

A Unique Identifier for Network Hosts

Mark R. Meiss
Indiana University

Abstract

Common problems in data network management include uniquely identifying network equipment and sharing network information with output parties without compromising network security. This paper presents a simple algorithm for constructing a 128-bit string that uniquely identifies a network host and can be shared without providing any addresses associated with the host.

1 The Need for a Unique Identifier

A network management application often needs to store information about the topology of a network segment or at least what hosts and network equipment are present on that segment. The most obvious choices for such an identifier are a host's network layer address (e.g., IPv4 address) or its data link layer address (e.g., MAC address). However, there are serious drawbacks to either of these choices.

A network layer address, particularly an IPv4 or IPv6 address, is neither unique nor an accurate identifier. A device may have multiple IP addresses; consider the case of a router, a switch with multiple VLANs and management addresses, or a host with several network interface cards. IP addresses may change, especially in a network environment that includes DHCP. If the IP address of a network element has changed to the former address of another network element, the change may be undetectable. Finally, IP addresses cannot be shared outside an organization without revealing information about how to contact the host in question, leaving it more vulnerable to attack.

A data link layer address, such as a MAC address, is a promising alternative. Under normal circumstances, a piece of equipment with a globally valid

MAC address will not change its MAC address unless its network interface is replaced. Sharing a MAC address outside a network reveals less potentially damaging information; in general, an outside party cannot contact a host using its MAC address. However, an unscrupulous person could use this information to forge the source of network attacks or even word processing documents. We also have the same multiple address problem that we saw with IP addresses. Because every physical interface has a unique MAC address, it is not uncommon for a switch to have as many as 80 different MAC addresses.

It is tempting to select one IP address or MAC address out of all the addresses of a device and designate it as the “canonical address” of the device. Unfortunately, it is unclear how one would select this address in a logical way and deal with the other drawbacks to using IP and MAC addresses as unique identifiers.

There is also no standard way of using the Simple Network Management Protocol (SNMP) to query the manufacturer’s serial number for a device. When such information is available, the method of accessing it varies from manufacturer to manufacturer; every company’s equipment is a special case.

Since the more obvious methods of calculating a unique identifier turn out to be generally unsuitable, let us enumerate the traits that would be desirable for such an identifier if we had it.

The ideal unique identifier of a device:

1. can be calculated quickly;
2. will change when the hardware configuration of the device changes;
3. is guaranteed to be unique;
4. has a short, fixed length; and
5. can be shared outside an organization without revealing the device’s addresses.

The **uniQ**ue **ID**entifier (QID) detailed in the next section satisfies the first, fourth, and fifth of these requirements, and it has almost certain probability of satisfying the second and third as well.

2 The UniQue *ID*entifier (QID)

Before given the algorithm for determining the QID of a network host, it is useful to define a few notational conventions.

Given a network host H :

- Let L be an ordered list of the network interfaces on H . If H supports network management via SNMP, then a network interface is any interface with an entry in the device's *ifTable* [1] object, and the interfaces are arranged in order of increasing *ifIndex*. If H does not support network management, then a network interface is any physical network interface with a data link layer address, and the interfaces are arranged in topographical order of their data link layer addresses.
- Let I be an individual network interface from the list L .
- Let $M(I)$ be the physical address associated with I . In the case of a managed host, this is the value of *ifPhysAddress* for I . In the case of an unmanaged host, this is the data link layer address for I .
- Let B be a buffer of bytes that can grow to arbitrary size.
- Let $MD5(B)$ be the result of using the contents of B as input to the MD5 Message Digest Algorithm [2]. The output of this algorithm is a 128-bit string.
- Let Q be the QID of H .

To calculate Q :

1. Determine the list L .
2. Initialize the buffer B .
3. For every interface I in L , append $M(I)$ to B .
4. $Q \leftarrow MD5(B)$.

3 Requirements Met by the QID

As previously noted, the QID algorithm as described above absolutely fulfills some of our requirements and likely fulfills the others. This section provides support for this assertion.

First, the QID can be calculated quickly. For a host managed by SNMP, all that is required is to retrieve a single column of the *ifTable*, concatenate the results into a buffer, and execute the MD5 algorithm on the buffer. For an unmanaged host, the list of data link layer addresses will have to be determined in another way, such as querying the ARP tables of the routers to which the host is connected. It is also worth noting that the MD5 algorithm is available as a standard part of the Java class library.

Second, the QID is very likely to change when the hardware configuration changes. The statistical probability of two similar buffers of data link layer address producing the same MD5 hash is astronomically small, and the algorithm will be sensitive to new interfaces, removed interfaces, and changes in the order of interfaces. A changed QID will in practice be reliable and sufficient evidence for concluding that the physical makeup of a network device has changed as well.

Third, the QID is almost certainly likely to be unique. Although it is possible that a network contains two devices whose concatenated data link layer addresses map to the same MD5 hash, it is also extremely unlikely. This chance is small enough so that the QID can be used as a unique ID in a relational database table, for example.

Fourth, the QID does have a short, fixed length. It can be represented as a string of 16 octets, 32 hexadecimal digits, or 22 Base-64 digits in MIME encoding [3].

Fifth, it is possible to share the QID of a network device outside of an organization without providing any information about how to contact the device. The MD5 algorithm cannot be reversed, and even if it could, an outsider would know only data link layer addresses, and network addresses.

As an example of when it may be useful to maintain this level of security, consider a system in which a user from outside an organization needs to troubleshoot the data link layer path to a host in the local network. Using the QID algorithm, a system in the local network could provide the remote user with a list of switches traversed in the final router hop to the host without giving that user information about the network addresses of those switches. The remote user could then ask the network management application for

error data on those switches, using the QID as a key. We have thus enabled troubleshooting for users outside the organization without having to share either switch addresses or SNMP access.

4 Conclusion

The QID is a useful tool for assigning unique identifiers to network devices because of its sensitivity to hardware changes, its uniqueness, its fixed size, and its security. These properties make it especially useful as a key in network management databases, which are notoriously vulnerable to changes in network configurations.

To make the QID more accessible to the network management community, it would be useful if it were to become part of a standard MIB for SNMP-managed devices. Even in the absence of this support, it should ease development of network topology databases and secure troubleshooting applications.

5 Acknowledgements

Supported by the Data Network Services group of Indiana University and the Advanced Network Management Lab of the Pervasive Computing Laboratories at Indiana University.

References

- [1] K. McCloghrie and M. Rose, editors. *RFC 1213: Network Management Base for Network Management of TCP/IP-based internets: MIB-II*. IETF Network Working Group, 1991.
- [2] R. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm*. IETF Network Working Group, 1992.
- [3] N. Freed and N. Borenstein. *RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. IETF Network Working Group, 1996.

- [4] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *RFC 1157: A Simple Network Management Protocol (SNMP)*. IETF Network Working Group, 1990.
- [5] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1901: Introduction to Community-Based SNMPv2*. IETF Network Working Group, 1996.
- [6] J. Case, R. Mundy, D. Partain, and B. Stewart. *RFC 2570: Introduction to Version 3 of the Internet-standard Network Management Framework*. IETF Network Working Group, 1999.