Active Routing and Forwarding in Active IP Networks*

Jun-Cheol Park

Department of Computer Engineering, Hongik University, 121-791 Seoul, Korea
jcpark@cs.hongik.ac.kr

Abstract. In this paper, we address the problem of active packet routing and forwarding in an active IP network involving both legacy IP routers and active capable routers. We first argue that any traditional routing scheme does not work well in active networks. We then give a formulation of a generalized active routing problem and propose an efficient routing scheme based on the formulation. The scheme is active-destination initiated, and is used for computing paths to a set of active destinations(servers). This selective approach allows the scheme to be seamlessly deployed onto any active IP network. We also present a labeling technique such that a label embedded in a packet can uniquely identify the expected ongoing services on the packet's journey. With a forwarding table indexed by these labels, a simple and quick label lookup is sufficient to resolve any incoming active packet's next hop.

1 Introduction

Routers in the Internet pass any incoming packet without any examination or modification on the packet's payload. They process and modify the packet's header only. A packet in an active network can carry fragments of program code to be executed on the network nodes as well as data to be applied to the code. Thus network users can customize network behavior to suit their requirements and needs by embedding application-specific computation in the packet's code. As a result, it is possible for the users to change the behavior of network nodes on the fly. This ability allows network service providers to develop and deploy new services quickly without going through standardization or vendor-driven consensus.

For a while, results from active network research [1,2,3,4,5,6,7,8,9] will be tested as add-on features onto the existing network infrastructure before active networking is believed to be mature enough to be deployed in a global scale. At any rate, it seems unreasonable to simultaneously convert the whole Internet to a gigantic active network. In the meantime, only some of the routers in the network will support programmability. An active IP network [18], an intermediate step toward pure active networking, involves two different types of network nodes, legacy IP routers and active routers with active processing capability. Both nodes

* This work was supported by 2003 Hongik University Research Fund.

N. Wakamiya et al. (Eds.): IWAN 2003, LNCS 2982, pp. 58–67, 2004.

have been integrated into a single network, and take advantage of the same type of packets and node resources. As a result, users of the traditional IP service do not need to be aware of the existence of active routers. This allows active routers to be deployed seamlessly into an IP network.

In this paper, we address the problem of packet routing and forwarding in an active IP network. In the Internet, each IP packet's next hop is solely based on the packet's destination address. On the other hand, the route of an active packet has to include appropriate intermediate active processing node(s) in between the packet's source and destination. In addition, the current route of an active packet to its destination is subject to change to reflect any further constraint arising later at another intermediate active router. Thus the routing for active packets here is *dynamic* and differs from any Internet routing currently in use. Our goal is to design an efficient active routing scheme and provide a way to deploy the scheme onto active IP networks. The proposed scheme is active destination(server)-initiated: the destination of active application initiates a constrained path computation to itself using dynamic metrics reflecting active processing costs as well as link costs. We also propose a compact labeling technique so that a label embedded in a packet can uniquely identify the expected ongoing services on the packet's journey. The routing scheme constructs a forwarding table arranged by such labels at each active router. Then a simple table lookup suffices for an incoming active packet to determine the next active router to its destination.

The organization of the paper is as follows. Section 2 presents a graph-based formulation of the problem of routing in an active IP network. In section 3, we provide an active destination-initiated routing scheme that constructs a forwarding table indexed with labels at each active node. In section 4, we then show how a simple table lookup with a fixed length label can quickly determine an active packet's next active router. We provide related work of routing in active networks in section 5. In section 6, we conclude the work in the paper and present some future research problems.

2 Routing in Active IP Networks: Problem Formulation

Before we present the problem formulation, we need to resolve a couple of design issues. First, how to address active routers in an active IP network? Since we expect active routers in the network to coexist with other legacy IP routers, we stipulate that each active router is identified by its IP addresses like other IP routers. Second, how to differentiate active packets from ordinary IP packets? We use the Router Alert Option [11] to identify active packets. It allows an active node to process active packets and IP packets differently. An active packet does not cause any alarm when processed by non-active IP nodes and undergoes default IP processing.

The major constraint in active routing is that every active route must include at least one intermediate active node that performs some customized computation within the network. A general form of active routing is to select a sequence

of intermediate active nodes through which the active hosts are connected. For example, as observed in [12], an active application for secure data transmission may include both encryption and decryption active processing in that order.

Basically, an active flow can choose its own route on the fly at each intermediate active node in the network. Thus we can't safely assume that the path of an active packet computed at a certain active node remains unchanged throughout the packet's lifetime. For example, if an application can decide whether or not its data be compressed based on the current condition of the network possibly for saving bandwidth of a congested link, the packets of the application may or may not need decompression afterward. Hence this routing problem requires a dynamic routing scheme that should be able to reflect a new constraint, if ever come, on the fly.

We are now ready to formally state a routing problem in an active IP network. The formulation given here is similar to that of the work in [12]. The network is represented by a graph, $G = (V, E)$, in which the nodes correspond to routers and hosts, while the edges correspond to physical links between these nodes. We have a source $s \in V$ and a destination $t \in V$. An edge (u, v) has a non-negative value $cost(u, v)$ representing the "cost" of sending a packet across the corresponding link. Also, for $1 \leq i \leq k$, we let $A_i \subseteq V$ be a subset of the nodes and it contains nodes where the active service i may be performed. We assume that the destination t belongs to A_j for some j. For each node $w \in A_i$, a non-negative value $cost(w)$ representing the cost of performing active service i at the node w is associated. On its journey to the destination, each active packet p processed at an active node $w \in A_i$ has an associated *service list*, denoted $list(p, w)$, which is a sequence $(s_a, \ldots, s_b)$, where $1 \leq s_a, \ldots, s_b \leq k$, representing the active services p need to have from now onward upto the destination. A service list $list(p, w) = (s_a, \ldots, s_b)$ is *dynamic* so that it can be updated to $list(p, w')$ later, where for some $w' \in V_{s'}$, $s' \in \{s_a, \ldots, s_b\}$, $list(p, w')$ is not necessarily a proper suffix of $list(p, w)$.

Throughout this paper, for simplicity, we assume each active node performs only one active service. It is straightforward to extend our work to take care the case when an active node is able to select an arbitrary service among multiple different active services available at the node.

For an active packet, the purpose of our routing algorithm is to find a feasible path that keeps to its current active service constraint at every active node and has "least" cost. The cost of a path is defined as the sum of the costs of its edges and active processing nodes. A least cost path in a legacy IP network is not necessarily the one that our scheme is looking for since the path is obtained with no awareness of service constraint. Given a destination, moreover, the least cost feasible path of a packet at a node is not necessarily a proper suffix of the least cost feasible path of the same packet computed before. Since a packet's processing result in terms of routing is unknown until it is processed, no routing algorithm can always generate global optimum routes in terms of path costs.

A solution was proposed in [12] for finding a least cost feasible path for the special case that the service list constraint is known in advance and remains

fixed. We call this routing *static* in the sense that the service list for an active packet remains unchanged along the path to the packet's destination. In the general problem with dynamic nature, it is possible for a "good" route selected at a certain point on the path to be revoked and recomputed at later active nodes in the network.

3 Our Routing Scheme

3.1 Basic Idea

Our routing scheme governs route computation only for the set of active destinations(servers) in an active IP network. For non-active destinations in the network, a popular link state routing algorithm like OSPF2 [16,17] can do the route calculation. Because, in our routing scheme, each initiating active destination(server) takes charge of its route recomputations, the updating of routes to each active destination is uncorrelated. Moreover our active routing is independent of any traditional routing that computes paths for other IP nodes. Hence the scheme can control its routing overhead itself and do not necessarily have to be affected by the dynamic conditions in the active IP network.

A basic assumption in our approach is that each active node has a complete, global knowledge about node connectivity, link costs, and every other active node's information including its identity and providing active service. A modification to the link state protocol OSPF2 can provide the necessary capability. For supporting multicast, OSPF2's link state advertisement packet is marked by a particular bit in its option field. Likewise, an active router may use an unused bit in the option field to indicate its active capability when transmitting its own link state advertisement packets. Non-active nodes would simply ignore the active bit as they are not aware of the meaning of the bit, while active nodes would recognize the active bit and record the identity and location of the packet's sender. In this way, each active node can take active nodes apart from non-active nodes in the network while processing its incoming link state advertisement packets. The scheme can be deployed to any active IP network without modifying OSPF2 software on any legacy IP router.

From an active IP network given as a graph, we generate the *logical* network by extracting only the active nodes and connecting two active nodes with an edge if and when there exists either a direct edge or an indirect path of IP nodes between them. We assume that a link state routing algorithm is used throughout the network for computing paths and generating routing table at each router. A link state routing algorithm like OSPF2 computes the least-cost path between a source and destination using complete, global knowledge about the network. To obtain this information, each node broadcasts the costs of its attached links to all other routers in the network. As the result of the nodes' link state broadcast, all nodes have an identical and complete view of the network, from which each node can then compute the same set of least-cost paths as every other node. Obviously the computation here is for computing paths of IP packets only.

To take care active packet routing, each active node has to exchange information about the current active processing cost of its service with every other active node. The information exchange is done via the imposed logical topology of active nodes overlaying the overall network topology. Thus each active node in the network needs to run the routing algorithm for active packets with service constraints as well as the link state routing algorithm for IP packets. Here we discuss our routing scheme in terms of the logical network of a given active IP network.

3.2 Routing Message Exchange and Forwarding Table Construction

Our routing scheme is destination-initiated, which means each active destination initiates a constrained path computation to itself. The basic mechanism is message flooding in the logical network. An initial routing message is generated at a target destination node periodically. Upon the receipt of a routing message, each node updates its current best path to the destination based on the information carried by the message, modifies the message, and then floods the modified message to all neighbors except the one from which the original message was received.

Each routing message contains a TTL value to prevent it from appearing infinitely and an increasing sequence number to ignore old routing messages generated in the previous rounds. A node may receive more than one routing message with the same sequence number, indicating distinct paths to the destination. A routing message also contains the XOR's over the active services the message has (backward)traversed since its initiation at the destination node. This value becomes a label in the active forwarding table of the arriving node. The forwarding table at a node has a set of entries each represented by a tuple $< label, next\ hop(active\ node), cost, seq\ number >$, where *label* is the index key value for table lookup, *next hop* is the next hop on the current best route with the corresponding active list, *cost* is the cost of the route from the node, and *seq number* is used for considering only up-to-date routing messages. Labeling and table lookup methods are discussed in detail in the next section.

Let t be the active destination initiating the computation to its active service s_x. An initial routing message from t at an interval is of the form $[t, s_x, c, seq, ttl]$, where t is the originating node's address, s_x is an advertising service list, c, initially $cost(t)$, is the cost to reach the destination via the services contributed in s_x. Pseudocode is given in Figure 1 for the processing at an arbitrary active node a when it receives a routing message $[b, s_y, c, seq, ttl]$. First, we update the cost c to c' to incorporate the active processing cost at a and the link cost crossing the link (a, b). Then, to insert/update a tuple with the key value s_y, we lookup a's forwarding table. Consider the two cases. Case1: A matching entry is found. We investigate the sequence number *seq* of the message to see whether it is stale or not. If the message is not stale, then we compare the newly computed cost c' to the one currently stored in the table. In case c' is smaller(i.e., shorter), we replace the stored tuple in the table with the tuple newly having the cost(c'), next hop(b), and sequence number(seq). Then we flood an updated message

carrying the new cost, the new label with the active service provided at a is added via XOR, and a decremented TTL. In case the currently stored cost is not greater than c', we simply ignore and discard the message. If the message is stale, we ignore and discard the message as well. Case2: A matching entry is not found. We need to insert the tuple reflecting the message content into the forwarding table, and then flood an updated message similar to the one in Case1.

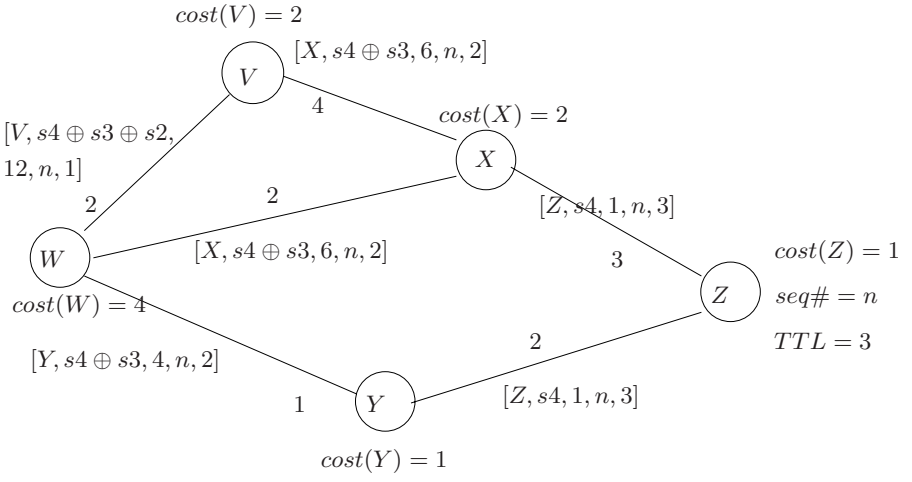
```
let c' ← c + cost(a) + cost(a, b);
lookup a's forwarding table entry with the key s_y;
 if matching entry found
     let < l, nh_cur, c_cur, no > be the tuple with the key s_y = l;
   if (no ≤ seq)
     if (c_cur > c')
       replace the tuple < l, nh_cur, c_cur, no > with < l, b, c', seq >;
       forward [a, s_y ⊕ i, c', seq, ttl − 1] to every neighbor of a but b;
         //i is the active service provided at a and
         //⊕ is the XOR operator
     else // c_cur ≤ c' : no better route found
       ignore and discard the message;
   else // no > seq : obsolete message
     ignore and discard the message;
 if matching entry not-found
     insert the tuple < s_y, b, c', seq > to the forwarding table;
     forward [a, s_y ⊕ i, c', seq, ttl − 1] to every neighbor of a but b;
```

Fig. 1. Processing of Routing Message $[b, s_y, c, seq, ttl]$ for Constructing a Forwarding Table at the Node a

As an example, let us consider the propagation of routing messages in the simple logical network shown in Figure 2. In this example, assume that the message forwarded by node X reaches node W before the one forwarded by node Y. Then the forwarding table entry at node W is updated with the new value representing a better route to the destination with the same service list.

3.3 Active Routing Metric

To measure the dynamic active processing cost at an active node, we use the delay measurement for representing node utilization. To do so, each incoming active packet is recorded with its time of arrival at the node and its departure time from the node. Note that the departure time minus the arrival time represents the amount of time the packet was delayed in the node. This number is averaged with the last reported cost to suppress sudden change. The range of value should be carefully adjusted to leverage the processing cost of a plain IP packet.



[W's forwarding Table]

label	*nexthop*	*cost*	*seq#*	
~~$s4 \oplus s3$~~	~~X~~	~~12~~	~~n~~	*: line overwritten
$s4 \oplus s3$	Y	9	n	
$s4 \oplus s3 \oplus s2$	V	18	n	

Fig. 2. A Forwarding Table Construction at W when Z initiates the message flooding

4 Labeling and Active Forwarding Table Lookup

The basic idea of labeling is to associate each path with a label computed as the XOR over the services done along the path, and then use this label for each packet of an active application (or flow) to follow that path. Here we assume that each service is uniquely identified among the network with a fixed sized identifier. A path is identified with its service list, say, $(s_1, s_2, \ldots, s_m)$, where s_m is the service available at the destination of a packet following the path. At a certain active node a, where $a \in A_{s_i}$ for some i, the service list $(s_{i+1}, \ldots, s_m)$ is encoded by the label $l = s_{i+1} \oplus \cdots \oplus s_m$. It should be obvious that forwarding tables constructed by the above routing scheme are consistent. The construction ensures that after a packet carrying a label l' generated from its expected service list is processed by service s_i at node a, the forwarding table at a must contain an entry with label $s_i \oplus l'$. Notice that l' should be $s_i \oplus s_{i+1} \oplus \cdots \oplus s_m$ and $s_i \oplus l' = s_i \oplus (s_i \oplus s_{i+1} \oplus \cdots \oplus s_m) = s_{i+1} \oplus \cdots \oplus s_m = l$. It is easy to see that the packet would follow exactly along the path that is linked by the next hop fields of participating nodes.

Active forwarding table lookup is a simple and efficient operation of finding a matching identifier. An active packet carries a short, fixed length label by taking XOR over the service identifiers along the prospect path. To determine the next hop of any incoming active packet, each active node refers its own forwarding table with the packet's label l'. Below we summarize how the forwarding decision is made at an active node a when it receives a packet p with the label l'.

```
the node a processes the packet p with the label l' with its service,
  say s_i;
let (s_{i+1}, ..., s_m) be the packet p's onward service list revealed at the
  above step;
let l denote s_{i+1} ⊕ ··· ⊕ s_m;
lookup the forwarding table at the node a with the computed label l;
  if matching entry found
    forward the packet p carrying the updated label l to the next hop
      found;
  else // unexpected service list
    compute the packet p's route on the fly;
```

Fig. 3. Forwarding Decision at the Node a for the Packet p with the Label l'

We apply the IP tunneling technique for ensuring active packets sent by an active node to correctly reach to another active node connected to the sending node by intervening IP routers. The receiving side active node, which is the destination of this IP packet, determines that it contains an active packet, and extracts the active packet for customized active processing. In this way, it is certain that the forwarding route of an active packet is always as expected, at least in terms of the correct destination, by the most recently visited active node.

5 Related Work

Maxemchuk and Low [14] discussed how active routing can extend the capabilities and utilization of various routing paradigms such as label switching, QoS routing, mobility, etc. They proposed some implementation techniques including pricing and sandboxes. Their techniques are mainly for enforcing and encouraging economic network resources. Our work has a different goal. We want to provide a way to efficiently deliver active packets with active list constraints in an active IP network. The work in [12] proposed an approach to the problem of configuring application sessions that require intermediate processing. The approach transforms the session configuration problem into a conventional graph theory problem. However, the proposed algorithms assumed that the active service constraint of a packet should remain fixed throughout the packet's lifetime, which limits the applicability. Also, it is unclear how the proposed algorithms

can be realized when many of the nodes in the network are non-active. On the contrary, our routing scheme is able to coexist with the popular link state routing algorithm and requires no modification of the behavior of legacy IP routers. The work in [10] might be used as the preprocessing step of our routing scheme in order to discover active neighbors in an active IP network. Unlike our assumption on the fixed locations of services in an active network, a work in [13] proposed a programmable selection and reconfiguration of service location. It would be interesting to see how our scheme could be extended to allow such a dynamic service provisioning. Our routing scheme is partly influenced by the work in [19], which proposes a destination-initiated route computation for a small number of hot destinations. The proposed routing scheme takes advantage of this selective routing idea by taking active routing apart from the traditional IP routing. Using labels for fast packet processing is not new to networking areas [15,20]. For active networks, the work in [15] uses labels for fast demultiplexing of active packets upto their active handlers. The purpose of labels in our scheme is, however, for efficient active path computation and forwarding, which are essential to active routing.

6 Conclusion

The major difference between active routing and traditional IP routing is that routing decision for an active packet depends upon not only the destination address of the packet and the routing table, but other factors including active processing result, policy, etc.

In this work, we addressed the problem of active packet routing in an active IP network, which involves both legacy IP routers and active routers. We formulated a generalized active routing problem in an active IP network and then proposed an efficient active routing scheme based on the formulation. Our scheme is active-destination initiated, and it is used only for the set of active servers without affecting the traditional IP routing for other destinations. This selective approach allows the proposed scheme to be seamlessly deployed onto any active IP network. Also, it reflects dynamic active processing costs metric in addition to link costs when computing routes to active destinations. In order for fast active packet forwarding, we provided a compact labeling technique such that a label embedded in a packet can uniquely identify the expected ongoing services on the packet's journey. With a forwarding table indexed by these labels, it is possible for an incoming active packet to determine its next hop active node by a simple and fast table lookup.

We believe that a large scale experimental study on the proposed scheme is necessary and beneficial to investigate any further questions and problems. As a first step, we are currently working on the extension of OSPF2 protocol and its deployment on a small scale active network testbed. An important problem that must be addressed would be the integration of the proposed routing scheme with existing exterior gateway protocol like BGP4 for a wide area active IP network.

References

1. D. Tennenhouse, and D. Wetherall, "Towards an active network architecture", Computer Communication Review, vol. 26, no. 2, 1996.
2. D. Alexander, W.A. Arbaugh, A.D. Keromytis, and J.M. Smith, "A secure active network environment architecture: realization in SwitchWare", IEEE Network, vol. 12, no. 3, 1998.
3. M. Hicks, P. Kakkar, J. Moore, C.A. Gunter, and S. Nettles , "PLAN: A packet language for active networks", Proc. ACM SIGPLAN Int'l Conf. on Functional Programming(ICFP'98), 1998.
4. B. Schwartz, A. Jackson, T. Strayer, W. Zhou, R. Rockwell, and C. Patridge, "Smart packets for active networks", Proc. IEEE Conf. on Open Architectures and Network Programming(OPENARCH'99), 1999.
5. D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: A toolkit for building and dynamically deploying network protocols", Proc. IEEE Conf. on Open Architectures and Network Programming(OPENARCH'99), 1999.
6. S. Bhattacharjee, K. Calvert, and E. Zegura, "An architecture for active networking", Proc. IEEE INFOCOM'97, 1997.
7. Y. Yemini, and S. da Silva, "Towards programmable networks", Proc. IFIP/IEEE Int'l Workshop on Distributed Systems, Operations, and Management, 1996.
8. D. Decasper, and B. Plattner, "DAN: Distributed code caching for active networks", Proc. IEEE INFOCOM'98, 1998.
9. J. Moore, M. Hicks, and S. Nettles, "Practical programmable packets", Proc. IEEE INFOCOM 2001, 2001.
10. S. Martin, and G. Leduc, "RADAR: Ring-based adaptive discovery of active neighbour routers", Proc. IFIP-TC6 4th Int'l Working Conf. on Active Networks(IWAN 2002), 2002.
11. D. Katz, "IP Router Alert Option", Internet Request for Comments 2113, 1997.
12. S. Choi, J. Turner, and T. Wolf, "Configuring sessions in programmable networks", Proc. IEEE INFOCOM 2001, 2001.
13. A.B. Kulkarni, G.J. Minden, V. Frost, and J. Evans, "Survivability of Active Networking Services", Proc. 1st Int'l Working Conf. on Active Networks(IWAN 1999), 1999.
14. N.F. Maxemchuk, and S.H. Low, "Active routing", IEEE Jr. on Selected Areas in Communications, vol. 29, no. 3, 2001.
15. T. Wolf, D. Decasper, and C. Tschudin, "Tags for high performance active networks", Proc. IEEE Conf. on Open Architectures and Network Programming(OPENARCH2000), 2000.
16. J. T. Moy, OSPF Anatomy of an Internet Routing Protocol, Addison-Wesley, 1998.
17. J. T. Moy, OSPF Complete Implementation, Addison-Wesley, 2001.
18. D. Murphy, Building an Active Node on the Internet, M.E. Thesis, MIT, 1997.
19. J. Chen, P. Druschel, and D. Subramanian, "A new approach to routing with dynamic metrics", Proc. IEEE INFOCOM 1999, 1999.
20. I. Stoica, and H. Zhang, "LIRA: An approach for service differentiation in the Internet", Proc. NOSSDAV'98, 1998.