Rbridges: Transparent Routing

Radia Perlman
Sun Microsystems Laboratories
radia.perlman@sun.com

Abstract— **This paper describes a method of interconnecting links that combines the advantages of bridging and routing. The basic design is a replacement for a transparent bridge and makes no assumption about higher layer protocols. It involves creating an infrastructure of switches (which we call Rbridges, for "routing bridges") in which packets are routed, although, as with bridges, layer 2 endnode location is learned through receipt of data packets. It avoids the disadvantages of bridges, since packets within the infrastructure need not be confined to a spanning tree, and packets are protected with a hop count and not proliferated while in transit, so there is no need for any artificial startup delay on ports to avoid temporary loops. This allows IP nodes to travel within a multi-link campus without changing IP addresses. The paper introduces further optimizations for IP, such as avoiding flooding ARP messages through the infrastructure, and (for IP nodes), allowing Rbridges to avoid learning on data packets.**

Index terms-- system design

I. INTRODUCTION

Bridges can transparently connect many physical links into what appears to IP (or a layer 2 protocol) to be a single LAN. However, this transparency is bought at a price. It requires the topology on which traffic is forwarded to be a tree. This causes traffic concentration on links that were chosen for the spanning tree. It also causes suboptimal paths.

In addition, bridge forwarding can be dangerous. There is no hop count in the header, and worse yet, bridges forward onto multiple ports (when the location of the destination is unknown), and multiple bridges might choose to forward a packet seen on a link. This causes exponential proliferation of packets. As a result, bridges need to be conservative about forwarding onto new links, in order to avoid temporary loops. The spanning tree as originally designed [9] used a timer to avoid temporary loops. Since then various optimizations have been proposed, such as making a special case of ports for which the neighbor is known to be an endnode (rather than another switch). If the assumption is correct that the neighbor is an endnode, there is no danger of loops by immediately starting to forward on that port. Also, [12] introduces other optimizations to avoid the delay in some cases.

Given the possibility of exponential proliferation during temporary loops, the spanning tree algorithm can become unstable, as reported recently in an event at a hospital in Boston [1]. Attempts to make it less conservative in order to forward more quickly are likely to increase the number of such incidents. In a distributed algorithm, there is no way, based on totally local information, for a bridge to know it is safe to start forwarding onto a new link. And even if such an algorithm were to be devised, a component such as a repeater can cause a loop which bridges would not be able to prevent.

So why not simply use routing? The IP protocol (v4) is pretty much universal. However, it is not universal. There are other layer 3 protocols that are in use, and there are some protocols (such as IS-IS [7], or LAT) that work directly over layer 2.

Even if the IP protocol were universal, however, it has a disadvantage. IP routes only to links. Each link is assigned an address prefix, and all IP nodes on that link must have an IP address with that prefix, and any node not on that link must have an IP address that does not start with that prefix. That means that a node that has multiple links must have multiple addresses, and a node that moves from one link to another must change its address.

Note that for the purposes of this paper, IPv6 is sufficiently similar to IPv4 that throughout this paper "IP" is intended to mean both IPv4 and IPv6. Differences such as the use of the "neighbor discovery" protocol for IPv6 vs. ARP for IPv4 are straightforward translations of functionality.

Another disadvantage of IP routing is that it requires configuration. It has improved over the years with the addition of protocols such as DHCP [3]. However, the routers need to be configured with prefixes for the links. There have been proposals for having routers within a campus, given a prefix, automatically subdivide the prefix into link-specific prefixes, so that each link within a "campus" would automatically have its own unique prefix within the shorter campus-wide prefix. This approach avoids configuration of the routers, but still has disadvantages:

- a node that moves within the campus must change its address
- a node that has attachments to multiple links must have multiple addresses
- it is wasteful of IP addresses, since it is impractical to make sure that every link is fully populated (makes full use of its prefix).

CLNP [5] is a protocol similar to IP, but unlike IP, there is no link-specific prefix. Instead there is the concept of "level 1 routing" within an area. An area may contain many links. All nodes within the area share the same prefix. A node with multiple links within the area can have a single layer 3 address, and a node that moves within the area need not change its layer

3 address. CLNP could do this because all CLNP endnodes implemented a protocol known as ES-IS [6]. The ES-IS protocol has endnodes periodically announce themselves, to an address listened to by the routers, so that all the routers on the link know which endnodes are on that link, and can detect (based on no longer receiving ES Hello messages) when an endnode has gone down or moved.

Because of the ES-IS protocol, an area can have an arbitrary physical topology and routers can route to endnodes using a shortest path algorithm.

IP endnodes cannot be relied upon to do any such announcements.

The Rbridge design presented in this paper accomplishes several things:

- it coexists with standard bridges, so that a bridged campus can be upgraded slowly, by replacing bridges one at a time with Rbridges. The more Rbridges, the more advantages of Rbridges will be gained, such as more optimal use of the topology.
- it allows interconnection of IP nodes within a campus with a result similar to a CLNP area, but without relying on IP endnodes to do anything new
- it bridges layer 2 protocols (where "bridges" means transparently interconnects), while being able to maintain shortest paths and safe routing within the campus
- as a result of working at layer 2, and making no assumptions about higher layers, it works for any layer 3 protocol
- it makes no assumptions about physical topology. Not only is the interswitch topology unconstrained, but interswitch links may be shared media, with endnodes residing on these links.
- In many cases Rbridges will support dissimilar layer 2 technologies. This paper will describe what mechanisms will be needed, and what cases will not work.

The paper also presents variations that drop various assumptions, such as the functional requirement of supporting anything other than IP, or the assumption that endnodes might reside on shared media interswitch links.

Dropping the requirement to support anything other than IP avoids the necessity for the Rbridge to learn station location from data packets, and avoids the necessity of encapsulation, two Rbridge requirements that present implementation difficulties for some switch hardware.

Note, though, that the design optimized for IP can coexist with a design that handles non-IP packets, much like brouters could route some protocols and bridge the others.

II. Other Work

Some approaches to high-speed interconnection not only require loop-free routing, but deadlock-free routing. Deadlock-free routing not only ensures that each path is individually loop-free, but that paths do not interact in a way that would ever create deadlocks. Autonet [11] is such a system. Deadlock-free routing has the disadvantage that it constrains paths, but the clever algorithm in Autonet allows near-optimal paths. However, Autonet requires globally correct routes at all times. This is accomplished by freezing the network after a topology change, and discarding all data until the network is known to have safely converged to the new topology.

Anther approach that requires freezing the network until it has converged after a topology change is SmartBridge [10]. SmartBridge additionally requires forwarding from each source to be along a spanning tree rooted at that source (and therefore this design does not allow load splitting).

The Rbridge concept does not need to be careful to avoid temporary inconsistencies while the topology is changing (due, among other things, to including a hop count in the interswitch header). This avoids the necessity to throw away data during topology transition times, and allows faster convergence, since during a topology change some routes might still be correct, and there is no need to artificially throw away packets for those routes. And the network will converge more quickly than knowledge that the network has converged, so even routes affected by the topology change will start forwarding correctly more quickly in the Rbridge scheme.

An approach similar to the Rbridge design, but with stricter topological assumptions (no shared medium links between switches) is in [4]. It specifically deals with layer 2 forwarding, and makes no optimizations for IP.

III. Basic Rbridge Design

The basic design of an Rbridge consists of several parts:

- Within a campus, Rbridges execute a link state protocol such as IS-IS, so that all Rbridges know a path to each other Rbridge. IS-IS is a particularly good choice because of its flexible encoding that allows including new information (such as layer 2 addresses of reachable endnodes).
- On each link, a single Rbridge is elected DR (Designated Rbridge). That is the only Rbridge on that link that is allowed to learn the membership of endnodes on that link, and is the only Rbridge allowed to forward traffic onto that link that is destined for that link.
- The DR, like a bridge, learns which endnodes are located on its link by observing the source address of packets that have originated on that link.
- The Rbridge distributes the addresses of endnodes on its link in the link state protocol. This enables all Rbridges to know which Rbridge is the appropriate destination Rbridge for each endnode.
- The egress Rbridge from a link (usually the DR, but an optimization would allow another Rbridge to forward a packet off the link) encapsulates the packet with an additional header that contains, at the minimum, a hop

count, and preferably also a destination Rbridge identifier.

- Packets in transit are distinguished from originating packets, since they contain the encapsulation header. Therefore, there is no confusion between packets originating on a link and packets transiting that link; the DR will know not to assume that the source of a transit packet does not reside on the link from which the packet was received.
- Rbridges additionally calculate a spanning tree. This is for the purpose of delivering layer 2 multicast packets, and packets to unknown destinations. There is no need to implement an additional protocol in order to calculate a spanning tree, given that the Rbridges have a link state database.
- When packets are to be sent through the spanning tree, the encapsulation header specifies a value such as destination Rbridge ID=0. The packet is forwarded through the spanning tree and each DR (in addition to forwarding it through the spanning tree), removes the encapsulation header in order to forward the packet onto the DR's link.

This design can be thought of as performing the functionality of a bridge to endnodes, i.e., transparently interconnecting links, but it avoids the disadvantages of bridges. Since the transit packets are routed, with a header that contains a hop count, it is safe to have temporary loops. Packets with a specified destination Rbridge will not proliferate (as bridged packets would) during a temporary loop, and they will quickly be discarded due to the hop count. The hop count can even be set to be exact, unlike a hop count written into the header by an endnode, since the Rbridge can calculate the number of hops necessary to reach the destination. So, ignoring for the moment packets that must be sent through the spanning tree, Rbridge routing enjoys the following advantages over 802-style bridging:

- packets travel via an optimal path
- during temporary loops, packets do not proliferate
- packets have a hop count
- temporary loops are not a problem, so routing changes can be made instantaneously based on local information, safely.

A. The encapsulation header

The goals of the encapsulation header are:

- Allow Rbridges to differentiate packets originated by an endnode from transit packets
- Include a hop count
- Be compatible with bridges on the path between Rbridges

If we want bridges to coexist with Rbridges, so that a bridge might be on the path between two Rbridges, the packet must still contain what looks like an ordinary layer 2 header, so that bridges will be able to forward it.

The way to accomplish this is to use something in the outer layer 2 header that can be recognized by Rbridges as an encapsulated packet. The most straightforward would be a new protocol type (or SAP), that would mean "Rbridged encapsulated packet", which we'll call the Rbtype protocol type (or SAP). An Rbridged transit packet consists of an otherwise normal layer 2 header with protocol type Rbtype, followed by the encapsulation information including the hop count, followed by the original packet as transmitted by the source. When forwarding to the destination, the encapsulation header is removed, so that the goal of transparency to endnodes is accomplished. The destination will see the packet as transmitted by the source.

The layer 2 source and destination in the outer header should be the transmitting and receiving Rbridge. It is safe for bridges to learn layer 2 addresses within the bridged spanning tree terminated by Rbridges, since that mini-LAN is a normal bridged topology in which packets travel on a spanning tree. It would not be safe for the layer 2 source address to be the original source endnode's layer 2 address, because packets are not routed along a spanning tree throughout the campus, and therefore are injected into the mini-LAN from different directions. Therefore, bridges in a mini-LAN would see packets from a MAC address appear from different directions, if a MAC address outside the mini-LAN appears as a source in the outer header. This will confuse bridges about that MAC's location, and they may filter packets destined for that address.

The addresses in the outer header must be MAC addresses local to the mini-LAN, to avoid this problem. This outer header is rewritten on an Rbridge-hop by Rbridge-hop basis.

After the outer header is additional information of use to Rbridges. This includes the hop count, which would be decremented on each Rbridge hop. And although theoretically it might be avoided, it is desirable to also include the destination Rbridge address, or in the case of the destination's location being unknown, an indicator (such as destination Rbridge address specified as "0") that the packet is to be sent through a spanning tree.

In theory it is not necessary to specify the destination Rbridge, since all Rbridges would know how to forward to the known layer 2 destinations within the campus. However, it is safer to include it, to avoid loops in transition cases where some Rbridges know the destination Rbridge and others don't, and think the packet must be flooded, or some Rbridges disagree about which destination Rbridge owns the destination. This would not be too bad, though, because there is the hop count.

Another possibility for avoiding inclusion of the destination Rbridge is to include enough information to distinguish flooded packets from packets to presumed known destinations. This could be done by using a second reserved protocol type, say Rbflood.

The original packet is preserved after the encapsulation header so that the packet can be received transparently by the destination, without evidence of it having been handled by Rbridges.

B. *Temporary Loops*

1) *Caused by repeater or bridge*

If a component such as a repeater or bridge came up, it is possible that two links become merged. This could result in there temporarily being two DRs on a link. A DR will not be able to distinguish a packet from remote source S that has been decapsulated and injected by the other DR, from a packet originated by S. This will cause the DRs' learning to be faulty, and might introduce loops that are not protected by the hop count (because the hop count is removed when a DR decapsulates a packet onto the link).

This is likely to be a rare event, and more quickly detected and corrected because it is link-local. It will take less time for link-specific knowledge to converge than global knowledge, which is required for the bridge spanning tree algorithm to converge.

2) *Flooded Packets*

Some packets need to be flooded through the Rbridged campus along a spanning tree. Packets that need to be flooded are packets for destinations whose location is unknown, or packets with layer 2 multicast addresses. Just as unicast routing can have temporary loops, since a distributed algorithm cannot have all nodes instantaneously switch to a new topology, the spanning tree might temporarily have loops. With unicast routing loops, packets will not proliferate. With a spanning tree loop, packets will be duplicated.

However, the Rbridge spanning tree is far less dangerous than the 802-bridge spanning tree, because the encapsulation header contains a hop count. The Rbridge that injects the spanning tree packet into the Rbridge cloud can calculate the minimal hop count necessary for the packet. Additionally, for each port, the Rbridge can calculate a different hop count (in case leaves on one port are further away than leaves on another port).

Because of the hop count, spanning tree loops will, in practice, be unlikely to cause much harm. However, Rbridges can add additional conservative measures to prevent even the limited proliferation. As with regular bridges, they can impose a timer before starting to forward flooded packets onto new links. And given that they have a link state database, they can even calculate whether forwarding flooded packets onto a new link might cause a temporary loop.

Note that this section is only concerned with messages that must be flooded. Messages that are directed to a known destination location will not have any danger of being proliferated during temporary loops.

So in all cases Rbridges are far safer than regular bridges. They are not, however, as safe as CLNP level 1 routers, since CLNP level 1 routers would never need to flood packets to unknown destinations. But Rbridges work without requiring the ES-IS protocol, which only exists for CLNP.

IV. Rbridging IP

The basic design of the Rbridge, presented in section 1, can bridge layer 2 packets, but using optimal paths within the campus. If this is all they did, it would support IP. The campus would appear to IP to be a single LAN. However, there is one case that would not work. If the Rbridge only used layer 2 addresses, it would fail to interconnect two IP nodes within the campus if they resided on dissimilar layer 2 links, for instance ones with dissimilar addresses.

A. *The Dissimilar Layer 2 Address Issue*

Suppose source IP node S resides on a link with a different layer 2 address structure than destination IP node D. Since the Rbridged campus appears to be a single IP subnet, S will assume D is a neighbor, and issue an ARP. Unfortunately, the layer 2 address in D's reply will not be understandable to S, and not be expressible in the layer 2 header when S attempts to forward to B.

To support this case (S and D are IP nodes residing on links with incompatible layer 2 addresses), Rbridges reply to ARP queries, if necessary, with the layer 2 address of an Rbridge (see next section).

B. *Handling ARPs*

Let's say that source IP node S is on a link with DR R1, and target node D is on a link with DR R2. The goal is that when S does an ARP request for D, if the Rbridges already have learned about D, that R1 can reply with an ARP request to S, informing S of D's layer 2 address, without needing to flood the ARP request to other links.

We'd also like to support the case where S and D have incompatible layer 2 addresses. In this case R1 will reply to the ARP request with R1's layer 2 address.

How do the Rbridges learn the ARP information?

Let's assume that S wishes to speak to D. S issues an ARP request. S's DR, R1, replaces the source S in the ARP query with its own address, and remembers (S,D) so that, when it receives the ARP reply from D, it sends an ARP reply to S.

Each Rbridge R2, in addition to forwarding the flooded ARP request through the spanning tree, sends an ARP query on its own LAN, with itself (R2) as source, remembering that if it receives a reply from D it must send an ARP reply to R1.

When R2 (the DR on D's link) receives an ARP reply ("my layer 2 address is d") from D, R2 sends an ARP reply to R1, and also reports ownership of (D,d) in its link state information. The other Rbridges will now know the location of IP destination D, and the associated layer 2 address d.

To avoid a denial of service attack by having S issue too many ARP queries (which result in flooded packets and a lot of processing by Rbridges), the R1 will remember recent ARP queries, and refuse to issue another ARP query for D for some time.

If a second node, S2, on R1's link, issues an ARP query for D between the time S issued its query and D's reply is returned, R1 does not flood an ARP query. Instead, R1 remembers that an ARP reply from D (triggered by S's ARP query) should be sent to both S and S2.

In the case where D's layer 2 address is incompatible with the querying node S, S's DR (R1) replies to the ARP with R1's layer 2 address.

This design has the following properties:

- ARP queries will not need to be flooded once the Rbridges learn the location of the target IP node.
- IP will work even if the source and destination within the campus reside on links with incompatible layer 2 address types.

So, when an IP source emits an ARP, it will either be told the true layer 2 address of the destination, or the layer 2 address of its own Rbridge, depending on whether the destination's layer 2 address is compatible with the source's layer 2 address.

It might be conceptually simpler to have the Rbridge always respond to an ARP query with its own layer 2 address. The reason for using the destination's layer 2 address when possible (when the layer 2 address is compatible) is so that the source IP node's ARP cache will not need to change when the local Rbridge goes down and a different DR is elected.

An alternative design could use a logical layer 2 address for the Rbridge, say X. In this alternate design all IP endnode ARP caches would indicate X as the layer 2 address of all destinations.

There might be true bridges mixed in with the Rbridges (and transparent to the Rbridges, just as bridges are transparent to routers). Therefore, X must not be used as a source address, so that its location will not be learned by bridges.

The only disadvantage of this approach is that all packets to X would need to be flooded (within the very small spanning tree created by true bridges on the link between Rbridges).

Using the destination's true layer 2 address, when possible, avoids this slight suboptimality.

In the case of an endnode directly connected to an Rbridge port with a point-to-point links, none of these disadvantages apply, and there is no disadvantage to having the Rbridge always respond to ARP requests from S with the Rbridge's layer 2 address.

One other issue is a timing issue. It is possible that S will receive the ARP reply before R2's link state information has propagated. This case would be handled by a pure Rbridge (one that forwards based solely on layer 2 addresses) like a packet to an unknown destination; the packet will be flooded. For Rbridges forwarding IP packets based on the IP header (see section VI), this would cause the Rbridge that does not know the (IP, layer 2) binding to issue an ARP. It could store the data packet until the ARP reply was received, or flood the packet, or drop the packet.

C. Prompt Dead-Node Detection for IP

Rbridges can take advantage of the properties of IP in order to detect in a prompt manner when an IP node has moved or has died. With layer 2, there is no protocol in which a node is required to answer. However, with IP, if the DR knows that (D,d) resides on its LAN, the DR can periodically issue ARP queries for D, to reassure itself that D still resides on its LAN.

D. Optimizing the path

If the DR is always the egress and ingress point for the link, it is possible for packets to be two hops suboptimal. Given a particular source and destination, the ingress DR and the egress DR might each be one-hop suboptimal.

This suboptimality would only occur on shared media. Most topologies today really consist of switches and point-to-point links. If all the switches were Rbridges, and all links pt-to-pt, then there would be no such suboptimality.

However, if there are shared links (or switched links with the switches being bridges, so the link would appear to the Rbridge as if it were a shared link), there can be up to a 2-hop suboptimality.

The first hop's suboptimality can be avoided by having the Rbridges on the link implement a careful algorithm in which they calculate, for each destination, which of them is the optimal Rbridge for handling the packet. This can be done by calculating a Dijkstra tree with the link as the Root, and with a deterministic tie-breaker.

The suboptimality at the destination cannot be avoided, since it would be dangerous for any Rbridge other than the DR to inject a decapsulated packet onto the link. The DR would not be able to distinguish that from a packet that originated on that link, and the DR would falsely assume the source address in the packet resided on that link.

There is another form of route suboptimality. In IP, there might be several routers on the link, and endnodes on the link would choose a router, essentially at random, for forwarding packets to destinations that are not on that IP subnet (that do not share the same campus-wide prefix as the source IP node). IP already has the mechanism, if a router forwards a packet onto the same link from which it was received, for the router to send a Redirect message. This will avoid many cases of suboptimality.

E. What an Rbridge cannot do

There are some cases that an Rbridge cannot support:

- If there are links with incompatible layer 2 address formats, then an Rbridge would not be able to interconnect nodes on those two links unless those nodes were speaking IP.

- If the links have different packet sizes, an Rbridge would not be able to forward packets that are too large.

The first issue (incompatible layer 2 address formats) could in theory be supported if the addresses could be translated. Any two technologies that can be bridged can be Rbridged.

Rbridges could support interconnection within the campus of disparate layer 2 technologies provided that the source S and the endnode D resided on compatible link types, because the Rbridge encapsulates the packet and can send it to the destination Rbridge. Fragmentation and reassembly could, in theory, be supported by the encapsulation header.

F. Traffic Engineering

Rbridges are really routers, and therefore can do any sort of routing that routers do. In some networks, it is desirable to use MPLS in order to create special routing, for instance, to allow certain customers to have paths that have certain service guarantees. This is not incompatible with the Rbridge concept.

V. LESS GENERAL RBRIDGES

In this section we drop some of the generality of the Rbridge, and see what advantages it can give.

A. Transit links switch-switch only

If we assume that endnodes only exist on leaf links, and switches are aware of which ports are endnode ports, then there is no necessity to restrict endnode location learning. Each Rbridge is allowed to learn station locations for each of its ports on which an endnode might reside. We avoid the one-hop suboptimality trivially since there is only one Rbridge on each endnode link.

If we are *really* sure that an interswitch link will never be mistaken for an endnode link, then encapsulation would not be required for Rbridge learning. However, the encapsulation header includes a hop count, making forwarding during temporary loops safer. (Note that even without the hop count, Rbridge forwarding of packets for known destinations is safer than bridge forwarding, since Rbridges, like routers, will forward only in one direction.) However, for Rbridge flooded packets, without the encapsulation header, Rbridge forwarding would be as dangerous as bridge flooding.

VI. IP-SPECIALIZED RBRIDGE

Two requirements of the Rbridge design presented in the rest of the paper are implementation challenges for some switch hardware. These requirements are:

- The necessity to learn when forwarding data packets
- The necessity to encapsulate and decapsulate packets

These requirements can be avoided by dropping the goal of supporting anything other than IP. In this section we will assume all packets are IP packets.

A. Avoiding Encapsulation

There are three reasons for the encapsulation header: to have a hop count for safety, to distinguish transit packets from endnode-originated packets, and to distinguish packets to be flooded from those to known destinations.

Note that we will assume that IP packets also contain a layer 2 header. What we are avoiding is the use of an *additional* layer 2 header.

1) Hop Count

We will not need the encapsulation header for carrying a hop count, since the IP header contains a hop count. Rbridges can decrement the hop count in the IP header.

Some ISP customers consider an apparently small hop count across an ISP as superior service, and having Rbridges decrement the IP header's hop count would mean that the customers would see the Rbridge hops as IP hops.

One could argue (and be quite correct), that this is a completely false assumption on the part of the customer. An Rbridge hop (or a bridged hop) is no better than a router hop. Better service should be measured by metrics such as delay, bandwidth, and reliability, not by perceived numbers of hops. However, it is often politic to give the customers what they want rather than argue with them. So having Rbridges decrement the IP header hop count might be perceived as a disadvantage with some customers.

2) Transit Packets

Since we are assuming IP packets, it is not necessary for the original layer 2 header to preserved. Therefore, any information necessary for Rbridge forwarding can be carried in the layer 2 header.

To distinguish transit packets, we can replace the protocol type (which would indicate IP) by a protocol type indicating that it is an Rbridged transit IP packet.

The layer 2 source and destination should be replaced, at each hop, by the transmitting and receiving Rbridge on that Rbridge hop.

Although this might appear to be as much work as encapsulation, this is what routers do (rewrite the layer 2 header on each hop). But it avoids the encapsulation issues of requiring an *additional* layer 2 header, which might violate the maximum packet size.

At the final hop, the final Rbridge replaces the protocol type to indicate to the destination that it is an IP packet.

3) Flooded packets

As before, flooded packets can be distinguished by using a different reserved protocol type.

B. Avoiding Data Packet Learning

Some switch hardware is optimized for fast forwarding of data packets, and it is not possible for it to do anything other than forward. In particular, it cannot learn source addresses from data packets.

This form of switch is, of course, only used as a router, since bridges must learn based on data packets. But this requirement of bridges is because it is not possible to assume any sort of protocol by the endnodes.

If, however, we assume that all endnodes are only issuing IP packets, or associated control packets (such as ARP packets), then we can design an Rbridge that does need to learn from data packets.

If Rbridges will not learn IP destinations based on data packets, then they must learn them through ARP replies or link state information.

1) D unknown by endnodes and Rbridges

The description of how ARP requests and replies are handled is in section IV.B. The ARP requests and replies can be handled by the control plane.

Let's assume that destination endnode D is unknown. Source S wishes to speak to D. S issues an ARP request. The ARP request is not a data packet, so it can be dealt with in the slow path (the control plane). The first Rbridge, R1, replaces the protocol type with a protocol type indicating "flooded ARP", replaces the source S with its own address, and remembers that, when it receives the reply from D, it must send an ARP reply to S.

Flooded ARPs can also be handled by the control plane because they can be recognized based on the protocol type. Each Rbridge R2, in addition to forwarding the flooded ARP request through the spanning tree, sends an ARP query on its own LAN, with itself (R2) as source, remembering that if it receives a reply from D it must send an ARP reply to R1.

When an ARP reply is received by D, D will send the ARP reply to R2. Then R2 sends an ARP reply to R1, and also reports ownership of IP destination D, and associated layer 2 address, in its link state information. The other Rbridges will now quickly learn the location of IP destination D, and the associated layer 2 address (so they can respond locally to ARPs).

2) D known by Rbridges, not by S

In this case, when source endnode S wants to talk to D, S will issue an ARP query. The first Rbridge, R1, has learned, based on link state information issued by R2, where D resides, and also what its layer 2 address is. R2 then does not forward the ARP reply, and instead answers with an ARP reply on behalf of D.

3) D known by S and not by Rbridges

Endnode S might know D's layer 2 address, and yet D might be unknown to the Rbridges. This might occur because S's ARP cache might have a longer retention time than Rbridge caches. Or maybe D's Rbridge has been restarted and D is no longer included in its link state information.

So in this case an Rbridge will see a packet for an unknown IP destination address (but within the campus Rbridged prefix). Most likely this will be the first Rbridge. However, in a transition case where link state information has partially propagated, it might be a transit Rbridge. But the Rbridge will behave the same way in either case.

The Rbridge will drop the unknown IP destination packet, and instead issue an ARP query, with itself as source. This will cause a flooded ARP query, with each Rbridge issuing an ARP query on its own link. The Rbridge on D's link will receive an ARP reply, and inform the other Rbridges, through the link state flooding.

To avoid causing a lot of flooded ARP messages, Rbridges should remember recent unknown IP destinations that have caused an ARP flood, and not issue another one for some amount of time.

VII. Summary

The Rbridge design achieves the transparency of bridging without the disadvantages. It achieves the ability to create a campus that looks like a single link.

The campus can include links with different layer 2 technologies. An Rbridge would not be able to allow two nodes to speak at layer 2, if they reside on incompatible link types; however, the Rbridge would enable those nodes to communicate if the nodes were speaking IP.

Rbridges have great advantages over bridging. They allow optimal paths and path splitting. They need not be conservative about creating temporary loops because packets do not proliferate, and there is a hop count. The hop count with Rbridges is set by the source Rbridge, which is armed with link state information, so that the source Rbridge can calculate the minimum necessary hop count. This means that during temporary loops unicast packets (packets to known destinations) will be removed more quickly than packets in a traditional layer 3 network.

For flooded packets (layer 2 multicast or packets to unknown destinations), Rbridges, like regular bridges, may duplicate packets during temporary loops. However, with Rbridges there is a hop count, the hop count can be set to be the minimum necessary, and additionally the Rbridge can use global information to make conservative temporary loop-avoidance decisions, so its loop-avoidance behavior will be more timely and accurate than anything a true bridge could do.

For IP packets, the Rbridge has the additional advantage that ARP packets need not be flooded, and instead can usually be answered by the source Rbridge. Additionally, the location of IP endnodes can be kept promptly up to date by using local link mechanisms such as ARP queries.

Rbridges could perform better for IP if IP included a mechanism such as ES-IS, that was universally implemented by all endnodes. But Rbridges achieve almost as good a result with no assumptions on IP behavior other than the classic IP design.

If it is reasonable to *only* support IP endnodes, the overhead of encapsulation and learning from data packets can be avoided.

ACKNOWLEDGMENTS

I would like to thank Ivy Hsu, Dino Farinacci, Stewart Bryant, John Ioannidis, Charlie Kaufman, and the anonymous reviewer, for giving helpful feedback.

VIII.

References --

[1] Boston Globe, November 26, 2002.

[2] Callon, R., "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments", RFC 1195, December 1990.

[3] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[4] Garcia, R., Duato, J., Silla, F., "LSOM: A Link State Protocol over MAC Addresses for Metropolitcan Backbones Using Optical Ethernet Switches", Proceedings of the second IEEE International Symposium on Network Computation and Applications (NCA '03).

[5] ISO, "Protocol for Providing the OSI Connectionless-Mode Network Service", ISO 8473.

[6] ISO, "End System to intermediate system routeing information exchange protocol for use in conjunction with th eprotocol for providing the connectionless-mode network service", ISO 9542.

[7] ISO, "Information technology -- Telecommunications and information exchange between systems -- Intermediate system to Intermediate system intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473).

[8] Lui, K., Lee, W., Nahrstedt, K., "STAR: A Transparent Spanning Tree Bridge Protocol with Alternate Routing", ACM Sigcomm Computer Communications Review, July 2002.

[9] Perlman, R., "A Protocol for Distributed Computation of a Spanning Tree in an Extended LAN", 9th Data Communications Symposium, Vancouver, 1985.

[10] Rodeheffer, T., Thekkath, C., and Anderson, D., "SmartBridge: A Scalable Bridge Architecture", ACM Sigcomm 2000.

[11] Schroeder, M., Birrell, A., Burrows, M., Murray, H., Needham, R., Rodeheffer, T., Satterthwaite, E., Thacker, C., "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links", IEEE Journal on Selected Areas in Communication, October 1991.

[12] IEEE 802.1w-2001, IEEE Standard for Information technology---Telecommunications and information exchange between systems---Local and metropolitan area networks---Common specifications Part 3: Media Access Control (MAC) Bridges---Amendment 2---Rapid Reconfiguration [Amendment to IEEE Std 802.1D, 1998 Edition (ISO/IEC 15802-3:1998) and IEEE Std 802.1t-2001]