Macro-routing: a new hierarchical routing protocol

Sanda Dragos and Martin Collier
School of Electronic Engineering, Dublin City University, Dublin, Ireland
Email: {dragoss, collierm}@eeng.dcu.ie

***Abstract*— In a continually evolving Internet, tools such as *Quality of Service routing* must be used in order to accommodate user demands. QoS routing raises scalability issues within very large networks, which can be avoided by using hierarchical routing strategies. However, such strategies can lead to inaccurate path selection due to the aggregation process. To avoid such problems we propose a hierarchical routing protocol, called *Macro-routing*, which can distribute the route computation more efficiently throughout the network using mobile agents. It processes more detailed information than conventional hierarchical routing protocols so is more likely to find the best path between source and destination. Also, by using mobile agents, more than one available path can be found. This provides a fast recovery mechanism, where no protocol restart is needed in a failure situation.**

I. Introduction

Quality of Service (QoS) routing is the process of identifying efficient paths that can satisfy QoS constraints (e.g. bandwidth, delay, delay variation). Thus, QoS routing requires frequent updating of state information. Such update messages consume significant network bandwidth and processing power. This creates scalability concerns for large networks.

We can reduce such protocol overhead by reducing the frequency of update messages. This reduces the network ability to quickly adapt to changing traffic conditions. Alternatively, we can reduce the volume of update traffic. This is commonly achieved by employing topology aggregation, which organizes the network into domains. Detailed routing information is delivered only inside each domain, and only aggregated routing information is transmitted across domain boundaries. Although such aggregation greatly reduces the routing protocol overhead, it introduces inaccuracy which typically has a negative impact on QoS routing performance [1].

The choice of topology aggregation method represents a trade-off between accuracy and compactness. The most accurate information is offered by the *Full-Mesh* method, but the amount of information to be advertised increases as the square of the number of border nodes. The greatest reduction of advertised information is offered by the *Symmetric-Node* method, but it does not adequately reflect any asymmetric topology information or capture any multiple connectivity in the original topology [2]. Such inaccuracy increases with the number of aggregation levels.

In this context, we propose a new hierarchical routing protocol, called *Macro-routing*, which implements the routing computation in a highly parallel and distributed fashion, by using mobile agents. The protocol has the following benefits.

• No routing information need be disseminated through the network as the mobile agents consult it *in situ*. Since the resulting protocol is not constrained by the amount of state information to be broadcast, the *Full-Mesh* representation can be used in the interest of efficient routing and network resource reservation.

• Since the routes within each domain are computed in parallel by multiple mobile agents, while this process and the aggregation process (described later) are also carried out in parallel in all domains at the same hierarchicalaggregation level, the computation of the overall path is significantly accelerated.

• Multiple paths which satisfy the requirements are found in parallel.

• Complex QoS constraints can be employed for traffic flows with special requirements without significant overhead (unlike link-state methods, where the relevant metrics would be broadcast, even if processed only for a handful of routes).

However, high communications overhead may be expected when mobile agents are used for routing purposes. In this paper we address questions like: *"How much traffic will* Macro-routing *generate?"* and *"How can we limit the traffic?"*.

II. Related Work

Private Network-to-Network Protocol (PNNI) [3] is the only QoS aware hierarchical routing protocol standardized and implemented. It is used in ATM networks and allows up to 104 hierarchical levels. A drawback of PNNI is that the route computation load is distributed unevenly among the network nodes. Also, the aggregation process used in PNNI leads to inaccurate state information advertisements [1], [2] which can result in the inefficient utilization of network resources.

A number of research projects propose other hierarchical routing strategies. These include the Hierarchical Distribution Protocol (HDP) and the Viewserver architecture.

HDP [4] is a proposal for a hierarchical routing protocol within Multiprotocol Label Switching (MPLS) networks. It uses cluster-based server farms as managing nodes which collect all the routing information from their domains and compute them centrally. Its main advantage of is that by computing the routes within different domains on one level in parallel, the setup time of a path is reduced. This is at the expense of an increase in the number of messages [4]. Also, by starting the path computation at the top of the hierarchy and progressing downwards, the aggregation strategies may be used inefficiently as some routing information can be already obsolete by the time the protocol reaches the lower levels of the hierarchy.

Viewserver [5] is an inter-domain routing protocol which seeks to overcome the problem of inaccurate routing information caused by the aggregation process. Therefore, the path

computation is done by the source node, which gathers centrally all the required routing information by traversing the hierarchy upwards (to find the parent "*view server*") and downwards (to collect detailed routing information about transit and destination domains). However, the setup time is long as the whole path is computed on a single node, while the amount of state information gathered for an end-to-end path may give rise to scalability issues for very large networks.

III. MACRO-ROUTING

We call our protocol *Macro-routing* because, being an inter-domain routing protocol, its routing decisions at the higher levels are macro-decisions, as opposed to the detailed or micro-decisions made at the lowest level of the hierarchy.

Instead of advertising state information, small mobile agents are dispatched to process such information at each node. The information used to compute routes can be much more detailed that in traditional link-state protocols (e.g., featuring multiple QoS constraints, or a Full-Mesh aggregated representation). Moreover, by using mobile agents which can replicate at each node and therefore analyze a large set of paths, route computations are done in a distributed and parallel manner which reduces the time required for path setup and distributes the processing burden amongst multiple mobile agents.

Topology aggregation

The hierarchical organization of *Macro-routing* consists at the lowest level of a number of domains which are typically independent administrative areas. The nodes within such domains are physical network nodes (i.e. router or switches). Each domain has a *managing node*, which must support mobile agents. It can either be selected from the nodes of the domain (as with PNNI) or it can be a distinct node (as in HDP). Its main function is to maintain an aggregated representation of the domain it is managing.

As the hierarchy is decided administratively, each domain at the lowest level of the hierarchy may choose its own routing strategy, i.e. it may use standard link-state methods or use mobile agents for route discovery. The latter method implies the existence of a mobile agent interpreter on each router or switch. The only *Macro-routing* requirement is that the managing node must contain the aggregation representation of the managed domain. The maintenance of that aggregate representation is the responsibility of the domain administrator.

For the higher levels of the hierarchy the managing node creates an aggregated representation in four steps:

1. Each border node (including the source and destination nodes) activates a small mobile agent that floods the domain by replicating itself at each node. Its goal is to find all possible paths to all the other border nodes. Each mobile agent records the path it follows and processes the routing information at each node. If one mobile agent is revisiting a node, or the path it has traversed to date does not satisfy the given QoS constraints, it will be discarded. If it reaches another border node it will transmit the path used and its cost to the managing node.

2. The managing node chooses one optimal path between each pair of border nodes. In this paper we use only additive path cost constraints (e.g., total delay), because it simplifies our discussion. However, the selection can be based on any QoS constraint.

3. A Full-Mesh aggregation topology is created using the selected paths. The costs of the selected paths will become *nodal costs* when computing paths at the next level of the hierarchy.

4. Some or all of the other computed paths, which have not been selected for the Full-Mesh representation, can be cached for recovery purposes or as alternative paths.

There are three major phases in the *Macro-routing* protocol whereby it finds and selects a QoS path from a given source to a given destination.

A. Determination of participant domains

The first phase consists in determining the domains through which the path is likely to pass. It develops in two stages.

In the first stage, the source node initiates a "*upwards search*" in the hierarchy for the lowest level **parent node** which has a view of both source and destination, as in HDP and Viewserver.

In the second stage, the **parent node** initiates a "*downwards search*" in parallel to all its children. Recursively, the nodes reached will continue the search to all their children until they reach the lowest level of the hierarchy. All the physical domains reached by this search will be ***participant domains***.

B. Path computation

The second phase consists in the actual finding of the requested path. This can be done either on-line or off-line.

For the on-line path computation, every managing node of the ***participant domains*** will create its own aggregate representation by calculating routes between all domain border nodes. Starting from the second level of the hierarchy, *nodal costs* will be considered as well as link costs when computing the path cost. The topmost domain will have as border nodes only the aggregated representation of the source and destination domains. The managing node of this domain will determine all the possible paths between its border nodes (the source and the destination) and based on their costs it can determine the optimal path. The other paths found during this process can be used by fast recovery mechanisms or as alternative paths.

For off-line path computation all the aggregated representations can be determined off-line and only updated ifwhen necessary.

C. Path reservation and set-up

To accommodate the traffic for which the request has been made, the final path must be set up and the resources reserved. The overall path can be determined by traversing the hierarchy downwards and interrogating all the managing nodes along the chosen path about the detailed sub-paths across their domains.

The path set up and resource reservation can be done either by existing resource reservation protocols (such as the Resource reSerVation Protocol (RSVP)), or by using suitably

programmed mobile agents. Within an MPLS network, setting up a hierarchical path is straight-forward using the MPLS label stack capabilities, as it can be treated independently within every domain and every level of the hierarchy.

When all the resources have been successfully reserved and the overall path has been set up, the request is served and the traffic may flow. In the case of resource unavailability or linknode failure, alternative paths which are already computed can be used for a fast recovery.

IV. Implementation details

The *Macro-routing* protocol can be implemented using any mobile agent technology. We have chosen the Wave technology, which we briefly describe below.

The Wave technology [6] is based on parallel spreading of recursive program code (or *waves*) in computer networks, accompanied by dynamic creation of virtual Knowledge Networks (KNs). Such networks can persist and reflect any declarative or procedural information. Moreover, they may become active and capable of self-evolution, self-organization and self-recovery. Other *waves* can navigate, control and modify KNs. All these actions are performed without a central memory or a centralized control. Another important Wave feature is that routines such as synchronization, message passing and garbage collection are implemented within the Wave Interpreter which resides on physical nodes rather than being implemented within mobile agents as with other mobile agent technologies. This and its syntax make Wave code very compact, perhaps 20 to 50 times shorter that equivalent programs written in CC++ or Java [7].

The use of Wave in MPLS networks for routing purposes has already been advocated in [8] to discover multi-point to point trees. Here it is used for hierarchical routing.

To implement our protocol, we used a Solaris-Unix implementation of the Wave Interpreter, written in C, which is available for public download on the Internet [9].

We describe below only the implementation of the *Path computation* phase of the protocol. The other two phases: *Determination of participant domains* and *Path reservation and set-up* have a straightforward implementation.

The *Path computation* phase starts from the lowest level of the hierarchy and sequentially progresses through the hierarchical levels until it reaches the root (the lowest level parent of both source and destination). Within a single level of the hierarchy, our protocol evolves in parallel within all domains. Each managing node initiates a search from each border node of its domain to all other border nodes. We will present in detail the implementation of such a "wave-based path search" later in the paper.

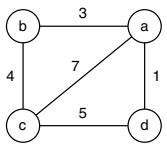
The Wave approach to mobile intelligence requires us first to construct a virtual *Knowledge Network* (KN) which is an abstract view of the physical network assembled by the wave agents as they probe its links and nodes. A KN ([6]) can be created using very simple code as in Fig. 1.

The algorithm depicted in Fig. 2 can be applied for the path search within first-level domains. The nodal costs at this level are zero as no aggregation has been performed yet. The corresponding Wave implementation is depicted in Fig. 3. Since the Wave implementation is very compact, the *waves* used are very small.

At the next level of the hierarchy we must consider non-zero nodal costs before we do a *local broadcast to all neighboring nodes*. Specifically, since the nodal cost depend on the outgoing link, we must (to use *wave* terminology) make an *explicit jump* to a specific next-hop node, rather than a *broadcast jump* (to all downstream nodes), adjusting the nodal cost as appropriate. The extension of our algorithm for this operation is in Fig. 4.

V. Macro-routing performance and test results

We compared our hierarchical routing protocol with HDP which is a protocol designed for the same context as our own, i.e., MPLS networks. For the example network depicted in Fig. 5, the path chosen by HDP has a cost of 85, while our protocol found a path with a cost of 59. The HDP approach to topology aggregation resulted in a suboptimal path being chosen.




```
CR(@#a.Fa=A.7#b.\
   4#c.Fc=A.5#d.\
   2#Fa.1#Fc)
```

(b)

Fig. 1. The Wave implementation for creating a four-node KN

```
BorderNodes=" ...."
for every NODE in BorderNodes{ start netspan (NODE) }
netspan (NODE) {
 BORDER_NODE=BorderNodes-{NODE}
 direct hop to NODE
 Fcost = 0
 repeat{  within each current_node
  if ((the node wasn't visited yet) and
  (the link just visited satisfies the requirements)) {
    add the address of the current_node to Fapth
    update the current path cost in the Fcost
    if (an BORDER_NODE was reached) {
      send Fapth and Fcost to the managing node
      TERMINATE }
    else local broadcast to all neighboring nodes }
  else TERMINATE }}
```

Fig. 2. The path computation algorithm (for zero nodal costs).

```
Falg=RP(OS(Fcost==0,A~Fborders).\
        A~Fpath.Fcost<Frequired.Fpath&A.Fcost+L.♯).\
Fborders=ARG1.Frequired=ARG2.Fmanaging=ARG3.\
@♯Fborders.Fcost=0.\
Falg.\
Fpath&A.Fcost+L.@♯Fmanaging.Npath&Fpath.Ncost&Fcost
```

Fig. 3. The Wave implementation of Fig. 2.

```
else { local broadcast to all neighboring nodes
  Neighbor_nodes=" ...."
  for (every NEXT-NODE in Neighbor_nodes){
      determine the input-output port pair
      update the current path cost in the Fcost
      local jump to the NEXT-NODE }
}
```

Fig. 4. Modification of the path computation algorithm of Fig. 2 to account for non-zero nodal costs.

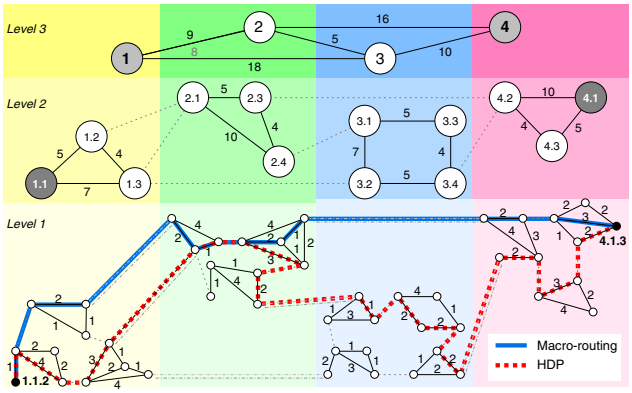


Fig. 5. Macro-routing versus Hierarchical Distribution Protocol

Macro-routing rapidly finds the optimal path using path computations executed in parallel not only in different domains at the same hierarchical level but also between any two border nodes of a domain. However, this parallel operation can result in a large number of *waves* traversing the network. Simulations and analytical models are used below to investigate this overhead, which, if excessive, would render Macro-routing impractical.

To determine the amount of traffic generated by the protocol we have to determine both the size and the number of mobile agents (*waves*) involved in the routing. The size of a mobile agent (in bits) depends on the mobile agent technology used but, unless excessive, is not likely to limit the protocol's scalability. The *waves* generated by the Macro-routing protocol vary between 200 and 300 bytes in size. For counting the number of mobile agents we can either simulate the protocol operation over different topologies (such as Fig. 6-a) or we can develop a mathematical formula. Both approaches are considers below. For both cases we considered the scenario where all paths between source and destination satisfy the QoS constraints, since this gives rise to the most *waves*.

A. The mathematical model

We consider only the *waves* generated by a single border node (see Fig. 6-b). The total number of mobile agents generated within one domain can be determined by multiplying the number of mobile agents generated by one border node by the number of border nodes.

We seek a mathematical formula which yields the number of *waves* which traverse $n+1$ nodes based on the number of *waves* which traverse n nodes. Such a formula is described by Markov Chain branching processes (also known as Galton-Watson processes): $X_{n+1} = \sum_{X_n}^{k=1} Z_n^{(k)}$, where $Z_n^{(k)}$ is the number of *waves* generated by node k.

We assume that $Z_n^{(k)} = 0$ if the current *wave* revisits the node k (there is a cycle) or if node k is a border node (a path has been found) and that $Z_n^{(k)} = \alpha$ otherwise, where α is the average node degree. Hence we may write: $X_{n+1} = (X_n - C_n - B_n) * \alpha$, where C_n is the number of *waves* which end up in cycles and B_n is the number of *waves* which found a border node. Moreover, if we consider: $C_n = X_n * p_{cn}$ and $B_n = X_n * p_{bn}$, where p_{cn} and p_{bn} are the respective probabilities of a *wave* ending up in a cycle, and of finding a border node, we obtain: $X_{n+1} = X_n * (1 - p_{cn} - p_{bn}) * \alpha$. This formula gives us the number of *waves* generated by one border node. The values of the parameters p_{cn} and p_{bn} may be estimated by simulation.

B. Simulation results

We used the *Georgia Tech Internetwork Topology Models* (GT-ITM) [10] to generate random network topologies by varying the number of nodes N and the connectivity degree $2L/N$. For each topology we counted the number of *waves* at each stage n (i.e. *waves* that already visited n nodes). We considered the number of: **alive waves** - waves that are still alive and continue to evolve in the network; **cycle waves** - waves which end up in a cycle; **total waves** - the total number of waves within the one stage; **border waves** - waves which reached a border node. Using this results we then calculated the *cycle probability* and the *effort for obtaining long paths*(i.e, the ratio of the number of useless *waves* which end up in cycles to the number of productive *waves* which find a path).

Fig. 7 presents the number of *waves* present within a network of $N = 12$ nodes and $L = 20$ links as a function of the path length. The total number of *waves* increases with the number of nodes visited until the path length is nine. Thereafter the high probability of *waves* ending in a cycle or finding a border node causes the total number of *waves* to decrease. A path length of thirteen results in a *wave* population of zero, since the probability of a *wave* ending up in cycle is unity (all twelve nodes having been visited).

The results for the *cycle probability* and the *effort for obtaining long paths* for networks with $N = 12$ nodes and a number of links ranging from $L = 11$ links (the minimal level of connectivity: $L = N - 1$) to $L = 66$ links (full mesh connectivity: $L = N(N-1)/2$), can be seen in Fig.s 8-9.

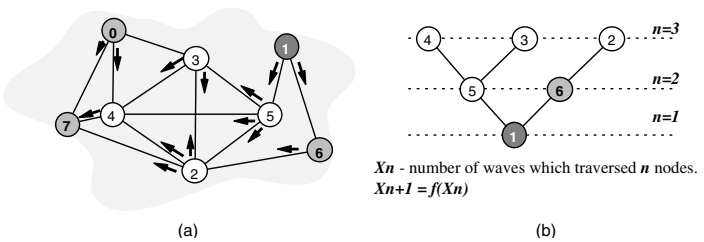


Fig. 6. Wave counting tests by (a) experiments or (b) mathematical model

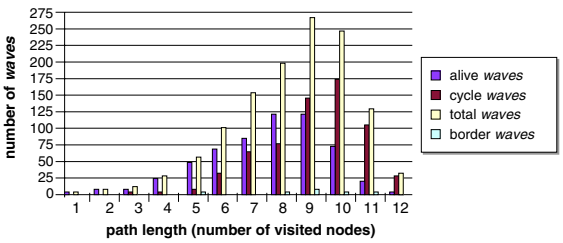


Fig. 7. The number of waves propagating from a border node.

The results in Fig. 8 closely match the function $F(x) = (x < 3)?0 : (x-3)/(N-3)$ (where x is the path length and $N = 12$), which estimates the cycle probability as the number of possible cases $(x-3)$ (nodes already visited) divided by the number of total cases $(n-3)$ (the total number of nodes). The number 3 appears because cycles can occur only if more than 3 nodes have been visited as a *wave* will not return through the link from which it came.

Fig. 9 shows that the effort involved in finding long paths is excessive. Hence the protocol must be modified to ensure its scalability.

Limiting the population of mobile agents

We introduce a parameter called *lifespan* to our algorithm which resembles the TTL field used in the IP protocol. Its purpose to to limit the number of *waves* generated during route set-up by limiting the number of generations which the parent *wave* can produce. The rationale for this is that the law of diminishing returns is assumed to apply - it is unlikely that an exhaustive search of every possible path is necessary to find the optimal path. The modified algorithm is no longer guaranteed to find the optimal path (and indeed may find no path if the destination is more than *lifespan* hops away).

The influence of the *lifespan* parameter on protocol performance was investigated by simulation. A hierarchical network with two levels was chosen. Each level contains 9 node domains with the average node degree 3.5 (i.e $2*L/N = 3.5$). Therefore the overall network contained $9 * 9 = 81$ nodes.

The efficiency is the ideal path cost divided by the cost of the path obtained by the (sub-optimal) lifespan-limited algorithm. Fig. 10 shows that the performance of the latter rapidly approaches optimality as the lifespan increases.

Determining the ideal choice of this parameter will require further study, but is clearly related to the network diameter. Longer paths will be appropriate in a sparse topology, where we want to exhaustively search the few routes available, while a shorter lifespan will be appropriate for highly connected networks (where otherwise *waves* will proliferate).

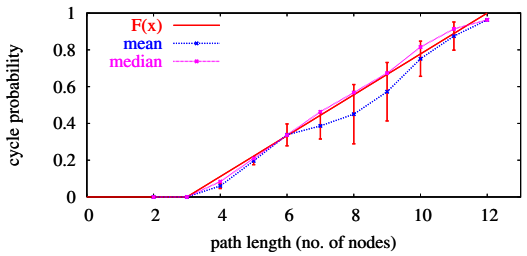


Fig. 8. The *cycle probability* (=cycle waves/total waves)

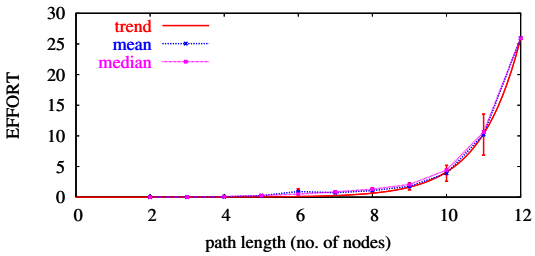


Fig. 9. The *effort for obtaining long paths* (=cycle waves/alive waves)

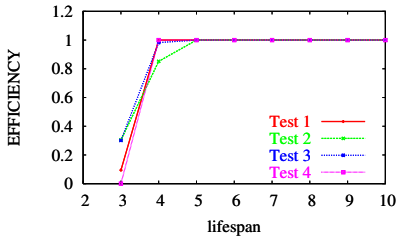


Fig. 10. Macro-routing's performance while applying the *lifespan*

VI. Conclusions

We have described Macro-routing, a new approach to hierarchical routing for MPLS networks. By using mobile agents, our approach allows routes to be discovered rapidly without the imprecision introduced by topological state aggregation in other approaches. The price paid for this level of performance is that a large number of mobile agents (implemented as *waves*) traverse the network. However, the level of wave traffic can be restricted by limiting their *lifespan* without significantly impairing the algorithm performance.

The algorithm is of particular use for applications which require reliable transmission. Since the waves return results not just for the best path but all feasible paths, communication can be quickly re-established in the event of linknode failure. Furthermore, disseminating data across multiple paths to the destination may be easily accomplished.

Future work will investigate algorithms for the optimal choice of the lifespan parameter, and the use of *Macro-routing* to efficiently implement constraint-based routing.

References

[1] R. Guerin and A. Orda. QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms. In *IEEE Conference on Computer Communications*, pages 75–83, April 1997.

[2] W. Lee. Topology Aggregation for Hierarchical Routing in ATM Networks. *ACM Computer Communication Review*, 25(2):82–92, 1995.

[3] ATM Forum. Private network-network interface specification , version 1.0. Technical Report af-pnni-0055.000, ATM Forum, March 1996.

[4] M. El-Darieby, D. Petriu, and J. Rolia. A Hierarchical Distributed Protocol for MPLS path creation. In *7th IEEE International Symposium on Computers and Communications*, pages 920–926, July 2002.

[5] C. Alaettinoglu and A. Shankar. The Viewserver Hierarchy for Interdomain Routing: Protocols and Evaluation. *IEEE Journal of Selected Areas in Communications*, 13(8):1396–1410, 1995.

[6] P. Sapaty. *Mobile Processing in distributed and Open Environments*. Wiley, 2000.

[7] S. Vuong and I. Ivanov. Mobile Intelligent Agent Systems: WAVE vs. JAVA. In *1st Annual Conference of Emerging Technologies and Applications in Communications*, May 1996.

[8] S. Gonzalez-Valenzuela and V. Leung. QoS Routing for MPLS Networks Employing Mobile Agents. *IEEE Network Magazine*, 16(2):16–21, 2002.

[9] The wave page. http://www-zorn.ira.uka.de/wave/wave.html.

[10] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602, 1996.