Recursive Routing in the Cognitive Packet Network

Peixiang Liu, Erol Gelenbe
Electrical and Electronic Engineering Department
Imperial College London, SW7 2BT London UK
Email: {p.liu, e.gelenbe}@imperial.ac.uk

***Abstract*—Applications such as streaming multimedia, Voice over IP (VoIP) and video teleconferencing require Quality of Service (QoS), and developing algorithms that find routes from a source to a destination which meet the users' QoS requirements under rapidly changing network conditions remains a technical challenge. Recently, the *Cognitive Packet Network* (CPN) has been proposed so as to offer adaptive QoS driven routing based on on-line sensing and monitoring. In this paper, we propose a recursive routing algorithm for CPN which breaks large scale route discovery problems into smaller ones. The solutions to those smaller routing problems are cached in the intermediate nodes of the network, which can be utilized by *Smart Packets* to solve larger scale routing problems. The experiments conducted on a 46-node network test-bed indicate that when recursive routing is deployed, the network connection establishment time can be drastically reduced without sacrificing the QoS of the routes discovered, and that the QoS that the users experience is also improved.**

I. Introduction

Applications such as streaming multimedia [1], Voice over IP (VoIP) and video teleconferencing are some of the main applications which require Quality of Service (QoS) from packet networks, and discovering routes from a source to a destination which meet users' QoS requirements under rapidly changing network conditions is a major challenge. Different approaches such as IntServ&RSVP, DiffServ and MPLS have been proposed by industry sources to provide QoS to the network users. In IntServ&RSVP [2], [3], [4], a packet is sent out to negotiate resource reservations with the network, so that the QoS of the connection is ensured. In DiffServ [5], [6], traffic is classified into different service types before entering into the DiffServ cloud. The traffic which has a higher QoS category type is given higher priority to use the network resources. Therefore, the QoS of certain classes of traffic is ensured at the expense of the service received by others. In MPLS [7], [8], [9], the edge routers analyze packets and attach a label for each packet based on the classification results. Then routers in the MPLS core network check only the label of the packet to determine where it should be forwarded. All the packets belonging to a particular flow are assigned the same label and take the same route. Since a dedicated route is associated with a particular label, QoS can be offered more easily with MPLS.

The *Cognitive Packet Network* (CPN) [10], [11] provides adaptive QoS driven routing based on on-line sensing and monitoring of network QoS. In CPN, *Random Neural Networks* (RNNs) [12], [13] reside at each router and make routing decision. CPN has three types of packets: *Smart Packets* (SPs), *Dumb Packets* (DPs) and *Acknowledgement* (ACKs). The SPs keep on exploring the network to discover routes from the source to the destination and collect QoS measurement. The corresponding ACKs feed the gathered QoS information to the RNNs at routers along each path and trigger the updating of the weights of RNNs, using *Reinforcement Learning* (RL) [14] based on the users' QoS goals so that the RNNs then make the routing decision adaptively. The ACKs also bring the routes discovered by the SPs back to the source and keep them in the *Dumb Packet Route Repository* (DPRR). SPs themselves call upon the RNNs to select their route based on the users' QoS need. DPs then carry the payload data using the routes obtained from DPRR. CPN's ability to satisfy different user QoS metrics based on RNNs and RL, distributed at each network node, was discussed in [15]. Its extension to a wireless ad hoc environment was covered in [16]. We have also demonstrated that CPN can discover the shortest paths when the hopcount is used as the QoS goal [17].

In this paper, we propose a QoS based recursive routing algorithm which breaks a large scale routing problem into smaller ones. A route $r_{(s,d)}$, where s is the source and d is the destination, can be obtained from partial routes $r_{(s,i)}$ and $r_{(i,d)}$, where i is an intermediate node between s and d. Therefore, any routing problem can be solved by combining the solutions of smaller routing problems. The intrinsic routing mechanism in CPN supports the recursive routing automatically since every ACK packet keeps not only the partial route from the intermediate node to the destination, but also the QoS of that route. However, to accelerate the route discovery process, partial routes and their QoS can be cached at intermediate nodes and SPs can then construct complete paths from the partial routes they find at intermediate nodes. However if these cached partial routes are based on "old" information, their QoS values may not be accurate and could lead to a reduction of overall QoS delivered by the network.

We have implemented the QoS based recursive routing algorithm with CPN as a Linux loadable kernel module. The experiments that we have conducted on a 46-node test-bed indicate that when the recursive routing algorithm is deployed, the network connection establishment time can be drastically reduced without sacrificing significantly the QoS of the routes that are discovered, and the QoS that the users experience is also improved. The experiments also show that sorting the routes in the DPRR by their QoS does not affect CPN's learning capability but improves the QoS the users experience since only those best routes are provided to the DPs.

The paper is organized as follows. Section II presents the QoS based recursive routing algorithm and its implementation. In Section III we report the experimental results and discuss the performance that we observe. Section IV concludes the paper.

II. Recursive Routing in CPN

A *route* starting at node s and ending at node d is a sequence of nodes $r_{(s,d)} = (s, \ldots, d)$, where no node in the sequence appears more than once. That a node i is in the route $r_{(s,d)}$ is denoted by $i \in r_{(s,d)}$. Let $R(s,d) = \{r^1_{(s,d)}, r^2_{(s,d)}, \ldots, r^n_{(s,d)}\}$ be the set of all distinct routes from node s to node d in the network.

For any node $i \in r_{(s,d)}$, the route $r_{(s,d)}$ is the concatenation of two shorter routes:

$$r_{(s,d)} = r_{(s,i)} \cdot r_{(i,d)} \tag{1}$$

where $\cdot$ is the concatenation operator. We use $p_{(x,y,z)}$ to represent a partial route, where x is the starting and y the ending node of the partial route, and z is the destination node of the whole route. Then (1) becomes:

$$r_{(s,d)} = p_{(s,i,d)} \cdot p_{(i,d,d)} = p_{(s,i,d)} \cdot r_{(i,d)} \tag{2}$$

Thus, searching for a route $r_{(s,d)}$ can be solved by discovering a partial route $p_{(s,i,d)}$ and another route $r_{(i,d)}$.

The basic idea behind QoS based recursive routing is to store those already discovered partial routes at intermediate nodes. Thus route $r_{(i,d)}$ would be stored at the intermediate node i if it is already found. Once a SP discovers a partial route $p_{(s,i,d)}$, it will have found a route $r_{(s,d)}$ if a route $r_{(i,d)}$ has already been cached at node i.

We have modified the CPN code to support recursive routing. Now, each router can be configured to cache the partial routes discovered by the SPs. The RNN runs at every router to make routing decisions, and as in traditional CPN, the SPs always enquire the RNN for the routing decision. The arrival of the ACKs triggers the updating of the weights of the RNN, and the previous routing decision is either "rewarded" or "punished" depending on the QoS those packets experienced who have taken the RNN's routing advice. Therefore, the RNN can always make the routing decision adaptively according to the current QoS measurement of the network.

If a router is configured to support recursive routing, the DPRR at that router is used to cache the partial routes. There are two advantages to this approach. Firstly, since the DPRR structure already exists at each router for CPN to store the whole routes discovered by SPs, it is not necessary for us to create a new data structure to hold the partial routes. For any router, all the routes in its own DPRR are whole routes, while the routes kept in the DPRRs of other routers are partial routes that can serve *other* routers. Secondly, by storing the partial routes in the DPRR, the packets initiated from that node may reuse them if a cached "partial route" of another router happens to be the desired "whole route" of itself. In that case, a valid route is already available in the DPRR even before the SPs are sent out to search for one. In addition, the router which supports recursive routing treats the packets passing through differently:

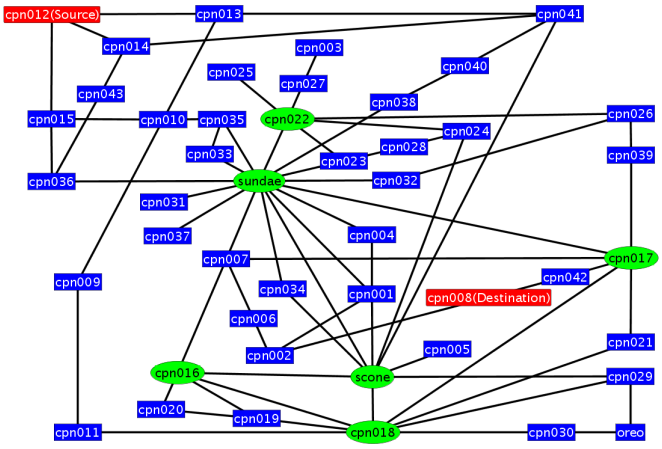

Fig. 1. The 46-node test-bed topology

- SP: when a SP arrives at the router, the key (i, d, p) is used to look up a cached partial route entry in the DPRR, where i is the current node, d is the destination node that SP intends to reach and p is the protocol type. If there is a matched entry in the DPRR, the partial route discovered by the SP so far, $p_{(s,i,d)}$, is concatenated to the partial route which is cached in the DPRR, $r_{(i,d)}$, to form a whole route from the source s to destination d, $r_{(s,d)}$. The whole route is put into a *Recursive ACK* (RACK), which is sent back to the source as a normal ACK packet. The SP continues searching for the destination d in order to collect new routing information. If no matched entry is found, the SP just jumps to the next hop which is advised by the RNN to continue its route discovery.
- SACK/RACK: When a SACK or RACK arrives, the partial route from the current node to the destination is cached in the DPRR with its QoS. The weights of the RNN are also updated based on the QoS information the ACKs carry.
- DP and DACK are dealt with in the same way as in the conventional CPN.

III. Performance Evaluation

Experiments were conducted on the 46-node test-bed (Fig. 1) to demonstrate the improvements that the recursive routing brings to CPN. In each experiment, the UDP traffic at 10Mbps were sent from source node, cpn012, to the destination node, cpn008, with hopcount as the QoS goal. Tcpdump ran at the source and destination node to capture the packet header of each packet arrived or departed. Each experiment lasted 60 seconds, during which about 73,000 DPs were sent out from the source to the destination. The same experiment was repeated 5 times. The smart packet sending rate was set as 10% in all the experiments.

Every set of experiments were run under four different settings:

- S-NR: Sorting on DPRR, no recursive routing. Here "Sorting on DPRR" means that the routes in the DPRR are ordered according to their QoS, rather than according to how recently they have been discovered. Under the setting of "no recursive routing", routers are not configured to cache the partial routes.
- NS-NR: No sorting on DPRR, no recursive routing. "No sorting on DPRR" means that the routes are ordered by the time when they were discovered, so that the route a DP uses is the newest route brought back by the ACKs.
- NS-R: No sorting on DPRR, recursive routing.
- S-R: Sorting on DPRR, recursive routing.

When recursive routing is used, the following specific nodes were configured to cache the partial routes: *scone*, *sundae*, *cpn016*, *cpn017*, *cpn018* and *cpn022*. Those nodes were selected just because they have more neighbors than others, and each of them has more than 5 neighbors. Since we use hopcount as the QoS goal, the effect of the QoS of each link is not included. Note that the more neighbors a node has, the higher the possibility is that it will be chosen to forward traffic, so that the partial routes that are cached at the node are reutilized more frequently.

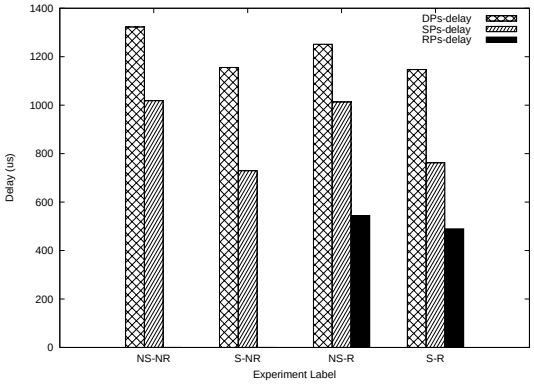

Fig. 2. Average Delay

A. Average Delay

The average round trip delay is reported in Fig. 2. The value of average delay has different meaning for different type of packets. DPs-delay denotes the QoS the users experience; SPs-delay denotes the average connection establishment time, i.e. how fast the SPs can discover a route from the source to the destination; while RPs-delay is the average connection establishment time when the SPs take advantage of the cached partial routes in the intermediate nodes. In the case of NS-R, RPs-delay is $544.34\mu s$ and SPs-delay is $1013.24\mu s$. While in the case of S-R, RPs-delay is $489.76\mu s$ and SPs-delay is $762.27\mu s$. Note that RPs-delay is much lower than SPs-delay in both cases, which means the route discovery time can be reduced dramatically when the recursive routing algorithm is used. We also notice that the DPs-delay of NS-R ($1251.48\mu s$) is smaller than that of NS-NR ($1322.62\mu s$), which means the QoS that the users experience is also improved when recursive routing is applied. No matter if the recursive routing is configured or not, the DPs-delay always decreases if the sorting is applied on the DPRR which is reasonable since in that case the DPs always use the best route in the DPRR.

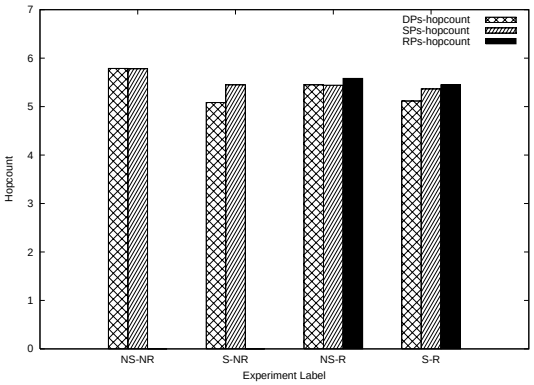

Fig. 3. Average Hopcount

B. Average Hopcount

The average hopcount is reported in Fig. 3. The average hopcount denotes the learning capability of CPN since we use hopcount as the QoS goal. The smaller the average hopcount is, the better the performance is. In Fig. 3, we notice that the SP-hopcount of NS-R (5.44) is smaller than that of NS-NR (5.78), which imply that recursive routing improves the average quality of the routes the SPs discover. In case of S-NR and S-R, DPs-hopcount is 5.08 and 5.11 respectively. We notice that both of them are close to 5, which is the shortest distance from the source to the destination. When the sorting algorithm is applied on the DPRR, the routes are ordered by the length of the routes. Therefore, the best(shortest) route in the DPRR is always used by the DPs. When sorting algorithm is not used, the average DPs-hopcount is almost the same as the average SPs-hopcount because the DPs use whatever the SPs discovered.

C. Path Usage

1) NS-NR: In Fig. 4 (Top), we show the length of all the paths brought back by the SPs and how frequently each path is discovered by different SPs in the case of NS-NR. From the figure, we can see in total 89 different routes are discovered by the SPs, 4 of which are length of 5 and 9 of which are length of 6. Among all the SPs sent out, 67.22% of them find the shortest path and 25.15% of them find the routes which are length 6. Since there is no sorting, the routes used by the DPs depends on what the SPs discover. in Fig. 4 (Bottom), we note 83 of 89 routes discovered by the SPs are used by the DPs to carry the payload. The more times a route is discovered by the SPs, the more frequently it is used by the DPs. For example, 67.21% of the DPs use the shortest path, which is almost the same as the percentage of the SPs that discover the shortest path (67.22%). This pattern can also be observed from the similarity of the curves on both figures.

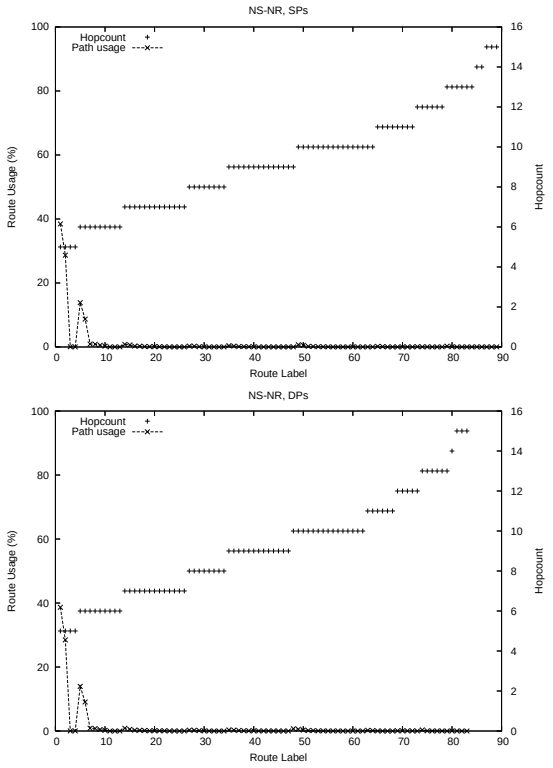

Fig. 4. Path usage in NS-NR (Top: SPs, Bottom: DPs)

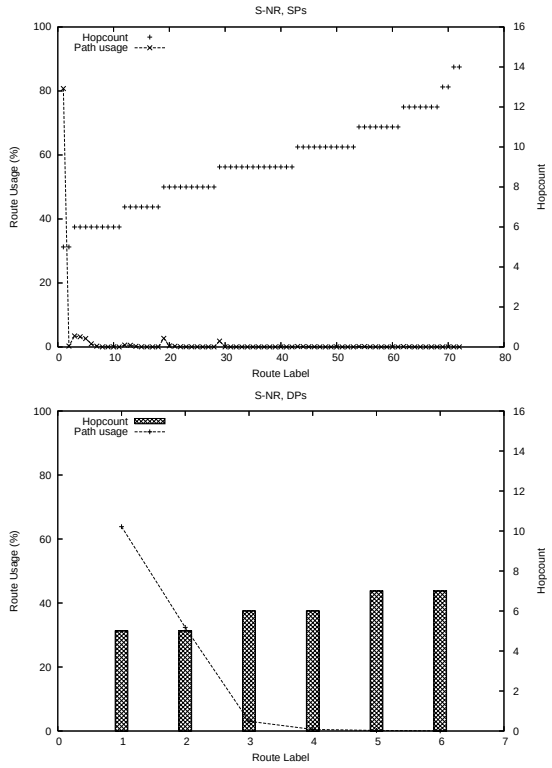

Fig. 5. Path usage in S-NR (Top: SPs, Bottom: DPs)

2) S-NR: Fig. 5 shows the path usage in the case of S-NR. 72 different routes are discovered by the SPs and 81.04% of them find the shortest path. Due to the sorting, only the best routes are used by DPs. Fig. 5 (Bottom) indicates only 6 shorter routes are used by the all the DPs, with 96.22% of which using the shortest paths.

3) NS-R: Fig. 6 shows the path usage in the case of NS-R. From the figure, we notice that both SACKs (Top) and RACKs (Middle) bring back high quality routes. SACKs bring back 69 routes in total and 84.14% of them are the shortest paths; RACKs bring back 77 routes and 84.25% of them are the shortest paths. Since the routes are not sorted, the DPs simply use the paths what the SACKs or RACKs bring back. The path usage pattern of DPs is similar to the distribution of those routes brought back by the SACKs and DACKs.

4) S-R: Fig. 7 shows the path usage in the case of S-R. The same as in the case of NS-R, both the SACKs (Top) and RACKs (Middle) discover the shortest path. SACKs bring back 90 routes in total and 87.69% of them are the shortest paths; RACKs bring back 122 different routes and 84.87% of them are the shortest paths. Since the routes in DPRR are sorted, only 8 shorter routes are used by DPs, 97.57% of which use 4 of the shortest paths.

D. Routes Diversity

We observed another interesting phenomenon: RACKs bring back more routes than the SACKs do. In the case of NS-R, RACKs discover 77 routes while SACKs find only 69 of them. In the case of S-R, RACKs discover 122 routes in total and SACKs find only 90 of them. The intermediate router caches the partial routes, which were discovered by the previous SPs. When the following SPs arrive at this intermediate route, they take the cached partial routes and return as the RACKs. which means any route brought back by RACKs is the concatenation of two partial routes discovered by two different SPs at different time. Therefore, recursive routing also helps to increase the diversity of the routes discovered by SPs.

IV. Conclusions

We have proposed a QoS based recursive routing algorithm which can break a large scale routing problem into some smaller routing problems. Partial routes are cached in the intermediate nodes which can be used to provide a fast estimate of the "best" QoS based route that is needed by an arriving packet. We conducted experiments on a 46-node network test-bed and observed that when recursive routing is applied, the average time for a smart packet to discover a valid route is reduced dramatically, the QoS that the users experience is also slightly improved, and more routes are discovered by the SPs since one SP can make use of the partial routes discovered by other SPs so that the diversity of the routes increases. We have also observed that selecting the paths based on QoS rather than on the "freshness" does not reduce the adaptivity of CPN, and that it improves the QoS experienced by the users by recommending only the best routes.

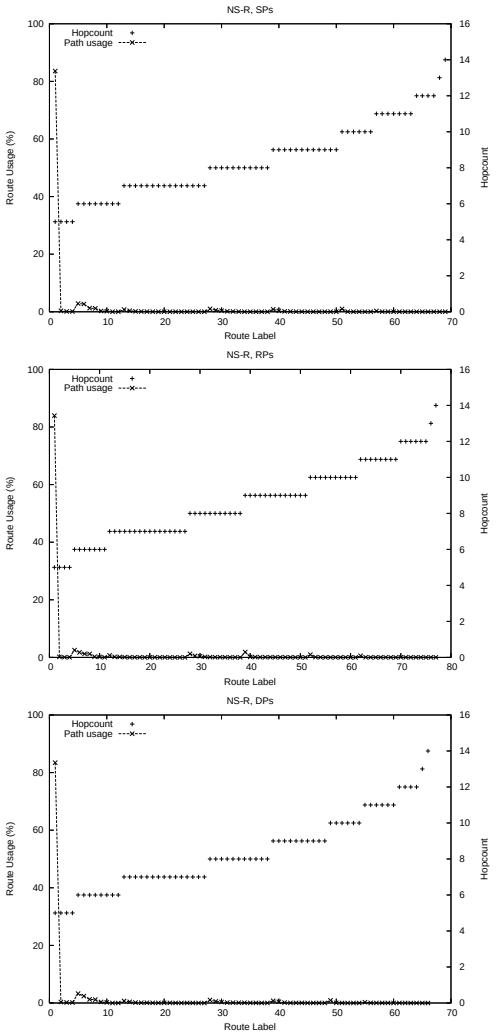

Fig. 6. Path usage in NS-R (Top: SPs, Middle: RPs, Bottom: DPs)

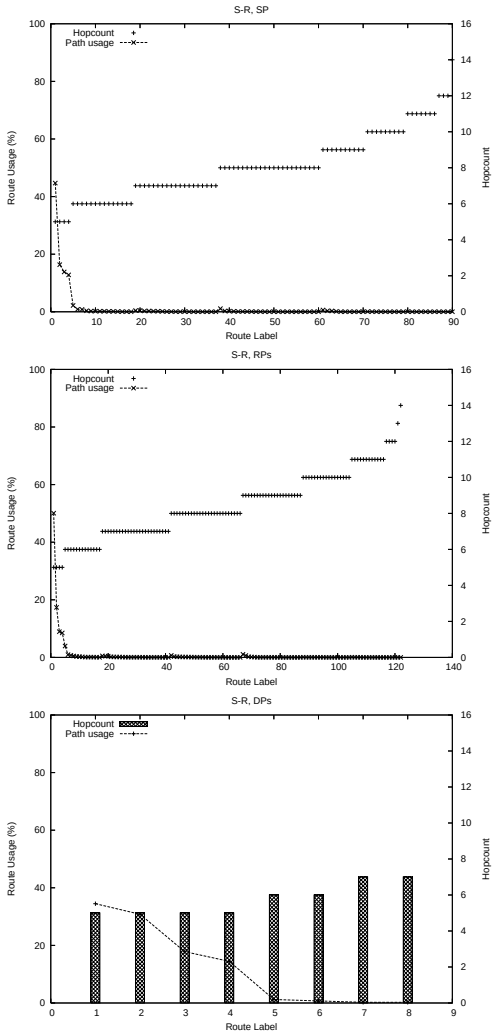

Fig. 7. Path usage in S-R (Top: SPs, Middle: RPs, Bottom: DPs)

REFERENCES

[1] C. Cramer, E. Gelenbe, and H. Bakircioglu, "Low Bit-rate Video Compression with Neural Networks and Temporal Subsampling," *Proceedings of the IEEE*, vol. 84, pp. 1529–1543, October 1996.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview." RFC 1633 (Informational), June 1994.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification." RFC 2205 (Proposed Standard), September 1997. Updated by RFCs 2750, 3936.

[4] J. Wroclawski, "The Use of RSVP with IETF Integrated Services." RFC 2210 (Proposed Standard), September 1997.

[5] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." RFC 2474 (Proposed Standard), December 1998. Updated by RFCs 3168, 3260.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service." RFC 2475 (Informational), December 1998. Updated by RFC 3260.

[7] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture." RFC 3031 (Proposed Standard), January 2001.

[8] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "MPLS Label Stack Encoding." RFC 3032 (Proposed Standard), January 2001. Updated by RFC 3443.

[9] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP Specification." RFC 3036 (Proposed Standard), January 2001.

[10] E. Gelenbe, R. Lent, and Z. Xu, "Measurement and Performance of a Cognitive Packet Network," *Computer Networks*, vol. 37, no. 6, pp. 691–701, 2001.

[11] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware Networks and QoS," *Proceedings of the IEEE*, vol. 92, pp. 1478–1489, September 2004.

[12] E. Gelenbe, "Learning in the Recurrent Random Neural Network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[13] E. Gelenbe and K. Hussain, "Learning in the Multiple Class Random Neural Network," *IEEE Trans. Neural Networks*, vol. 13, pp. 1257–1267, June 2002.

[14] U. Halici, "Reinforcement Learning with Internal Expectation for the Random Neural Network," *Eur. J. Oper. Res.*, vol. 126, no. 2, pp. 288–307, 2000.

[15] E. Gelenbe, M. Gellman, R. Lent, P. Liu, and P. Su, "Autonomous Smart Routing for Network QoS," *Proc. First International Conference on Autonomic Computing (IEEE Computer Society)*, pp. 232–239, May 2004.

[16] E. Gelenbe and R. Lent, "Power-aware ad hoc Cognitive Packet Networks," *Ad Hoc Networks Journal*, vol. 2, pp. 205–216, July 2004.

[17] E. Gelenbe and P. Liu, "QoS and Routing in the Cognitive Packet Network," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 517–521, June 2005.