Network Working Group
Request for Comments: DRAFT

Philip Almquist
Stanford University/BARRNet
July 1991

Ruminations on the Next Hop

Status of This Memo

status per RFC1111, to be filled in by the RFC Editor (This is an informational document; it does not propose a standard, although the author hopes that ideas contained herein will influence future standards).

This is the fourth draft version of this memo, dated July 23, 1991 3:09 am. A final version is (hopefully) forthcoming.

Distribution of this memo is unlimited.

Summary

This memo discusses how routers do, can, and should decide how to use the routing information available to them to route packets. It is particularly concerned with the problems of border routers and other routers which have to choose from among routes learned from multiple routing domains, often using different routing protocols.

Although many of the details of this paper are specific to the problem of forwarding Internet Protocol (IP) packets, I believe that the problems identified and the general approach to the solution would also be relevant to those working with CLNP or other connectionless Network Layer protocols.

Although this memo is not an official product of the Router Requirements Working Group, it is being published as an INTERNET DRAFT because it is required reading for those who wish to actively participate in the Router Requirements sessions during the IETF meeting in Atlanta.

Acknowledgments

This paper owes considerable debt to the IETF's Router Requirements Working Group, which started me to thinking about the issues in this paper and also reviewed several earlier drafts. I am particularly indebted to Ross Callon, who made me understand enough about area routing to understand why the algorithm in section 4.2 is inadequate, and also pointed out several errors in my description of Dual IS-IS.

Vint Cerf and Noel Chiappa provided additional useful comments.

Contents

1. Introduction 3
1.1. Overview of the Routing Function 3
1.2. The Problem 4
1.3. Outline of this Paper 4
2. Some Historical Perspective 5
3. Route Lookup Algorithms 6
3.1. Pruning Rules Used in This Paper 7
3.2. Example of Other Possible Pruning Rules 10
4. Some Route Lookup Algorithms 10
4.1. The Revised Classic Algorithm 10
4.2. The Router Requirements Algorithm 11
4.3. The Variant Router Requirements Algorithm 12
4.4. The OSPF Algorithm 13
4.5. The Dual IS-IS Algorithm 14
5. The Problem Revisited 15
5.1. The “Master Protocol” Approach 15
5.2. The “Rank Ordering of Routing Protocols” Approach 16
5.3. The “Multiple Virtual Router” Approach 16
5.4. The “Meta-lookup Algorithm” Approach 17
6. The Meta-Lookup Algorithm Approach in Detail 18
6.1. The Recommended Meta-Lookup Algorithm 18
6.2. The Recommended Policy Pruning Mechanism 19
6.2.1. Details of the Policy Pruning Step 20
6.2.2. Assignment of Preference Values 20
7. Performance and Implementation Issues 22
7.1. Serialization of the Recommended Algorithm 22
7.2. Precomputation of Routes 23
7.3. Route Cacheing 23
8. Conclusion 24
References 25
Security Considerations 27
Author’s Address 27

1. Introduction

The fundamental thing that a router does is to receive packets, examine their headers, and decide where to send them next. During the past year I have chaired several meetings at which a group of experts (the Internet Engineering Task Force's Router Requirements Working Group) attempted to reach consensus on a precise description of how routers should make this decision. Lengthy and often heated debate produced little concrete agreement. I was rather fascinated to discover that this seemingly simple process of deciding where to send the packet next is apparently not well understood, and in fact seems to hide some difficult open questions.

This paper attempts to carefully examine how a router should use the routing information available to it to decide how to route packets. A companion document, *Ruminations on Route Leaking* [1], discusses the related issue of how a router passes on routes available to it to other routers.

Except insection 7, this paper discusses conceptual models, not implementations. While efficient router implementation is an interesting and important topic in its own right, I have chosen in this paper not to obscure the concepts with implementation details.

1.1 Overview of the Routing Function

The fundamental purpose of a router is to switch packets. To do so, the router maintains a *forwarding information base (FIB)*, a conceptual database of all routes known to the router. In order to allow the router to choose from among the routes in the FIB, each route has various attributes associated with it.

In order to maintain its FIB, a router generally belongs to one or more *routing domains*. A routing domain is a collection of routers which coordinate their routing information using a routing protocol. Each router adds routes to and deletes routes from its FIB based on rules prescribed by the protocol. Most routers also allow the router's manager to manipulate the contents of the FIB through configuration or network management commands. For purposes of this paper, a router whose FIB contains routes added by such manual means can be thought of as being a member of its own (degenerate) routing domain as well as any routing domain(s) it may be a member of by virtue of its use of routing protocols.

When the router receives a packet to be forwarded, it examines attributes of the packet and compares them to values of the attributes of routes in the FIB. This allows the router to determine the appropriate route for the packet (or to determine that the packet should not be forwarded because no matching route exists).

Because the size of the FIB would likely be unwieldy if it contained a route for each possible combination of attributes a packet might have, comparing the attributes of the packet to the attributes of the routes in the FIB is not usually a straightforward

equality test of the attributes. Instead, the router uses a *route lookup algorithm* to determine which route best matches the attributes of the packet.

It is important to note that the FIB talked about in this paper is not exactly the same as the forwarding table found in most routers. The FIB contains all of the current routing information which has been provided to the router by its routing protocols. A router can often determine that its route lookup algorithm would never choose certain routes, and that it may therefore safely discard them. The fact that real implementations usually partially precompute the results of the route lookup algorithm in this manner is an important bit of implementation-related complexity which is nonetheless ignored throughout most of this memo.

1.2 The Problem

As I shall discuss in greater detail shortly, there has traditionally been a consensus on an appropriate route lookup algorithm. Unfortunately, a number of factors have conspired to render that algorithm obsolete:

- New routing protocols are being developed which require different, incompatible route lookup algorithms. Examples of such protocols in the Internet protocol suite include OSPF [7] and Dual IS-IS [3].
- The traditional route lookup algorithm makes its decision based on only one attribute of the packet being forwarded: its destination address. There is a growing consensus that the algorithm should also consider the Network Layer quality of service (TOS) requested by the packet. Some people believe that additional attributes should also be considered in the forwarding decision.
- The traditional route lookup algorithm assumes that a router is a part of (at most) one routing domain, and therefore contains no mechanism for choosing between routes to the same destination learned from different routing protocols (or multiple instances of the same routing protocol). To solve this, router implementors have had to invent a variety of policy mechanisms. Unfortunately, these mechanisms are highly implementation-dependent, contributing to the difficulties of building a network using routers from multiple vendors.

1.3 Outline of this Paper

Section 2 goes briefly through the history of this topic, and describes the traditional view of how routers decide how to route packets.

Section 3 provides a generic model of how route lookup algorithms work, and explains related terminology that is used in this paper.

Section 4 gives concise but fairly precise descriptions of several route lookup algorithms which are used or have been proposed. It also discusses the good and bad points of each.

Section 5 considers the lessons we might learn from Section 4. Is there one "correct" route lookup algorithm? How could a router choose an appropriate route when it learns routes from several routing protocols simultaneously? Several mechanisms are discussed, and one of them (the "Meta-Lookup Algorithm" approach) is picked as the most workable alternative.

Section 6 considers the Meta-Lookup Algorithm approach in detail. A specific meta-lookup algorithm is proposed, along with some policy controls.

Section 7 briefly considers efficient implementation.

2. Some Historical Perspective

First, a little history. I begin by presenting what I think of as the "classic model" of how a router makes routing decisions. This model predates IP (see, for example, reference [10]). Under this model, a router speaks some single routing protocol, an IGP such as RIP [4]. The protocol completely determines the contents of the router's FIB. The route lookup algorithm is trivial: the router looks in the FIB for a route whose destination attribute exactly matches the network portion of the destination address in the packet. If one is found, it is used; if none is found, the destination is unreachable. Because the routing protocol keeps at most one route to each destination, the problem of what to do when there are multiple routes which match the same destination cannot arise.

Over the years, this classic model has been augmented in small ways. With the advent of default routes, subnets, and host routes, it was possible to have more than one routing table entry which in some sense matched the destination. This was easily resolved by a consensus that there was a hierarchy of routes: host routes should be preferred over subnet routes, subnet routes over net routes, and net routes over default routes.

With the advent of variable length subnet masks, the general approach remained the same although its description became a little more complicated. We now say that each route has a bit mask associated with it. If a particular bit in a route's bit mask is set, the corresponding bit in the route's destination attribute is significant. A route cannot be used to route a packet unless each significant bit in the route's destination attribute matches the corresponding bit in the packet's destination address, and routes with more bits set in their masks are better than routes which have fewer bits set in their masks. This is simply a generalization of the hierarchy of routes described above, and will be referred to for the rest of this paper as choosing a route by preferring *longest match*.

Another way the classic model has been augmented is through a small amount of relaxation of the notion that a routing protocol has complete control over the contents of the routing table. First, static routes were introduced. For the first time, it was possible to simultaneously have two routes (one dynamic and one static) to the same destination. When this happened, a router had to have a policy (in some

cases configurable, and in other cases chosen by the author of the router's software) which determined whether the static route or the dynamic route would be preferred. However, this policy was only used as a tie-breaker when longest match didn't uniquely determine which route to use. Thus, for example, a static default route would never be preferred over a dynamic net route even if the policy preferred static routes over dynamic routes.

After static routes there came EGP [8]. At each Internet site, there was a strange new beast called an "exterior gateway", which spoke EGP to one or more BBN Core routers (which made up the Internet backbone at the time) at the same time as it spoke its IGP to the other routers at its site. Both protocols wanted to determine the contents of the router's routing table. Theoretically, this could result in a router having three routes (EGP, IGP, and static) to the same destination. Because of the Internet topology at the time, it was resolved with little debate that routers would be best served by a policy of preferring IGP routes over EGP routes. However, the sanctity of longest match remained unquestioned: a default route learned from the IGP would never be preferred over a net route from learned EGP.

Although the Internet topology, and consequently routing in the Internet, have evolved considerably since then, this slightly augmented version of the classic model has survived pretty much intact to this day in the Internet (except for routers which use the OSPF or the Dual IS-IS protocols, which will be discussed shortly). Conceptually (and often in implementation) each router has a routing table and one or more routing protocol processes. Each of these processes can add any entry that it pleases, and can delete or modify any entry that it has created. When routing a packet, the router picks the best route using longest match, augmented with a policy mechanism to break ties. Although this augmented classic model has served us well, it has a number of shortcomings:

- It ignores (although it could be augmented to consider) path characteristics such as type of service or MTU.
- Nobody has been able to devise policy mechanisms for tie-breaking which can easily be configured to do the "right thing".
- It doesn't support routing protocols (such as OSPF and dual IS-IS) that require route lookup algorithms different than pure longest match.

3. Route Lookup Algorithms

A route lookup algorithm chooses the best route (if there is one) for a packet from those available in the FIB.

Conceptually, any route lookup algorithm starts out with a set of candidate routes which consists of the entire contents of the FIB. The algorithm consists of a series of steps which discard routes from the set. In this paper, these steps are referred to as *pruning rules*. Normally, when the algorithm terminates there is exactly one route

remaining in the set. If the set ever becomes empty, the packet is discarded because the destination is unreachable. It is also possible for the algorithm to terminate when more than one route remains in the set. In this case, the router may arbitrarily discard all but one of them, or may perform “load-splitting” by choosing whichever of the routes has been least recently used. Load-splitting is not discussed in detail in this memo.

Some steps are mandatory, and must be applied even if the set has already been reduced to a single route. Some algorithms also contain optional steps, which are applied only if the set contains more than one route. In general, the steps must be applied in order, since rearranging the steps can change the result.

3.1 Pruning Rules Used in This Paper

There are a number of common pruning rules. The route lookup algorithms in use generally use some combination os these rules.

Basic Match This step discards any routes to destinations other than the destination of the packet. For example, if a packet was addressed to 36.144.2.5, this step would discard a route to net 128.12.0.0 but would retain any routes to net 36.0.0.0, any routes to subnet 36.144.4.0.0, and any default routes.

More precisely, we assume that each route has a destination attribute, which we’ll call “route.dest”, and a corresponding mask, which we’ll call “route.mask”, to specify which bits of route.dest are significant. We’ll call the IP destination address of the packet being routed “ip.dest”. This step discards all routes from the set of candidate routes except those for which (route.dest & route.mask) = (ip.dest & route.mask). This is (logically, at least) the first step performed by any route lookup algorithm.

Longest Match This step was described on page 5.

The precise definition of longest match is a refinement of basic match, described above. After the basic match pruning is performed, the remaining routes are examined to determine the maximum number of one bits set in any of their route.mask attributes. The step then discards from the set of candidate routes any routes which have fewer than that maximum number of one bits set in their route.mask attributes.

Weak TOS There is a growing consensus that routers should consider the IP type of service requested when making their routing decisions. This could be done in a number of ways, but the

one that seems to have the largest group of proponents is one I refer to as Weak TOS. Under this method, a route which has the exact TOS requested is used if there is one; otherwise, a route with the default (zero) TOS is used if there is one. Routes which have a TOS which is neither the default are not considered, even if they are the only available routes.

More precisely, we assume that each route has a type of service attribute, which we'll call "route.tos", whose bits correspond to those in the TOS field of the IP header. Routing protocols which distribute TOS information fill in route.tos appropriately; other routing protocols set route.tos to the default TOS. We'll call the TOS field in the IP header of the packet being routed "ip.tos". The set of candidate routes is examined to determine if it contains any routes for which route.tos = ip.tos. If so, all routes except those for which route.tos = ip.tos are discarded. If not, all routes except those for which route.tos = 0 are discarded from the set of candidate routes. Additional discussion of routing based on Weak TOS may be found in [2].

IS-IS Route Class Routing protocols which have areas or make a distinction between internal and external routes divide their routes into classes, where classes are rank-ordered in terms of preference. A route is always chosen from the most preferred class unless none is available, in which case one is chosen from the second most preferred class, and so on. In Dual IS-IS, the classes (in order from most preferred to least preferred) are intra-area, inter-area, and external (where external routes with internal metrics are considered to be inter-area).

More precisely, we assume that each route has a class attribute, called "route.class", which is assigned by the routing protocol. The set of candidate routes is examined to determine if it contains any for which route.class = "intra-area". If so, all routes except those for which route.class = "intra-area" are discarded. Otherwise, the set of candidate routes is examined to determine if it contains any for which route.class = "inter-area". If so, all routes except those for which route.class = "inter-area" are discarded.

OSPF Route Class OSPF route classes work identically to Dual IS-IS's. However, the set of classes defined by OSPF is different, such that there isn't a one-to-one mapping between OSPF

route classes and IS-IS route classes. The route classes used by OSPF (in order from most preferred to least preferred) are intra-area, inter-area, type 1 external, and type 2 external.

The OSPF internal class is equivalent to the Dual IS-IS internal class. Likewise, the OSPF external type 2 is equivalent to Dual IS-IS's external class. However, Dual IS-IS's inter-area class is represented by two classes in OSPF: inter-area, for true inter-area routes, and external type 1, for external routes with internal metrics. Thus, OSPF prefer true inter-area routes over external routes with internal metrics, whereas Dual IS-IS gives the two types of routes equal preference.

Best Metric

Routes have metrics, to allow the router to compare the relative goodness of otherwise equally attractive candidate routes. However, since metrics are meaningful only within a routing domain, it makes no sense to compare the metrics of routes learned from different routing domains.

More precisely, we assume that each route has a metric attribute, called route.metric, and a routing domain identifier, called route.domain. Each member of the set of candidate routes is compared with each other member of the set. If route.domain is equal for the two routes and route.metric is strictly "inferior" for one than the other, the one with the inferior metric is discarded from the set. The determination of "inferior" is usually by a simple arithmetic comparison, though some protocols may have structured metrics requiring more complex comparisons.

Policy

Policy is sort of a catch-all to make up for the fact that the previously listed rules are often inadequate to chose from among the possible routes. Policy pruning rules are extremely vendor-specific.

IDPR Policy

A specific case of Policy. The IETF's Inter-domain Policy Routing Working Group is devising a routing protocol called Inter-Domain Policy Routing (IDPR) [5],[9] to support true policy-based routing in the Internet. Packets with certain combinations of header attributes (such as specific combinations of source and destination addresses or special IDPR source route options) are required to use routes provided by the IDPR protocol. Thus, unlike other Policy

pruning rules, IDPR Policy would have to be applied before any other pruning rules except Basic Match.

Specifically, IDPR Policy examines the packet being forwarded to ascertain if its attributes require that it be forwarded using policy-based routes. If so, IDPR Policy deletes all routes not provided by the IDPR protocol.

3.2 Example of Other Possible Pruning Rules

Of course, many other pruning rules are possible, though are not used in any of the route lookup algorithms discussed in this paper. Here are a few examples:

Strong TOS Similar to Weak TOS described above, but does not fall back to a default TOS if no suitable route with the requested TOS is available.

Very Weak TOS Also similar to the Weak TOS described above. However, this allows use of routes whose TOS is neither the TOS requested or the default TOS when no routes are available with either the requested or default TOS. Very Weak TOS was used in very early versions of the algorithm describes in section 4.2, but was discarded in favor of Weak TOS in later versions.

MTU If any of routes in the set of candidate routes have an MTU large enough to forward the packet without fragmenting it then all routes which would require fragmentation are discarded from the set.

Source AS A specific case of Policy. Routes are tagged with the AS number from which they were learned, and a (perhaps partial) preference ordering is established over the AS numbers. If some routes in the set of candidate routes have AS numbers which are less preferred than the AS numbers of some of the other routes, those routes whose AS numbers are less preferred are discarded from the set.

4. Some Route Lookup Algorithms

This section examines several route lookup algorithms that are in use or have been proposed. Each is described by giving the sequence of pruning rules it uses. The strengths and weaknesses of each are also enumerated.

4.1 The Revised Classic Algorithm

The Revised Classic Algorithm is the current version of the traditional algorithm discussed on page 5. The steps of this algorithm are:

1. Basic match
2. Longest match
3. Best metric
4. Policy

Some implementors omit the policy step, since it is needed only when routes may have metrics that are not comparable (because they were learned using different routing protocols).

The advantages of this algorithm are:

1. It is widely implemented.
2. Except for the Policy step (which an implementor can choose to make arbitrarily complex) the algorithm is simple both to understand and to implement.

As I pointed out in Section 1.2, its disadvantages are:

1. It does not handle IS-IS or OSPF route classes, and therefore cannot be used for Dual IS-IS or OSPF.
2. It does not handle TOS or other path attributes.
3. The policy mechanisms are not standardized in any way, and are therefore are often implementation-specific. This causes extra work for implementors (who must invent appropriate policy mechanisms) and for users (who must learn how to use the mechanisms). This lack of standardization also makes it difficult to create consistent configurations for routers from different vendors. This presents significant deterrent to practical multi-vendor interoperability.
4. The proprietary policy mechanisms currently provided by vendors are often inadequate in complex parts of the Internet.

4.2 The Router Requirements Algorithm

The IETF Router Requirements Working Group has produced a number of draft versions of an RFC-1009 replacement. The early drafts mandated a route lookup algorithm which I will call the Router Requirements Algorithm. This algorithm is similar to the Revised Classic Algorithm, with two important differences. The first is that it includes support for type of service routing. The second is that it attempts to extend the Revised Classic Algorithm to explicitly support routers which are in more than one routing domain, primarily by standardizing the policy mechanisms (though still allowing implementors to supplement the standard mechanisms with proprietary ones).

The steps in the Router Requirements Algorithm are:

1. Basic match

2. Longest match
3. Weak TOS
4. Best metric
5. Policy

This model has some advantages over the Revised Classic Algorithm:

1. It supports type of service routing.
2. Its rules are written down, rather than merely being a part of the Internet folklore.
3. Partial standardization of the policy step (hopefully) will make life a lot easier for managers of those parts of the Internet where routing is complex.

However, this algorithm retains some of the disadvantages of the Revised Classic Algorithm:

1. It does not handle IS-IS or OSPF route classes, and therefore cannot be used for Dual IS-IS or OSPF.
2. Path properties other than type of service (e.g. MTU) are ignored.

4.3 The Variant Router Requirements Algorithm

Some Router Requirements Working Group members have proposed a slight variant of the algorithm described in the previous section. In this variant, matching the type of service requested is considered to be more important, rather than less important, than matching as much of the destination address as possible. For example, this algorithm would prefer a default route which had the correct type of service over a network route which had the default type of service, whereas the algorithm in the previous section would make the opposite choice.

The steps of the algorithm are:

1. Basic match
2. Weak TOS
3. Longest match
4. Best metric
5. Policy

Debate between the proponents of this algorithm and the regular Router Requirements Algorithm suggests that each side can show cases where its algorithm leads to simpler, more intuitive routing than the other's algorithm does (for further discussion of this topic, see [2]). In general, this variant has the same set of advantages and disadvantages that are described in the previous section, except that pruning on Weak TOS before pruning on Longest Match makes this

algorithm less compatible with OSPF and Dual IS-IS than the standard Router Requirements Algorithm.

4.4 The OSPF Algorithm

OSPF uses an algorithm which is virtually identical to the Router Requirements Algorithm except for one crucial difference: OSPF considers OSPF route classes. Also, although I have listed a Policy step in the algorithm, the OSPF document does not discuss this step at all, much less attempt to standardize it.

The algorithm is[1]:

1. Basic match
2. OSPF route class
3. Longest match
4. Weak TOS
5. Best metric
6. Policy

This algorithm has some advantages over the Revised Classic Algorithm:

1. It supports type of service routing.
2. Its rules are written down, rather than merely being a part of the Internet folklore.
3. It (obviously) works with OSPF.

However, this algorithm also retains some of the disadvantages of the Revised Classic Algorithm:

1. Path properties other than type of service (e.g. MTU) are ignored.
2. As in the Revised Classic Algorithm, the details (or even the existence) of the Policy step is left to the discretion of the implementor.

The OSPF Algorithm also has a further disadvantage (which is not shared by the Revised Classic Algorithm):

1. OSPF internal (intra-area or inter-area) routes are always considered to be superior to routes learned from other routing protocols, even in cases where the OSPF route matches fewer bits of the destination address. This is a policy decision that is inappropriate in some networks.

1. Some drafts of the OSPF version 2 specification have had Weak TOS immediately after step 1 instead of after step 3. The OSPF version 2 RFC, due out very shortly, will use the algorithm described here.

4.5 The Dual IS-IS Algorithm

The most recent version of Dual IS-IS uses an algorithm which is similar to but not quite identical to the OSPF Algorithm. Dual IS-IS uses a different set of route classes, and also differs slightly in its handling of type of service. The algorithm is:

1. Basic match
2. IS-IS route class
3. Longest match
4. Weak TOS
5. Best metric
6. Policy

Although Dual IS-IS uses Weak TOS, the protocol is only capable of carrying routes for a small specific subset (defined in Section 3.5 of [3]) of the possible values for the TOS field in the IP header. Packets containing other values in the TOS field are routed using the default TOS.

Type of service support is optional; if disabled, the fourth step would be omitted. As in OSPF, the specification does not include the Policy step.

This algorithm has some advantages over the Revised Classic Algorithm:

1. It supports type of service routing.
2. It's rules are written down, rather than merely being a part of the Internet folklore.
3. It (obviously) works with Dual IS-IS.

However, this algorithm also retains some of the disadvantages of the Revised Classic Algorithm:

1. Path properties other than type of service (e.g. MTU) are ignored.
2. As in the Revised Classic Algorithm, the details (or even the existence) of the Policy step is left to the discretion of the implementor.
3. It doesn't work with OSPF because of the differences between IS-IS route classes and OSPF route classes. Also, because IS-IS supports only a subset of the possible TOS values, some obvious implementations of the Dual IS-IS algorithm would not support OSPF's interpretation of TOS.

The Dual IS-IS Algorithm also has a further disadvantage (which is not shared by the Revised Classic Algorithm):

1. IS-IS internal (intra-area or inter-area) routes are always considered to be superior to routes learned from other routing protocols, even in cases where the IS-IS route matches fewer bits of the destination address and doesn't provide the requested type of service. This is a policy decision that may not be appropriate in all cases.

5. The Problem Revisited

In the previous section, we examined the routing algorithms used in the Internet, as well as a couple of proposed alternative ones. Among other things, we noted that OSPF and Dual IS-IS each require a specialized route lookup algorithm that doesn't work with any other Internet routing protocol. This is certainly unfortunate for implementors, since those who want to support both the new routing protocols and the traditional ones have to implement three different route lookup algorithms. However, things become even more problematic when a router wishes to simultaneously use multiple routing protocols. Unfortunately, the growing complexity of the Internet is requiring more and more routers to do just that.

Suppose, for example, that a router is simultaneously using RIP and Dual IS-IS, and a packet arrives at the router, waiting to be forwarded. Does the router use Dual IS-IS's route lookup algorithm, or the Revised Classic Algorithm that is commonly used with RIP? Does it try both, and hope that they both give the same answer? The rest of this section looks at several ways out of this dilemma that have been proposed.

5.1 The "Master Protocol" Approach

In the "master protocol" approach, a single routing protocol is given responsibility for maintaining the routing table. A route lookup algorithm appropriate for use with that routing protocol is used. Routes learned from other routing protocols are "leaked" into the master protocol, which generally considers these routes to be "external" (where the meaning of "external" depends on the particular master protocol in use).

The obvious advantage to this approach is that it requires no change to whatever route lookup algorithm is already being used. However, there are a number of substantial drawbacks:

1. If the master protocol used is one which automatically treats "external" routes as inferior, the protocol's own routes will always be considered better than routes learned from other routing protocols, even in cases where the protocol's own route matches fewer bits of the destination address and doesn't provide the requested type of service. This is a policy decision which may not be appropriate in all cases.

2. Redistributing routes into the master protocol can be non-trivial if the protocol whose routes are being redistributed uses a route lookup algorithm different than that used by the master protocol.
3. Most implementations assume that routes redistributed into the master protocol ought to be announced to other routers by the master protocol. This may or may not be desirable.

5.2 The "Rank Ordering of Routing Protocols" Approach

In this approach, the routing protocols in use are ranked according to a preference ordering. There is no global FIB; each routing protocol maintains its own, separate routing database. When a packet is to be routed, the most preferred routing protocol is queried to see if it has a route to the destination. That protocol uses a route lookup algorithm which is appropriate to it to determine whether it has a route. If so, that route is used; if not, the second most preferred routing protocol is queried, etc., until either a route is found or all of the routing protocols have been queried.

This approach also has a pleasant simplicity, and avoids some of the problems of the previous approach. However:

1. It potentially requires several route table lookups for each packet switched.
2. It is often not possible to rank order the protocols in a sensible way. More often, each protocol will supply some routes which the router would like to consider primary and some routes which it would like to consider to be backup routes.
3. A route provided by a more preferred routing protocol will be chosen even if it matches fewer bits of the destination address than a route provided by a less-preferred protocol. This may or may not be desirable.

5.3 The "Multiple Virtual Router" Approach

This approach was suggested at the Boulder IETF meeting. I thought I understood it at the time, but on further reflection I'm not sure that I do. Therefore, this description is probably sorely mangled.

In this approach, each interface on the router is considered to be a virtual router. Each virtual router is restricted to running a single routing protocol, which determines the contents of that virtual router's routing table. Each virtual router routes all (and only) the packets that arrive on its interface. The part I don't understand: it would seem that there must also be some virtual routing protocol running between the virtual routers. Presumably, each virtual router would use a Rank Ordering approach which preferred its own protocol over the virtual protocol.

This approach has several disadvantages:

1. Each interface has to run exactly one routing protocol (I'm not sure how protocols such as EGP, which don't really have the concept of interfaces, fit into this).
2. The router has to be able to determine which virtual router should receive each packet. Thus all logical interfaces on a physical interface must use the same routing protocol (since they must share the same logical router).
3. If a virtual router doesn't have a route to a destination, it has to choose which other virtual router to send the packet to.Thus, this approach doesn't really solve the problem, but merely shifts the problem of choosing a preferred route from the route lookup algorithm to the virtual routing protocol.

5.4 The "Meta-lookup Algorithm" Approach

As in the Rank Ordering approach, each routing protocol that is being used maintains its own independent routing database. When a packet is to be routed, each active routing protocol is asked to submit a candidate route. Each protocol chooses a route (or determines that it has no appropriate route) using a route lookup algorithm appropriate to the particular routing protocol.

After this takes place, the router is left with a set of candidate routes, containing at most one route suggested by each of the routing protocols. The router then uses a special route lookup algorithm to choose from among those routes.

This approach has some obvious disadvantages:

1. A router running `n` routing protocols has to do `n+1` route lookups per packet.
2. It isn't entirely obvious what the meta-lookup algorithm should be.

However, this approach has a number of fairly strong advantages:

1. Depending on what meta-lookup algorithm is chosen, it can support virtually any policy, including those supported by the previously listed options.
2. Unlike the previous options, it can support a policy of using the route from the routing protocol that provides the route which best matches the destination and TOS of the packet. This probably matches most peoples' intuitions about what the router ought to do. Given the complexity and difficulty of routing, I consider it extremely important that the solution match peoples' intuition.
3. Because the routing protocols operate independently of each other and maintain their own routing tables, there is no need to worry that interactions between the protocols might make some of the protocols act incorrectly.
4. Likewise, because each routing protocol uses its own route lookup algorithm on its own routing table, it is easy to ensure that any route chosen by this approach will in fact be the preferred route of one of the routing protocols. This greatly reduces the probability of routing loops.

Because of its strong advantages, the meta-lookup algorithm method seems to be the most promising choice. The remainder of this paper considers this approach in greater detail, and considers how its disadvantages can be mitigated.

6. The Meta-Lookup Algorithm Approach in Detail

So far, this paper has discussed the problem of how a router which simultaneously uses several routing protocols might choose an appropriate route to a particular destination. The previous section described several approaches to solving the problem, and identified the Meta-Lookup Algorithm approach as the most promising. This section considers that approach in greater detail.

6.1 The Recommended Meta-Lookup Algorithm

One of the things that gives the Meta-lookup Algorithm approach so much flexibility is that any meta-lookup algorithm one wants to choose will work and will not violate the assumptions of any of the router's routing protocols. However, it would be undesirable to leave the algorithm completely to the discretion of the implementor, since then it would continue to be difficult if not impossible to configure routers from different vendors in a consistent manner.

To me, the most obvious candidate for a meta-lookup algorithm would be a stripped down version of the Revised Classic Algorithm from Section 4.1 on page 10. This algorithm is familiar to both implementors and users, and intuitively does "the right thing" in most cases.

The Basic Match pruning step can be omitted, since it can be safely assumed that the none of the routes provided by the protocol-specific route lookups would be discarded by a Basic Match pruning step. Likewise, the Best Metric pruning step would never discard any routes, and can therfore be omitted. The reason for this is that (as described on page page 9) Best Metric only compares metrics of routes from the same routing domain.

One major shortcoming of the Revised Classic Algorithm for this purpose is that it would have to be augmented to include an IDPR Policy pruning step (described on page 9) in order to support the IETF Interdomain Policy Routing Working Group's IDPR protocol. This could be rectified by replacing the Basic Match pruning step in the Revised Classic Algorithm with an IDPR Policy pruning step. The other major shortcoming of the Revised Classic Algorithm for our purposes is that it leaves the details of the Policy Pruning Step unspecified, an omission that would need to be corrected to ensure easy interoperability among routers from different vendors. The next section suggests a possible standard Policy Pruning step.

Some people have suggested that the Router Requirements Algorithm (described in Section 4.2 on page 11) ought to be used in place of the Revised Classic Algorithm, since the former supports TOS while the latter does not. Using the

Router Requirements Algorithm would make it more likely that a route with the correct TOS would be chosen, but at the expense of implicitly preferring protocols that support TOS. Even with the Revised Classic Algorithm, no route would be chosen that didn't have either the requested TOS or the default TOS as long as all routing protocols which support TOS continue to choose Weak TOS over alternative pruning rules to select routes of an appropriate TOS.

To summarize, then, the recommended meta-lookup algorithm is:

1. IDPR policy
2. Longest match
3. Policy

where the first step may be omitted by routers which do not support the IDPR protocol.

6.2 The Recommended Policy Pruning Mechanism

In the previous section I observed that one difficulty with using the Revised Classic Algorithm as the standard meta-lookup algorithm is that it doesn't clearly specify what the Policy pruning step does. As a result, it can differ vastly between different implementations. This has been useful in that it has allowed the Internet community to experiment with a number of approaches, but has also lead to diminished interoperability among different implementations.

Therefore, this section suggests what I think would make a good standard Policy pruning step. In order to encourage continued experimentation, this section should be considered to propose some baseline capabilities, which implementors could choose to augment in ways of their own devising. These baseline capabilities are primarily not of my own invention; most can be found in *gated* or in existing commercial routers.

The basic idea behind the recommended Policy Pruning mechanism is that each route has associated with it a preference value, based on various attributes of the route. This preference value is an integer in the range [0..255], with zero being the most preferred and 254 being the least preferred. 255 is a special value that means that the route should never be used[1]. The Policy Pruning mechanism uses the preference value to eliminate less preferable routes.

1. Unfortunately, knowledge of this 255 value cannot be limited just to the Policy pruning step; the Basic Match step in the protocol-specific route lookup algorithms needs to discard routes whose preference value is 255. This is noted (only) here because its not obvious to me where in this memo this ought to go. I suspect that an actual implementation could (and would) simply discard these routes without installing them in any route databases.

6.2.1 Details of the Policy Pruning Step

As the first step in Policy Pruning, the preference values of the candidate routes are compared. Any route whose preference value is strictly less than the preference value of another candidate route is discarded.

If multiple routes remain after the first step (because they all share the same preference value), a second step is applied. If the set of candidate routes contains any routes learned via the EGP protocol, their metrics are compared. Any route learned from EGP whose metric is strictly higher than another route learned from EGP is discarded. This step makes no theoretical sense, since metrics learned from different EGP autonomous systems are theoretically not comparable, but this kluge is important for backwards compatability with the way routing is done in a number of places in the Internet.

If multiple routes remain after the first two steps, a router has several options. It may reduce the number of candidate routes by applying steps similar to the second step to routes learned from other routing protocols. It may arbitrarily discard some of the routes. It may retain more than one route and employ a “load-splitting” mechanism to divide traffic among them. Perhaps the only thing that can be said about the relative merits of the options is that load-splitting is useful in some situations but not in others, so a wise implementor who implements load-splitting will also provide a way for the network manager to disable it.

6.2.2 Assignment of Preference Values

It is important to give network managers adequate flexibility in assignment of preference values so that they can implement whatever policies they desire. In practice, this seems to require the ability to assign preference values at the granularity of individual routes. However, since a router may have thousands of routes, it is equally important that the network manager not be required to specify an individual preference value for each and every route. The following mechanisms seem to provide adequate expressive power without overly complicating the configuration process.

The implementor of the router chooses for each routing protocol (and for static routes) a default preference value. There is no standard way of choosing these preference values, but a good rule of thumb is to choose the values in such a way that static routes are preferred over routes learned from an IGP, and routes learned from an IGP are in turn preferred over routes learned from an inter-domain routing protocol (e.g. EGP or BGP). It is probably also wise to choose the default preference values such that no two routing protocols have the same default preference value.

The router associates with each routing domain it is a member of a default preference value, which will be assigned to any routes which are learned from that

routing domain but which do not have their preference value set by one of the mechanisms below. The router should allow the network manager to specify for each routing domain (including static routes) the default preference value for the routing domain. For any routing domain where the network manager has not explicitly chosen a default preference value, the default preference value for the routing protocol should be used as the default preference value for the routing domain.

The above mechanisms are usually adequate to assign preference values to most routes, but additional mechanism is needed to provide the fine-grained control. Although the only remaining mechanism that is strictly necessary is a mechanism to assign specific preferences to individual routes, configuration can be considerably simplified if mechanisms are provided to assign a particular preference value to all routes from the same routing domain which share a certain characteristic. The following mechanisms for classifying routes have proven to be useful (and, conveniently, is a superset of the set of classifications used by the route leaking mechanisms described in [1]):

Address range — It is useful to be able to specify a single preference value for all routes (learned from the same routing domain) to destinations which fall (numerically) into an address range that is a member of a specified set of address ranges.

Milo's Number[1] — It is useful to be able to specify a single preference value for all OSPF routes (learned from the same routing domain) which have in their Milo's Number field any of a set of specified values.

AS path — It is useful to be able to specify a single preference value for all BGP routes (learned from the same routing domain) whose AS path "matches" any of a set of specified values. It is not yet clear exactly what kinds of matches are most useful. A simple and reasonably useful option would be to allow matching of all routes which do (or do not) have a specified AS number at some point in the AS path. A more general but somewhat more difficult alternative would be to allow matching all routes for which the AS path matches a specified regular expression.

Route class — For routing protocols which maintain the distinction, it is useful to be able to specify a single preference value for all routes (learned from the same routing domain) which have a

1. This is the four octet tag field which OSPF carries with external routes. It's use is outside of the scope of the OSPF specification; it is intended for use by border routers to communicate among themselves. Most commonly, it is used to carry the autonomous system number of the source of the route. I need to check the OSPF specification to find the real name of this field.

particular route class (intra-area, inter-area, external with internal metrics, or external with external metrics).

Interface — It is useful to be able to specify a single preference value for all routes (learned from a particular routing domain) that would cause packets to be routed out a particular logical interface on the router.

Source Router — It is useful to be able to specify a single preference value for all routes (learned from the same routing domain) which were learned from any of a set of routers. A list of address ranges is a convenient way to describe the set. It may also be useful to be able to specify a single preference value for all routes learned from routers which are not in a specifed set.

It is important to note that routes learned from some routing protocol must never be assigned a preference value of 255, since that would cause the router to violate the rules of the protocol in ways that could cause routing problems. In particular, a route learned from SPF-based protocols such as OSPF and Dual IS-IS must never be assigned a preference value of 255.

7. Performance and Implementation Issues

This section will look at ways to mitigate the problem that the Meta-Lookup Algorithm approach (conceptually at least) requires `n+1` route table lookups for each packet forwarded. It assumes that the meta-lookup algorithm being used is the one from Section 6.1 on page 18, but the general approach is probably applicable to any meta-lookup algorithm.

7.1 Serialization of the Recommended Algorithm

Although it is conceptually necessary to do several route table lookups and then afterwards apply the meta-lookup algorithm to the results, it is possible to do everything necessary in a single pass through the FIB. The precise algorithm required to do this depends on which route lookup algorithms are used by the routing protocols that are supported, but the following algorithm is believed to support all of the existing IP routing protocols[1]:

1. Basic match
2. IDPR policy
3. Generic route class
4. Longest match

1. Except for some draft versions of OSPF version 2. The RFC version of OSPF version 2, to be issued shortly, is compatible with this algorithm.

5. Generic weak TOS
6. Best metric
7. Policy

Most of the steps are pruning rules which have been previously discussed, but pruning rules 3 and 5 are modified versions of previously discussed rules, and are defined as follows:

Generic Route Class This is simply an agglomeration of the protocol-specific route class pruning rules. OSPF route class pruning (which was described on page 8) is applied to any OSPF routes (but not any other routes) in the set of candidate routes. Likewise, IS-IS route class pruning (which was described on page 8) is applied to any Dual IS-IS routes in the set of candidate routes. If any other routing protocol are used which also support route classes, protocol-specific route class pruning would likewise be applied to those routes.

Generic Weak TOS This is exactly like Weak TOS, except that is applied separately to the routes learned from each routing protocol. Thus, for example, finding an OSPF route whose TOS matched the TOS requested in the packet might cause other OSPF routes to be discarded, but would not cause RIP or Dual IS-IS routes to be discarded.

Using this algorithm is more time-consuming than using one of the traditional route lookup algorithms, but should prove significantly speedier than performing one route lookup per protocol and then executing the meta-lookup algorithm on the resultant routes.

7.2 Precomputation of Routes

Further speedup is (probably) possible by generating a forwarding table such as is commonly used in routers.[1]

7.3 Route Cacheing

TBD (when you look up a route, cache a <destination address, TOS, next hop address> tuple. When such cacheing is used, the meta-lookup algorithm is executed only when there is a cache miss, so the algorithm's (lack of) performance is less important.)

1. I haven't had the time to figure out and write down the details, but it would seem that creating the table would be reasonably straightforward for routing protocols which can be handled by the serialized algorithm in section 7.1. Incremental update seems like a much harder problem.

8. Conclusion

This paper looked closely at how routers have traditionally chosen how to route packets to their destinations, and how that traditional method of choice is being overrun by changes in routing protocols and by the ever-increasing complexity of the Internet. The paper then examined several alternatives to the traditional method, and sketched out some of the details of a preferred one.

References

[1] P. Almquist
Ruminations on Route Leaking
Paper in progress.

[2] P. Almquist
Type of Service in the Internet Protocol
Paper in progress.

[3] R. Callon
Use of OSI IS-IS for Routing in TCP/IP and Dual Environments
Request for Comments (RFC) 1195, DDN Network Information Center, SRI International, Menlo Park, California, USA, December 1990.

[4] C. L. Hedrick
Routing Information Protocol
Request for Comments (RFC) 1058, DDN Network Information Center, SRI International, Menlo Park, California, USA, June 1988.

[5] M. Lepp and M. Steenstrup
An Architecture for Inter-Domain Policy Routing
To be published as an RFC; currently an INTERNET DRAFT dated February 1990.

[6] K. Lougheed and Y. Rekhter
Border Gateway Protocol (BGP)
Request for Comments (RFC) 1163, DDN Network Information Center, SRI International, Menlo Park, California, USA, June 1990.

[7] J. Moy
OSPF Specification, Version 2
To be published as an RFC; currently an INTERNET DRAFT dated June 1990.

[8] L. J. Seamonson and E. C. Rosen
"STUB" Exterior Gateway Protocol
Request for Comments (RFC) 888, DDN Network Information Center, SRI International, Menlo Park, California, USA, January 1984.

[9] M. Steenstrup
Inter-Domain Policy Routing Protocol Specification and Usage: Version 1
To be published as an RFC; currently an INTERNET DRAFT dated February 1991.

[10] E. Taft
Gateway Information Protocol (Revised)
Xerox Inter-Office Memorandum to: Communication Protocols, location: PARC/CSL, file: [Maxc1]<Pup>GatewayInformation.press, date: May 30, 1979.

[11] ??? (Gated reference)

Security Considerations

Security considerations are not addressed in this memo.

Author's Address

Philip Almquist
214 Cole Street, Suite 2
San Francisco, CA 94117-1916

Phone: 415-752-2427

EMail: almquist@Jessica.Stanford.EDU