QoS Routing for MPLS Networks Employing Mobile Agents

Sergio González-Valenzuela and Victor C. M. Leung
The University of British Columbia

Abstract

The implementation of new networking technologies, such as multiprotocol label switching and differentiated services, will introduce powerful features to the near-future Internet backbone, making a significant contribution to the overall end-to-end provision of quality of service. However, to achieve such an improvement these technologies require not only effective support from current routing algorithms, but also enhanced capabilities, which are currently being developed. To contribute to this development, a novel and powerful scheme is introduced in this article that provides a means of supporting QoS routing through the use of mobile software agents. Specifically, we describe the use of mobile agents to efficiently realize multipoint-to-point routing trees by means of the Wave paradigm, while satisfying the QoS requirements of the set of traffic streams involved in the process. Both benefits and important issues to be considered when using mobile agent schemes in QoS routing are further stressed.

During the past few years the field of mobile agents has gained a good deal of attention as the scope of their usefulness has expanded, now reaching an increasing number of proposed solutions for applications, including e-commerce, resource reservation, data mining, and even lower-level networking tasks [1]. Mobile agents can be programmed to work as standalone software entities capable of performing tasks on behalf of the user, but have also been employed to emulate the behavior of insect colonies, whose task is to achieve a collective objective in a highly cooperative manner, as shown in [2–4]. Thus far, most implementations are oriented to support applications at the higher layers as *brokers*, which run on top of the software infrastructure readily available at the networks' end systems. On the other hand, some platforms for mobile agents have been used only as simulation tools for the implementation of experimental solutions to problems found at lower-level stages in communications networks. This is easily understood, since there is currently no support available for mobile agent implementations throughout the existing networking infrastructure (e.g., routers, switches). This is unfortunate because the inherent benefits of mobile agent technology are therefore not fully taken advantage of. In [5], several advantages of using mobile agents are clearly defined. It can be seen that while many higher-layer applications do benefit from the use of mobile agents, these same features could bring even more remarkable benefits if a mobile agent infrastructure were available for use at the network layer level throughout intermediate nodes in a communications network (e.g., the Internet backbone).

In the next section, special emphasis will be put on the use of mobile agents as a plausible solution to solving routing problems in future Internet architecture implemented to enable the provision of quality of service (QoS). We will then briefly introduce the Wave paradigm and its advanced fea-

tures to support implementation and propagation of mobile agents. Then we will propose a novel routing scheme in support of QoS provisioning at the network layer, based on the Wave paradigm. Finally, we show some of the results obtained, and conclude this article with the lessons learned during this work.

The Role of Routing in the Provision of QoS

To validate the importance of providing a robust and efficient routing platform for supporting QoS in the near future Internet, it is first necessary to determine a possible evolution scenario for the physical infrastructure of the Internet backbone. At the intermediate layers level, both the multiprotocol label switching (MPLS) and differentiated services (DiffServ) architectures have been considered plausible technological solutions to satisfy future networking requirements. MPLS arises as a natural stage in the evolution of the label-swapping paradigm, offering improved performance in the organization and management of data streams [6]. The basic operational concept of MPLS is fairly simple. Labels are exchanged among neighboring network nodes, and are used to identify the forwarding equivalence class (FEC) of data packets. When a packet enters an MPLS domain, a label with a given value is attached to it depending on the FEC to which the respective packet stream belongs, which is commonly inferred from its IP destination address. At each node the packet's label is used to perform an indexed search in a table that identifies the output port for the next hop and the new label to be used to replace the old one for forwarding the packet over the next hop. Packets are thus forwarded over the MPLS network based on labels attached to them in each hop, and no further analysis of the network layer header is necessary. Moreover, MPLS does not define a new layer in the classical open systems interconnection (OSI)

communications model; instead, it can be seen as a bond between any protocols operating at the network and data link control (DLC) layers. These and other features give MPLS its superiority, and distinguish it as a promising new technology for supporting enhanced networking tasks, such as the design of virtual private networks (VPNs), traffic engineering, and explicit routing.

In a similar manner, DiffServ [7] has been favored as a viable QoS control framework for the Internet backbone, in part because it is highly scalable [8]. The basic operating principle of DiffServ establishes that data streams with similar forwarding precedence can be given equivalent scheduling treatment by grouping them at the supporting routers/switches. Nevertheless, DiffServ cannot exist as a standalone technology for supporting QoS; it must work concurrently with a forwarding technology that supports the aggregation scheme, which is the inherent operating nature of DiffServ for managing data streams. Clearly, MPLS offers the means to accomplish such a task, as already proposed in [9, 10]. MPLS provides support for data aggregation through the use of FECs, which can be seen as an organized way of assigning and managing labels, to efficiently achieve the desired aggregation scheme. As explained in [6], data traffic can be partitioned into FECs for each IP address prefix in a routing table. This may result in a number of data streams that belong to the same FEC following the same route, since they may share the same egress node according to their destination IP prefix. Thus, several FECs may be further aggregated into a single FEC within the same domain, for which a single label may be used. However, organized label distributions in MPLS require an external routing protocol to indicate what the routes will be, as well as a label distribution protocol.

A closer look at the requirements of an external routing protocol to enable FEC aggregation reveals the need to support multipoint-to-point (mp2p) connections; that is, a number of data streams originating at separate places that later converge at a common node in the network to either continue along the same path or perhaps diverge. Note that such data streams may not necessarily participate in a common data transfer session, as in a multicasting scheme [11].

A serious problem arises here since the current interdomain routing protocols, such as Open Shortest Path First (OSPF) [12], do not provide support for establishing mp2p routes. The question now becomes, how can the current routing protocols overcome this drawback? The lack of the required routing support will have a direct impact on the provision of QoS, should a DiffServ-over-MPLS network be deployed for such purposes. It can also be seen that performing QoS routing under classical paradigms might become troublesome. A random number of QoS-routed connections may cause a network to behave in a stochastic manner. The resulting network state uncertainty may trigger constant routing table updates, which in turn generate more network control traffic due to both the growing number of updates and the increased size of the routing tables that reflect current information on resource availability.

It is within this context that mobile agents come into play. Excessive routing traffic could be avoided by building multiple routes to honor different types of connection QoS requirements according to predefined service level agreements (SLAs), which are an intrinsic part of the DiffServ architecture. Thus, mp2p paths (trees) can be created either statically or dynamically, according to individual premises defined by an Internet service provider. A novel approach to solving this problem can be making efficient use of the Wave paradigm for the implementation and propagation of mobile agents, discussed next.

■ Figure 1. The layered Wave model and its role in the proposed routing scheme.

The Wave Paradigm

The philosophical origins of the Wave paradigm date back to the early '70s according to its creator, Dr. Peter Sapaty. At that time, the mobile agent concept was used to perform different types of computations in a hybrid network, comprising computers running distinct software engines. The Wave platform evolved substantially and saw its first enhanced prototype version released to the general public in the early '90s. Ever since, Wave has been used and tested in a number of universities in different countries, and has been part of projects conducted by companies such as Ericsson and Siemens, as well as the Defence Research Agency in the United Kingdom [13].

The Wave paradigm can be considered a radical technology created to process information in open and distributed environments, such as communication networks. Wave may be described as a set of defined strings representing operations, functions, and data able to propagate across a communication network [13]. Mobile agents designed within the Wave framework (also known as *waves*) may start their algorithmic execution at any node in the network and propagate in a highly parallel manner, while “conquering” space as the code execution evolves in time. In the Wave paradigm, a portion of a communication network may be regarded as conquered when the Wave agents use the information retrieved from the nodes they traverse and incorporate it into a virtual knowledge network (KN) they build. A KN created by a Wave system running on top of network nodes do not embrace or control the actual physical network the way closed systems traditionally do. Rather, it spreads and treats a network as an open system, or distributed supercomputer, thus facilitating the creation and expansion of the KN. This helps in the solution of problems in a parallel manner, without any centralized supervision. The Wave platform provides major features that are particularly suited, but not limited, to use in telecommunication applications. The most important features are strong migration, parallel processing, synchronous/asynchronous navigation, agent collaboration, flexibility, autonomy, fault tolerance, and compactness. These features make Wave a viable tool for use in telecommunications applications.

Mobile agents (waves) can be explicitly used to navigate a given network in order to build a KN on specific information they obtain (e.g., network congestion, resources availability). Furthermore, Wave programs are *conveniently compact*, typically from 20 to 50 times shorter than equivalent C/C++ or Java programs [14]. Figure 1 shows the layered structure of the Wave model, and its interaction with the routing scheme that will be explained shortly. Figure 2 shows a sample Wave program for illustrative purposes. Such a wave string contains the required instructions to create a colony of mobile agents that find simple shortest paths in a network. Starting in node *a* in a hypothetical network, the wave agent clones itself in a “benevolent” virus-like fashion, and hops to all of the neighboring nodes. Upon reaching a node, each of the cloned

@#a.F=0.RP(N~,F<N.N=F.N1=P.\$F+L)

■ Figure 2. Finding simple shortest paths using Wave.

agents is only allowed to continue its navigation through the network if the traveled distance is less than that recorded by previous agents (if any). Each wave agent repeats this process, until no further hopping is possible. The result is that every node will have a record of the next hop required to reach node a, regardless of the source. This is a clear example of how compact and powerful a wave mobile agent can be. Within the Wave paradigm, both the agent's code and its current execution state can move from one node to another by means of UDP packets. This feature is also used to create the Dynamic Track Layer for the establishment of a communication link among agents to further coordinate their spreading throughout the network, as shown in Fig. 1.

Wave agents have the capability of carrying along any information they may need while they travel through the network, which is only accessible to the agent that carries it (i.e., a private variable). Additionally, specific variables can also be created at the network nodes as a way of sharing information among agents (i.e., public variables). Information about the KN can be obtained by means of environmental variables supported by the Wave system. Wave can be used as an efficient platform for programming mobile agents to solve problems at intermediate layers of a communication network. Moreover, Wave provides support for interfacing with external programs, which is another handy feature, since any operation unsupported by Wave can be performed using external resources. A wave agent may use its interfacing capability to communicate with external programs (written in C, C++, Java, etc.) to accomplish specific tasks at the higher layers. Specific information on the current state of a network node can be obtained by external means and used to build a KN for a specific purpose, such as a QoS-based KN (defined next).

QoS Routing with Mobile Agents

All of the features previously mentioned are important factors contributing to the decision to use the Wave platform in this work. The routing scheme presented here is composed of two types of agents: one set of static agents used to obtain information on the availability of network resources, and a second colony of mobile agents deployed for the actual discovery of QoS-compliant routes. The only requirement for the network to support a virtual infrastructure for mobile agents is that the network nodes should be capable of accommodating a Wave interpreter, as shown in Fig. 3. Assuming also that the underlying hardware circuitry is capable of performing the actual data stream merge — say, virtual circuit (VC) merging in asynchronous transfer mode (ATM) [15] — the routing system designed for DiffServ-over-MPLS based on mobile agents works as follows.

The QoS-Based Knowledge Network

For creation of a QoS-based KN, a colony of *static agents* can be created to individually reside at every network node with the purpose of monitoring the available resources in a DiffServ switch for later use in updating the QoS-KN. In this respect, the wave interpreters (WIs) running on top of the DiffServ/MPLS switches, whose identities can be directly mapped from their own network addresses, define the virtual nodes of the KN. Depending on the operational premises under which the static agents are programmed, information regarding availability of resources between two network

nodes can be mapped to a simple weighted scalar value. This value is therefore used to represent the magnitude of a certain QoS metric value, such as bandwidth, delay, jitter, or perhaps a combination of them. The agent can update the value of each virtual link to all neighboring nodes (i.e., its portion of the KN). Figure 4 shows an example using bandwidth availability information of a particular link between two nodes in a network.

The benefits of this scheme can readily be seen. Compared to classical routing, there is no need to perform a whole routing table recomputation to update the QoS information at remote locations each and every time there is a change in the state of the individual switch. Therefore, the need to keep larger routing tables holding information on all QoS-related parameters is eliminated, since the values of the virtual links are meant to fulfill the same purpose. Moreover, there is no need to flood the network with routing tables to and from every network node, either periodically or triggered by significant changes. The KN will implicitly record QoS information in a completely distributed manner, which also results in reduced network traffic management and the elimination of routing with inaccurate information. The QoS information mapped to the KN is then ready for the second colony of agents to use to discover the routes.

Discovery of QoS-Compliant Routes

The objective here is to define a tree with minimized cost, given that a grade of service has been assigned to each edge in a graph, while being capable of providing such service as required. This problem is better known as the constrained Steiner Minimal Tree (SMT) in graph theory, which has been extensively studied over the past few decades [16]. SMTs are of special interest in telecommunications, since they represent problems related to networking issues, including routing. In reality, current routing protocols do not implement SMT problem solving algorithms because of their high complexity. Instead, they employ other methods for obtaining minimum spanning trees, which helps to keep the complexities of routing computations tractable.

The realization of this goal leads to minimization of network resources (i.e., both the number of communication links and switches/routers in a network, as well as reducing the number of MPLS labels used), while grouping similar data streams for better management. Thus, it is necessary to find a QoS-compliant route for each individual ingress node whose final path to the root may coincide, at least partially, with those of the other ingress nodes. The higher the number of partial overlapping routes, the better. A heuristic solution for the cited problem is therefore presented as follows in this work, which is directed to finding the maximum

■ Figure 3. The Wave Interpreter embedding in an MPLS switch.

number of edges where individual path intersections occur to minimize the overall cost of the tree. As a first step, all possible shortest path trees (SPTs) from every individual ingress node to the egress node are obtained. Then the intersections of all the SPTs found are evaluated by the agent colony to identify the edges of individual paths that contribute to realizing the final objective by pruning the paths not contributing to such optimization. A more detailed explanation of how to implement these procedures by means of the Wave paradigm now follows.

Recall the assumptions made for the SLAs within the Diff-Serv architecture that two types of mp2p connections are contemplated: static and dynamic. The static mp2p routes with prereserved resources can be set to honor connection requests by users with either best-effort or assured (probabilistic) services [8]. The protocol used to discover the mp2p routes works as follows. During initialization of the MPLS switches, a combined request from ingress switches is made to establish a route toward a common egress node to create the mp2p tree. For example, if the network consisted of five edge nodes, A, B, C, D, and E, the combinations of mp2p trees would follow the pattern B-C-D-E $\rightarrow$ A, A-C-D-E $\rightarrow$ B and so on. Provided that a specific QoS KN already exists, a coordinated colony of mobile agents is launched from all ingress nodes (say, A, B, C, D) in order to find the mp2p tree toward the common egress node (say, E). The agent is set to travel over the links of the QoS KN that meet the QoS constraint sought. Therefore, the routes being searched will inherently comply with the QoS requirements previously defined. The next priority is to search for the mp2p routes while minimizing network resources. Hence, as a first step the Wave engines at each individual node clone the original agent and launch as many copies of it as QoS-compliant outgoing links the node has, as defined by the QoS KN. Upon reaching an intermediate node, the partial distance traveled by individual agents is recorded in local variables. Any agent having traveled a shorter distance than one previously recorded by a different agent coming from the same origin node is allowed to continue its journey toward the destination. An agent carrying a longer distance is immediately discarded at any intermediate node as soon as this condition is detected. To finish this initial procedure, a second set of agents is similarly launched, but this time they are only allowed to continue their traversing toward the destination if the current traveled distance equals that recorded by

Figure 4. Bandwidth information used to update an individual link weight within a QoS KN.

the previous set of agents. A remarkable feature of this process is that not one, but all possible shortest paths on top of the QoS KN are found. Another important result is that the routes found are ensured to contain no cycles, since any agent recording a distance larger than a previous one is destined for discard, which is the case when a wave returns to an earlier visited node. This implies that *no further verification process from the MPLS part is necessary to ensure cycle-free paths*. As a second step of the mp2p routing procedure, a group of waves is launched from the multiple origin nodes toward the destination node, with the objective of identifying possible data merge nodes. The agents travel only through the shortest paths discovered previously. Upon reaching an intermediate node, agents first check for previous visits by other agents also launched by the same origin node that may be traveling through parallel (alternate) shortest paths. If the agent is the first to visit that particular node, it sets a flag to advertise its visit; otherwise, it continues traversing the path. Therefore, every agent coming from a different ingress node and reaching an intermediate node for the first time sets a unique flag that asserts such a visit.

Finally, the agents are sent back to travel over the same shortest paths previously found to determine the topology of the mp2p tree. During this process, they check every node reached to determine the number of visits from prior agents traveling from distinct origin nodes that left identifying marks. The more incidences of previous visits found, the more "weight" the path is credited. Thus, the weight credited to each individual shortest path is directly proportional to the degree of overlapping instances individually found. Upon finishing this procedure, every agent records the traversed path and its associated weight at the destination node. Any subsequent agent coming from the same origin node will only override the previous records if it brings a weight higher than the one recorded previously. The result is that the egress node will contain records of all the shortest paths, from every origin node to itself, that comply with the QoS constraint specified and contribute to the realization of the mp2p tree with minimized cost.

Figure 5. A graphic depiction of the mp2p routing process with mobile agents in an MPLS autonomous system: a) agents search for shortest paths; b) possible merge nodes are identified; c) definition of final routes.


```

Repeat until destination reached
{
  Clone Wave and hop through all QoS-compliant links
  If first wave to reach node,
    record origin node and distance traveled
  Else
    If distance brought < previous recoded
      Update new distance for respective origin
    Else Wave dies
}
Repeat until destination reached
{
  Record node traversed
  Clone Wave and hop through all QoS-compliant links
  If distance == previous, continue navigation
  Else Wave dies
}
Repeat until destination reached
{
  If first wave to reach node,
    record origin node and increment 'visits' variable
  Else hop to next node in current path followed
}
Repeat until destination reached
{
  Add up to frontal variable the number of 'visits' _
    received by current node
  Hop to next node in current path followed
}
If weight of path brought > weight of alternate path _
  previously recorded, override previous records
Else discard path brought

```

■ Figure 6. The algorithm used in the construction of mp2p trees.

Using the interfacing feature of Wave, the identities of the nodes found for each path (e.g., their IP addresses) can be passed to the MPLS switch. The MPLS switch then assigns the proper labels and calls on a label distribution protocol to set up the label switched path (LSP). The details of this procedure are out of the scope of this document. A graphic depiction of the mp2p routing process is shown in Fig. 5, while Fig. 6 shows the algorithmic structure of the agents used for such purpose.

In a dynamic routing case, the exact same procedure would apply, except the mp2p tree would be dynamically reconfigured every time a given connection either joins or leaves the tree. The routing procedure would have to be explicitly executed to find the best paths at the moment the connection is requested. This scenario could be used to honor requests of users with premium SLAs that require enhanced QoS provision from the network. The setting up of the paths is dynamic in nature, since it would be wasteful to reserve resources and leave them unused for an indefinite period of time.

Results

The algorithm previously described has been implemented using a Solaris-UNIX version of Wave written in C, which is available for public download on the Internet [17]. A dynamic scheme was created and implemented to simulate a number of random connections requesting to either join or leave existing mp2p connections. The arrival pattern of mobile agents at a given network node was monitored in intervals of 1 s during the simulations. Figure 7 shows an 1800-s sample of agent arrivals. It can be

readily seen that the arrivals of agents follows an utterly bursty behavior pattern. The mathematical modeling for studying the delay properties of this routing method becomes rather difficult. However, it is important to determine a suitable queuing model to describe the delay experienced by an agent reaching a node during the routing process. During the conducted simulations, the arrival pattern of connection requests was observed to follow an exponential distribution. For the sake of simplicity, it is assumed that the service time of the agents is deterministic, since the operations they perform at the network nodes are short, very similar, and well defined. Since there is a single Wave interpreter at every network node, the previous factors lead to the definition of an M/D/1 queuing model. However, when connection requests trigger the launching of the mp2p routing algorithm, not one but a whole batch of agents is actually generated. Therefore, the queuing model becomes an M/D/1 model with batch arrivals, also expressed as M^[X]/D/1. This implies that an agent arriving at a network node has to wait for service in a queue formed by agents from both its own batch and separate batches if such is the case. After several simulations, it was observed that the previous assumptions were indeed correct, and that the queuing behavior at the Wave interpreters followed the forecast model. According to several goodness-of-fit tests conducted, the size of the agent batches was measured and found to follow a gamma distribution as depicted in Fig. 8.

The complexity of the designed algorithm has a logarithmic upper bound defined by the expression $O(N * \log^2 N * Q)$, where N = number of edge nodes and Q = number of QoS classes. The reason for this conveniently bounded expression can be explained by the very nature of the mobile agent paradigm. In classical routing schemes, QoS-compliant routes are obtained by combinatorial-like computations, whose complexity and time performance may deteriorate by the growing nature of the routing tables. An increase in the number of network nodes, the inclusion of QoS metrics values, and even the introduction of longer IP addresses imposed by the emerging IPv6 standard have a direct impact in exponentially bounded expressions, which are commonly found in combinatorial algorithms [18]. Conversely, the ability of the agents to process the entirely distributed QoS information in a fully parallel manner boosts their performance as a whole, since no combinatorial computations were required. Not only Wave, but any mobile agent programming platform can benefit from this advantage.

■ Figure 7. Agent arrival pattern during simulations.

Conclusions

The combination of various factors was seen to play an important role in the traffic behavior and number of agents processed in the system tested. Among them are the degree of a node, the topology of the network, the migration pattern of the agents (determined by the physical location in the network of the destination node), and the QoS characteristics of each link.

Accordingly, the cooperative nature of the agents determines the amount of traffic generated by them during the routing sessions. When simultaneous routing sessions take place, a whole colony of agents is launched to discover the mp2p tree; therefore, the number of agents in the network may increase abruptly. This accounts for the bursty behavior observed over longer periods of time, as shown in the snapshot of Fig. 7. According to the probabilistic distribution of the agent batches shown in Fig. 8, careful consideration should be taken when designing a routing system working within the same routing scheme (mp2p or p2mp), since a sudden increase of agents in the system may cause congestion in the queues of the agent processing units in the network.

The previous conclusion raises an important observation. One of the most remarked on claims attributed to mobile agents is that they are expected to contribute to the reduction of network traffic. This has been considered an intrinsic characteristic of the mobile agent paradigm. However, according to the results obtained in this investigation, this assertion seems to be more valid for the types of agents that work individually (e.g., in brokering or e-commerce applications). According to the previous discussions, mobile agents that perform collective tasks can be expected to generate more network traffic than other agent applications. However, this factor is not expected to undermine their overall performance.

In this regard, it can be concluded that the framework in which a mobile agent scheme is implemented should depend on the type of application for which they are designed. For instance, mobile agent applications whose task is primarily focused on information processing on higher layers at the network terminals can take advantage of programming platforms such as Java. On the other hand, applications where the primary task is accomplished in the middle layers might be better suited to frameworks that offer compact code and network-oriented navigation capabilities, such as Wave. Finally, it can also be concluded that, to reduce the number of agents in the network and the bandwidth consumed by them, more efficient navigation techniques have to be investigated and tested. Complex navigation methods might imply increased computational complexity of the agent algorithms. A deeper investigation and trade-off analysis would be required to establish an adequate equilibrium between these important factors, while including other important networking issues such as traffic engineering, connection admission control, and resource reservation.

Acknowledgments

The authors wish to thank Consejo Nacional de Ciencia y Tecnología (CONACyT) of México, the Canadian Natural Sciences and Engineering Research Council for partially supporting this research, and Dr. Son T. Vuong at the Department of Computer Science of UBC for his invaluable help.

References

- [1] T. Magedanz and A. Karmouch, "Mobile Software Agents for Telecommunication Applications," *Comp. Commun.*, vol. 23, 2000, pp. 705-7.
- [2] K. Oida and M. Sekido, "ARS: An Efficient Agent-Based Routing System for QoS Guarantees," *Comp. Commun.*, vol. 23, 2000, pp. 1437-47.

Figure 8. Distribution of the agent's batch size.

- [3] M. Dorigo and G. Di Caro, "Mobile Agents for Dynamic Routing," *Proc. 31st Int'l. Conf. Sys. Hawaii*, Jan. 1998.
- [4] N. Minar et al., "Cooperating Mobile Agents for Dynamic Network Routing," *Software Agents for Future Communications Systems*, Springer-Verlag, 1999.
- [5] D. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents," *Commun. ACM*, Mar. 1999.
- [6] E. Rosen et al., "Multi-Protocol Label Switching Architecture," IETF RFC 3031, Jan. 2001.
- [7] S. Blake et al., "An Architecture for Differentiated Services," IETF, RFC 2475, Dec. 1998.
- [8] X. Xiao and L. Ni, "Internet QoS: A Big Picture," *IEEE Network*, Mar./Apr. 1999.
- [9] I. Andrikopoulos and G. Pavlou, "Supporting Differentiated Services in MPLS Networks," *Proc. 7th Int'l. Wksp. QoS*, London, U.K., May 1999.
- [10] F. Le Faucheur et al., "MPLS Support of Differentiated Services," Internet draft, draft-ietf-mpls-diff-ext-07.txt, IETF, Aug. 2000, work in progress.
- [11] B. Wang and J. Hou, "Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols," *IEEE Network*, Jan. 2000, pp. 22-36.
- [12] G. Apostolopoulos et al., "QoS Routing Mechanisms and OSPF Extensions," IETF RFC 2676, Aug. 1999.
- [13] P. Sapaty, *Mobile Processing in Distributed and Open Environments*, Wiley, 2000.
- [14] S. Vuong and I. Ivanov, "Mobile Intelligent Agent Systems: Wave vs. Java," *IEEE Comp. Soc.*, Mar. 1996.
- [15] I. Widjaja and A. Elwalid, "Performance Issues in VC-Merge Capable Switches for Multi-Protocol Label Switching," *IEEE JSAC*, June 1999.
- [16] D. Cieslik, *Steiner Minimal Trees*, Kluwer, 1998.
- [17] The Wave Page, <http://www.zorn.ira.uka.de/wave/wave.html>, hosted by the Faculty of Informatics, Univ. of Karlsruhe, and the Dept. of Elect. and Elec. Eng., University of Surrey.
- [18] E. Crawley et al., "A Framework for QoS-based Routing in the Internet," IETF RFC 2386, Aug. 1998.

Biographies

VICTOR C. M. LEUNG [SM] (vleung@ece.ubc.ca) received his B.A.Sc. (Hons.) degree in electrical engineering from the University of British Columbia (UBC) in 1977, and was awarded the APEBC Gold Medal as head of the graduating class in the Faculty of Applied Science. He attended graduate school at UBC on a Natural Sciences and Engineering Research Council Postgraduate Scholarship and obtained his Ph.D. degree in electrical engineering in 1981. From 1981 to 1987 he was a senior member of technical staff at MPR Teltech Ltd., Canada, specializing in the planning, design, and analysis of satellite communication systems. He also held a part-time position as visiting assistant professor at Simon Fraser University in 1986 and 1987. In 1988 he was a lecturer in the Department of Electronics at the Chinese University of Hong Kong. He joined the Department of Electrical and Computer Engineering at UBC in 1989, where he is a professor and holder of the TELUS Mobility Research Chair in Advanced Telecommunications Engineering. He is also a member of the UBC Institute for Computing, Information and Cognitive Systems. He is a project leader and member of the Board of Directors in the Canadian Institute for Telecommunications Research, a Network of Centers of Excellence funded by the Canadian Government. His research interests are in the areas of architectural and protocol design and performance analysis for computer and telecommunication networks, with applications in satellite, mobile, personal communications, and high-speed networks. He is a voting member of ACM.

SERGIO GONZALEZ-VALENZUELA (sergiog@ece.ubc.ca) received his B.S. degree in electronics engineering from the Instituto Tecnológico de Sonora (México) in 1996. He held various engineering positions in Mexican industry, and went on to pursue a Master of Applied Science degree in telecommunications at the University of British Columbia, from which he is a recent graduate. His main research interests relate to MPLS networks, routing, wireless communications, and the use of mobile agents to solve diverse data network issues.