performance analysis and the simulation of realistic traffic scenarios will be possible.

Lastly, the generic concept that was introduced in this paper could also be adapted to other scenarios as well. Examples include distributed caching systems and load sharing in between different processors.

VII. Conclusion

Under normal operating conditions in conventional Internet environments, shortest path routing provides good results, although the use of IP networks for carrier grade services place new challenges for QoS provisioning. Recent years have seen many efforts to enable the engineering of IP based backbone networks. One of these initiatives is multiprotocol label switching. On the other hand recent research indicates that OSPF weight engineering allows traffic management and improved network utilisation. The scheme SAPOR that has been proposed in this paper can be located between pure OSPF routing and MPLS network engineering. SAPOR allows that once chosen routes are persistent for the duration of micro flows and aggregated flows can still be redirected. SAPOR is less of a scheme that competes with existing efforts in OSPF weight optimisation and more of an enabling technology. Overflow traffic routing and SAPOR can use many existing schemes and therefore lays the groundwork for future developments.

VIII. Acknowledgements

The authors would like to thank the Australian Telecommunications Cooperative Research Centre (ATcrc) for their financial assistance for this work.

References

[1] John Moy, *OSPF Version 2*, IETF, April 1998, RFC 2328.

[2] 3rd Generation Partnership Project, *About 3GPP*, October 2002, http://www.3gpp.org.

[3] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *Requirements for Traffic Engineering Over MPLS*, IETF, September 1999, RFC 2702.

[4] Ying-Dar Lin, Nai-Bin Hsu, and Ren-Hung Hwang, "Qos routing granularity in MPLS networks," *IEEE Commun. Mag.*, pp. 58–65, October 2002.

[5] Bernard Fortz and Mikkel Thorup, "Optimizing OSPF/IS–IS weights in a changing world," *IEEE J. Select. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.

[6] J. Murphy, R.J. Harris, and R. Nelson, "Traffic engineering using OSPF weights and splitting ratios," *In Proceedings of Sixth International Symposium on Communications Interworking of IFIP - Interworking 2002, Fremantle WA, October 13-16*, 2002.

[7] J. Harmatos, "A heuristic algorithm for solving the static weight optimisation problem in OSPF networks," *Global Telecommunications Conference, 2001. GLOBECOM '01*, vol. 3, pp. 1605 –1609, 2001.

[8] T.Ye, D. Harrison, B. Mo, B. Sikdar, H.T. Kaur, S. Kalyanaraman, B. Szymanski, and K. Vastola, "Traffic management and network control using collaborative on-line simulation," *IEEE International Conference on Communications, 2001. ICC 2001*, vol. 1, pp. 204 –209, 2001.

[9] G. R. Ash, *Dynamic Routing in Telecommunication Networks*, McGraw-Hill, 1997.

[10] J. Regnier, F. Bedard, J. Choquette, and A. Caron, "Dynamically controlled routing in networks with non-DCR-compliant switches," *IEEE Commun. Mag.*, pp. 48 –52, July 1995.

[11] R.J. Gibbens, F.P. Kelly, and P.B. Key, "Dynamic alternative routing - modelling and behaviour," *Proc 12th International Teletraffic Congress (ITC 12), Turin Italy*, 1988.

[12] K. Kawashima and A. Inoue, "State- and time-dependent routing in the NTT network," *IEEE Commun. Mag.*, pp. 40–47, July 1995.

[13] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for internet load balancing," *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2000*, vol. 1, pp. 332 –341, 2000.

[14] David G. Thaler and Chinya V. Ravishankar, "Using name-based mappings to increase hit rates," *IEEE/ACM Trans. Networking*, pp. 1 – 14, February 1998.

[15] E. P. Rathgeb, "Modeling and performance comparison of policing mechanisms for ATM network," *IEEE J. Select. Areas Commun.*, vol. 9, no. 3, pp. 325–334, April 1991.