

## Filtering with MAC Addresses

Michael J. Martin

You may recall from my previous articles on performing network services audits that the media access control (MAC) address of the host is one of the critical data points collected as part of the network inventory. The MAC is one of the best unique-node values to filter and verify connectivity because it is not easily changeable. So when failures or changes occur, it's a good bet that some investigation is in order. The phrase "not easily changeable" may be ringing a bell for some of you. You're probably saying to yourself, "He's crazy: hardware MAC addresses are not changeable."

Each Ethernet based node and Ethernet switch/bridge port has a globally unique unchangeable MAC address stored locally on the network interface card, or on the on-board network controller used in network switches or system boards with built-in Ethernet interfaces. And each NIC/controller also has a unique hardwired OUI-derived MAC address, commonly known as the burned in address (BIA) MAC. Even so, it is possible for a node to use a user-defined MAC address to mask the BIA address and take the BIA's place as the MAC address used for sending and receiving Ethernet frames. So although the MAC is a pretty good bet to filter and track on, it's not foolproof. However, if a system is compromised or if you have rouge system administrator on your hands, all bets are off.

Then of course, there are all of those "soft MACs." These virtual MAC addresses are used by Layer 3 switch VLAN interfaces, by Hot Standby Router Protocol (HSRP) interfaces and by the various ether-channel NIC bonding and host availability schemes. (How else would nodes be able to forward packets to these interfaces?) While these "soft" addresses do not mask the BIAs, they do come up as part of the results of a network inventory and will be seen as active hosts on that network. The issue for auditors is that these "dual-identity" nodes now appear twice in the inventory. They are discovered, in both cases, with a different MAC and IP address.

Both the former and latter conditions are great examples of why it is so important to verify your inventory and audit findings with some kind of baseline data. Then establish Layer 2 and Layer 3 access control measures and continuous auditing. In this series we'll focus on Layer 2 access controls. Although we have not covered service auditing yet, we have talked about the network inventory portion of network auditing, and it seems a good time to discuss some strategies for controlling host network access using MAC address filtering. There are a number of uses for MAC filtering. The most common use for MAC filtering, sadly, is reactive. In response to certain types of network attacks and viruses, ingress MAC filtering can be applied to filter attacking hosts using bogus IP source addresses. (Once you have traced the source back to the origin network, you'll often find the users launching these attacks were unaware because the attack is the result of a virus.) The far better use for MAC filtering is to proactively control which nodes can access the network by implementing MAC-based filtering on Layer 3 interfaces and Layer 2 switch and bridge ports. Granted, this approach is a bit hardcore for "open environments" where the users are transient and mobile. This kind of filtering is a great method for controlling network access in data centers and in closed wireless environments, or in any network environment where the hosts accessing the network are consistent, where there is very little turnover and the introduction of new hosts should not take place without notification.

Let's talk about where and how we implement MAC filtering. Where this level of control should be best applied, in the end, is really the judgment of the network and security administrators. (Network administrators should not implement this kind of security approach without approval from their respective powers that be.) Now, let's look at how Cisco IOS-based routers and switches support MAC filtering...well sort of. On IOS based switches, MAC filtering is supported as an "inbound" port-level filter. However, while Layer 3 protocol filtering is supported on IOS based router interfaces, Layer 2 filtering is not. In order to support Layer 2 filtering on IOS routers, the router must be configured to support transparent bridging.

# Filtering with MAC Addresses

Michael J. Martin

## Collision Domains And Transparent Bridging

So, what is a collision domain or transparent bridge? Well, if you know what a network switch is, you're halfway there. In fact, you probably know what they both are, but maybe you forgot because these terms and concepts have gone by the wayside due to today's high-speed switching. Transparent bridges were the predecessors of today's network switches and both operate on the same premise. A transparent bridge is an interconnection device that joins two or more Layer 2 collision-domains. A collision-domain is a segment of an Ethernet LAN. Nodes connected to the segment "share" the available bandwidth to transmit Ethernet frames. It's all clear now, right?

The name "transparent bridge" refers to its operation in relation to the nodes connected to the Layer 2 segment. The nodes connected to the segment are unaware that the bridge (or bridges) are connected to the segment and forwarding frames between the different collision-domains. There are two types of bridges: translation and interconnection. Translation bridges translate different Layer 2 protocols so they can exchange frames without the need of a Layer 3 device. Common translation bridges are FDDI to Ethernet; Serial to Ethernet; and Token Ring to Ethernet. The basic operation of a translation bridge is to extract the data payload and re-frame it so it can be delivered. Interconnection bridges "extend" the length of the LAN by interconnecting two or more collision-domains. Transparent bridges all work under some basic operational premises:

- First, the bridge's interfaces run in promiscuous mode examining all of the frames transmitted across the connected segments. The bridge stores the frames in memory buffers to be forwarded on to the other segments. (The most common transparent bridge is a "store-and-forward" bridge.) The bridge determines where to forward the frames by building a forwarding table associating the MAC addresses of the nodes on the network with the bridge port the MAC was learned on. When the bridge sees a frame transmitted on the wire, it stores it, and performs a lookup against the bridge's forwarding database. If the destination address is on the same port the frame was seen on, it discards it. If the bridge has an entry for the destination node off of an adjacent port, it forwards the packet out of the associated bridge interface. If the bridge does not have an entry for the destination node, the frame is forwarded out of each of the bridge's ports, except for the port the frame came in on. Once the bridge has learned all of the active nodes, the bridge only forwards frames that need to be forwarded, unlike a repeater that forwards every frame that it sees.
- The second is that all frames are forwarded by the bridge unmodified. When a typical Ethernet node transmits a frame on the network, it places its own MAC in the frame's source address field. Bridges, however, do not place their own MAC addresses in the source address field of the frames they forward. As far as the source and destination are concerned, the frame was directly transmitted and never went through a bridge (hence the name transparent).
- Third, bridges operate under the same media access rules as all of the other nodes connected to the segments that are interconnecting. Bridges need to wait for access to the wire before they transmit. This means that a bridge interconnecting two congested collision-domains could introduce more latency than if the two nodes were connected to the LAN without the bridge between them. As a result, when bridges were first introduced, the potential introduction of additional latency made bridges very unpopular with network designers, even though they were developed to provide a means for extending Layer 2 network segments beyond their physical length and node density limitations. Which brings us back to collision-domains.

We understand that a collision-domain is an Ethernet-shared media segment on which multiple nodes are connected. The bandwidth of the collision-domain is shared between the connected hosts. While this definition is accurate, it really doesn't provide insight into what a collision-domain is aside from a networking construct. To get a better understanding, we need to quickly review how Ethernet

## Filtering with MAC Addresses

Michael J. Martin

transmits data. Ethernet uses Carrier Sense Multiple Access/Collision Detection (CSMA/CD) as its transmission contention protocol. The idea is quite simple: Multiple nodes are connected to a common transmission medium, and these nodes take turns transmitting data. Only one host can transmit data at one time. If two hosts transmit data at the same time a collision occurs, and both hosts stop their transmissions, wait and attempt to retransmit when the line is clear. The "Collision Detection" portion of the contention protocol (which behaves as a primitive flow-control mechanism making it possible for all of the hosts to share the line) is a component of the Ethernet protocol that is dependent on the proper construction of the collision-domain.

Now, the physical collision-domain is defined by three factors: physical length of the segment, the length of transmission protocol's bit period (which is different for each flavor of Ethernet) and propagation time (also different for each flavor of Ethernet). Most engineers focus on the maximum cable length between the end station and the repeater (which is the basis of all of the "X" Base-T Ethernet implementations). The 10/100 Base-T standard calls for the distance between the end station and the repeater to be no more than 100 meters. However, this is misleading because the actual measurement of collision-domain is based on bit periods, not cable length. This is because the X Base-T connections are made up of more than just cable, there are also repeaters and the DTEs themselves, each of which has a bit-period latency. It is the total sum of the physical components' bit-period latency that makes up the actual size of the collision-domain.

The bit-period value is the time it takes to transmit one bit of data. This value is equal to the transmission rate of the medium. For example, the bit period for 10-Base T Ethernet is 100 nanoseconds, and for 100 Base-T it is 10 nanoseconds. Propagation time is the amount of time it takes for a DTE on one end of the cable to transmit a 64 byte (512 bit period) frame to a DTE on the other end of the cable. The propagation delay of 10 Base-T Ethernet is 51,200 nanoseconds; for 100 Base-T it is 5120 nanoseconds. CSMA/CD works because the two nodes transmitting data will simultaneously see the transmission of the other, halt their transmission and institute the collision-detection process. The total size of the collision-domain cannot exceed the length of time it takes for this function to happen before the stations complete their data transmissions. Otherwise, the transmitting stations will not detect the collision in time. The maximum size of a collision-domain is the addition of the propagation delay, or in Ethernet parlance, "slot-time," and the "space" of time a transmitting hosts places between each frame, known as the interframe gap, which is 96 in bit time. Therefore, the maximum the collision-domain length for 10 Base-T Ethernet is 60,800 nanoseconds, and 6080 nanoseconds for 100 Base-T.

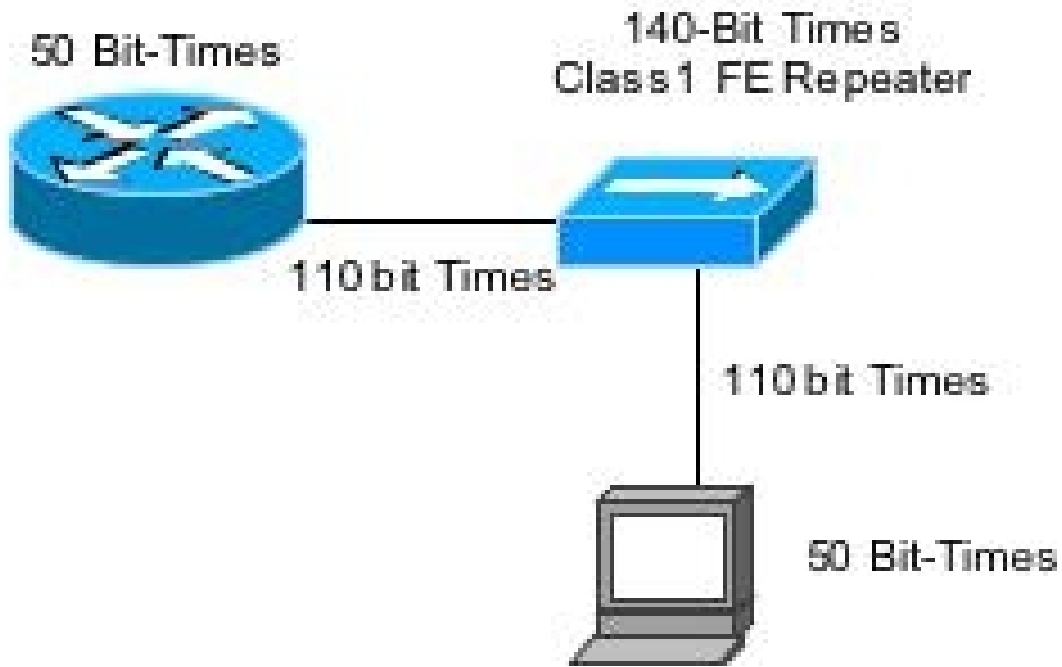
So you can see, the size the collision-domain is really more a measurement of time then physical length. From a design perspective, each collision-domain is the perspective of latency. The amount of tolerable latency between 10 Base-T and 100 Base-T is drastically different. This results in relatively large 10 Base-T segments and quite small 100 Base-T segments. Here are some standard bit-time delay values for 10 and 100 Base-T Ethernet:

- 10-Base T Repeater 100 Bit-Times
- 100-Base T, Class 1 Repeater 140 Bit-Times
- 100-Base T, Class 2 Repeater 90 Bit-Times
- 10/100 Base DTE 50 Bit-Times
- 1 Meter UTP Cable 1.1 Bit-Times

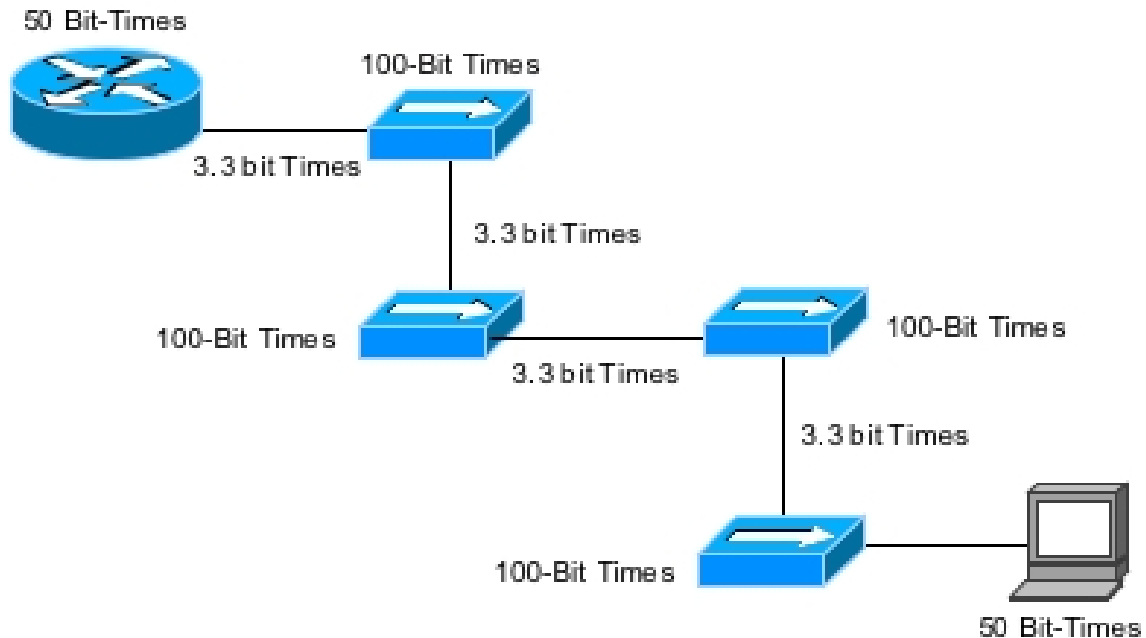
The ideal design of a shared-media segment is around 520 bit-times, supporting a depth of no more four repeaters deep for a 10 Base-T collision domain.

## Filtering with MAC Addresses

Michael J. Martin



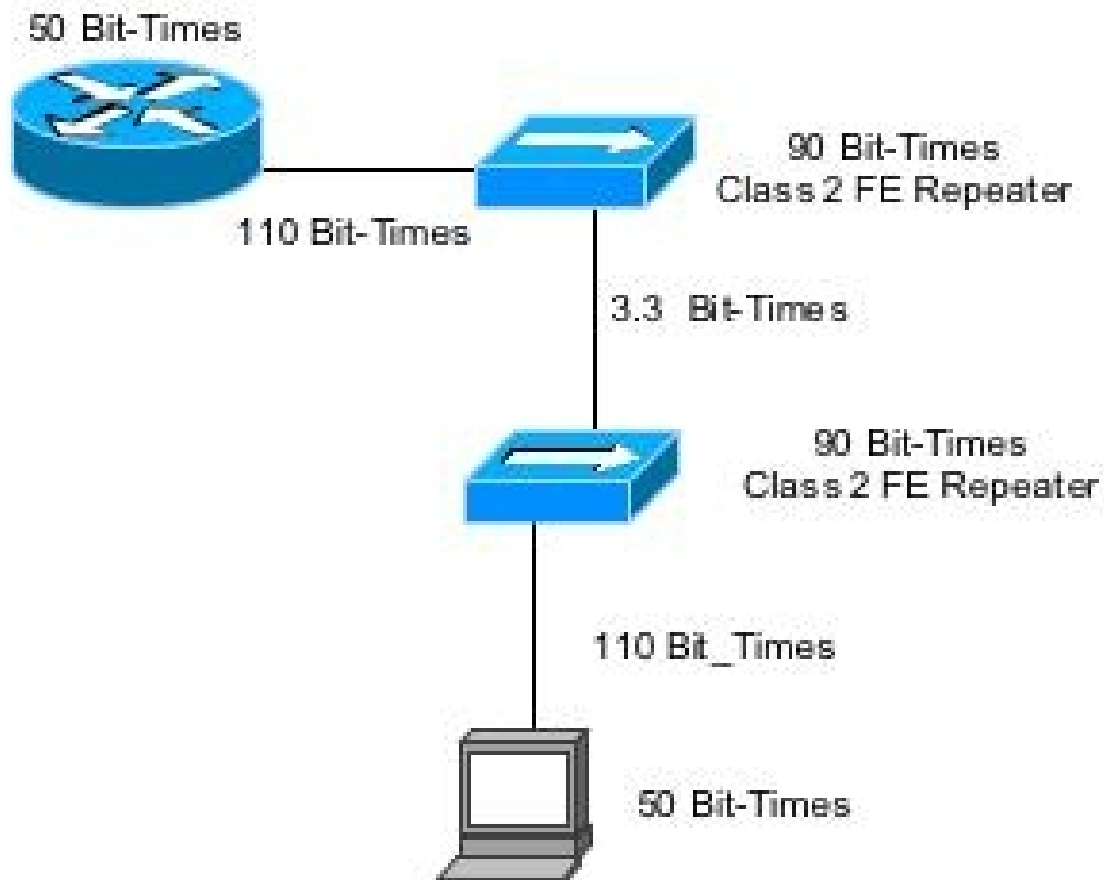
In a Fast Ethernet environment, the length is the same, but the actual supported density is far less. There are two options. In a Class 1 repeater configuration there can be only one repeater:



A collision domain constructed of Class 2 repeaters can have only two repeaters:

## Filtering with MAC Addresses

Michael J. Martin



So while transparent bridges existed for 10 Base-T, it was 100 Base-T that drove wider use to accommodate the reduced collision-domain segment sizes. By using bridges, the length (i.e., number of nodes that can be supported on the segment) of Layer 2 segments could be extended beyond the limitations of the Class 1 and Class 2 Fast Ethernet repeaters. This had the opposite effect in 100 Base-T networks than in 10 Base-T networks -- an increase in network performance due to the reduced number nodes in the collision-domains.

The alternative to implementing bridges between the collision-domain segments was the introduction of Layer 3 routing between collision-domain segments. This addressed the issue of Layer 2 segment size by increasing the number of Layer 3 segments. It also introduced greater complexity and additional latency at a much higher cost than a bridged solution. It was the introduction of LAN switches (by Klapana, Intel, Rapid City, and others) that exploited the true potential of bridging (and to a great extent the adoption of 100 Base-TX) by moving the concept from interconnecting collision-domains with bridges to interconnecting nodes. They scaled bridge density, making each port its own Layer 2 collision domain to which a single node, or small group of nodes, could uplink. That provided full bandwidth capability to the node level, along with opening the door to full-duplex transmissions and removing node density and segment-length limitations.

Now that you know more about the types of bridges, their basic operation and collision-domains, in my next article we'll move onto the joys to Spanning Tree and deploying bridges.

Before we sign off, there is one little thing that needs to be taken care of. I was asked what to do if access to the switches and routers was not available when conducting the network inventory. I

## Filtering with MAC Addresses

Michael J. Martin

thought this was a fair question, so I came with an alternative (assuming you have LAN access to the network you wish to build an inventory for). The issue is the ARPdump file. All the other data points can be collected remotely from the network being audited. That is why the Layer 3 switch or router is best suited for this kind of collection. To generate an ARPdump data file without accessing the router, you can connect a Unix/Linux system to the local subnet you wish to audit and use a nifty tool called arping. Arping does just what it says; it sends ARP request packets for specific MAC addresses and measures the responses. To audit a whole network segment, as root, run this command syntax:

```
Trinity:~ root# /usr/local/bin/arping -rR -c 1 ff:ff:ff:ff:ff:ff | awk
'{print
$2 " " $1}' | sed 's://1' | sed 's://2' | sed 's://3' | sed 's://./g'
172.30.71.4 000f.f713.08c1
172.30.71.7 0009.e8e2.a480
172.30.71.140 000d.9327.74b7
172.30.71.26 0003.93e7.f08b
172.30.71.1 000c.ce05.da77
172.30.71.25 0011.2400.06a6
172.30.71.8 0011.242b.28b5
```

Arping sends out a broadcast (ff:ff:ff:ff:ff:ff) ARP request, and all of the nodes connected to the segment respond. The syntax following the arping command formats the arping output into a format compatible with the ouilookup.sh/ouiresolv.pl scripts. To generate the arpresolve.txt file you need, just direct the arping command output to a file:

```
Trinity:~ root# /usr/local/net/arping -rR -c 1 ff:ff:ff:ff:ff:ff | awk '{print
$2 " " $1}' | sed 's://1' | sed 's://2' | sed 's://3' | sed 's://./g' >
arpresolve.txt
Trinity: #
```

Then you can run the ouiresolv.pl PERL script and translate the OUIs:

```
Trinity:/Users/mmartin/Desktop/AM-DC_Audit/bin/old-scripts root# perl
ouiresolv.pl
looking up OIDs.
172.30.71.25 -> Apple Computer
172.30.71.4 -> Cisco Systems
172.30.71.1 -> Cisco Systems
172.30.71.140 -> Apple Computer
172.30.71.7 -> Cisco Systems
172.30.71.26 -> Apple Computer, Inc.
```

As I have indicated before, this is a slightly more involved process because it requires you to have local network access. However, it does provide a workaround for environments where access to switches and routers is not available. In the meantime, if you have questions, comments, or article suggestions, please e-mail us.