OAM Functions: Error Reporting, Configuration, Management (ICMP, DHCP, NAT, SNMP): BRIEF VERSION

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Based in part upon slides of Prof. Raj Jain (OSU), S.Deering (Cisco), C. Huitema (Microsoft)
Overview

- Operations and Management (OAM)
- Error Reporting (ICMP); Tools: ping, traceroute
- Configuration: RARP, BOOTP, DHCP
- Address Management: DHCP, Private Addresses, NAT, RSIP
- Network Management: SNMP, RMON

Ref: Chap 5,6,9,20,23,30: Doug Comer textbook, Interconnections by Perlman
Reference Site: IETF NAT Working Group
Reference: RFC 2663: IP Network Address Translator (NAT) Terminology and Considerations: In HTML
Reading: RFC 3022: Traditional IP Network Address Translator (Traditional NAT):
ICMP Features

- Used by IP to send error and control messages
- Uses IP to send its messages
- Does not report errors on ICMP messages.
- ICMP message are not required on datagram checksum errors.
- ICMP reports error only on the first fragment
ICMP Message Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Header</td>
<td></td>
</tr>
<tr>
<td>Type of Message</td>
<td>8b</td>
</tr>
<tr>
<td>Error Code</td>
<td>8b</td>
</tr>
<tr>
<td>Checksum</td>
<td>16b</td>
</tr>
<tr>
<td>Parameters, if any</td>
<td>Var</td>
</tr>
<tr>
<td>Information</td>
<td>Var</td>
</tr>
</tbody>
</table>

- ICMP error messages normally include the IP header of the datagram that generated the error, plus at least 8 bytes following the IP header =>

**Typical ICMP message sizes = 70 bytes**
Sample ICMP Messages

- **Echo Request/Reply**: Used in ping

- **Source Quench**: Please slow down! I just dropped one of your datagrams.
  - Congestion control function: deprecated...

- **Time Exceeded**: Time to live field in one of your packets became zero.” or “Reassembly timer expired at the destination.

- **Fragmentation Required**: Datagram was longer than MTU and “No Fragment bit” was set.
  - Used in fragmentation/reassembly and path MTU detection
Sample ICMP Messages (Continued)

- **Address Mask Request/Reply**: What is the subnet mask on this net? Replied by “Address mask agent”
- **Redirect**: Send to router X instead of me.
  - *Configuration functions… Redirect used. Mask config handled by BOOTP/DHCP.*
- **Time Stamp Request/Reply**: used to find current time or RTT.
  - *Deprecated…*
### ICMP: Message Types Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Echo reply</td>
</tr>
<tr>
<td>3</td>
<td>Destination unreachable</td>
</tr>
<tr>
<td>4</td>
<td>Source quench</td>
</tr>
<tr>
<td>5</td>
<td>Redirect</td>
</tr>
<tr>
<td>8</td>
<td>Echo request</td>
</tr>
<tr>
<td>11</td>
<td>Time exceeded</td>
</tr>
<tr>
<td>12</td>
<td>Parameter unintelligible</td>
</tr>
<tr>
<td>13</td>
<td>Time-stamp request</td>
</tr>
<tr>
<td>14</td>
<td>Time-stamp reply</td>
</tr>
<tr>
<td>15</td>
<td>Information request</td>
</tr>
<tr>
<td>16</td>
<td>Information reply</td>
</tr>
<tr>
<td>17</td>
<td>Address mask request</td>
</tr>
<tr>
<td>18</td>
<td>Address mask reply</td>
</tr>
</tbody>
</table>
ICMP-based tools: **Ping**

- **Ping**: Used to test
  - destination reachability,
  - compute round trip time
  - count the # of hops to destination
  - may provide record route option.

- Ping failure does not guarantee unreachability. Firewalls may filter pings.
ICMP-based tools: *Traceroute*

- **Traceroute**: Exploit TTL and ICMP
  - Send the packet with time-to-live = 1 (hop)
  - The first router discards the packet and sends an ICMP “time-to-live exceeded message”
  - Send the packet with time-to-live = 2 (hops) etc…
  - Does not use optional features like record route
ICMP-based tools: *Path MTU Discovery*

- Send a large IP datagram with “Don’t fragment” bit set.
  - Failure to fragment at a link will result in ICMP message.
  - Later version of ICMP specifies MTU size in such ICMP messages.
- Reduce MSS until success (No ICMP message received)
Configuration: Issues

- Configuration: give protocols the parameters they need to operate
- Several things to configure… Eg scenario:
7 Things to configure…

- 1. End systems need Layer 3 address, names, masks
- 2. Router finds Layer 3 addresses of end systems
- 3. Router finds Layer 2 addresses of end systems
- 4. End systems find a (default) router, name server
- 5. End nodes on the same LAN discover that they can send directly to each other
- 6. End systems find the best router for exit traffic
- 7. End systems communicate on a router-less LAN

- Typically end systems only know their hardware (IEEE 802) address…
Method 1: Reverse ARP (RARP)

- H/w (MAC) address -> IP address mapping
  - End system broadcasts RARP request…
  - RARP server responds.
  - Once IP address is obtained, use “tftp” to get a boot image. Extra transaction!

- RARP design complex:
  - RARP server is a user process and maintains table for multiple hosts (/etc/ethers). Contrast: no ARP server
  - RARP needs a unique Ethernet frame type (0x8035) & works through a special kernel-level filter
  - Multiple RARP servers needed for reliability
    - RARP servers cannot be consolidated since RARP requests are broadcast => router cannot forward
  - After all this, you get only the L3 (IP) address
Method 2: BOOTP

- Runs over UDP/IP as a user process
  - IP software can broadcast (to 255.255.255.255) even if local IP address unknown => client broadcasts BOOTP request
  - Port number 67 for server and 68 for client (not an ephemeral port)
  - Delivers BOOTP reply to BOOTP client and not other UDP apps when reply is broadcast
  - Does not wake up other servers during broadcast reply
BOOTP (Continued)

- BOOTP requests/replies sent w/ DF bit set.
- Server can send reply via broadcast or unicast:
  - For unicast reply, BOOTP server knows the IP address, but the link layer address is not in the ARP cache
  - Note that the server cannot send an ARP message because client does not know its IP address
- Server can use ioctl(8) {or arp -s } to set the value of the cache based upon BOOTP request => can do this only if it has permission
BOOTP Features (Continued)

- Else send broadcast reply
- Reply: IP Address, Boot Server IP address, Default Router, Boot file name, subnet mask
- More information, but still only a single packet exchange
- Client gets boot image using TFTP => booting still a 2-step process
BOOTP features (Continued)

- Advantages of using UDP/IP:
  - Bootstrapping can occur across a router via a relaying mechanism
  - BOOTP uses checksum provided by UDP
- Multiple requests/replies
  - Process the first one
  - Client uses a transaction ID field to sort out replies
- Clients responsible for reliability
  - Uses timeout, retransmission & exponential backoff
  - Random initial timeout (betn 0 & 4s): simultaneous reboot after power restoration.
<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>H/W Type</td>
<td></td>
</tr>
<tr>
<td>H/W Length</td>
<td></td>
</tr>
<tr>
<td>Hops</td>
<td></td>
</tr>
<tr>
<td>Transaction Identifier</td>
<td></td>
</tr>
<tr>
<td>Seconds elapsed</td>
<td></td>
</tr>
<tr>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>Client IP Address</td>
<td>16 B</td>
</tr>
<tr>
<td>Your IP Address</td>
<td>64 B</td>
</tr>
<tr>
<td>Server IP Address</td>
<td>128 B</td>
</tr>
<tr>
<td>Router IP Address</td>
<td></td>
</tr>
<tr>
<td>Client H/W address</td>
<td></td>
</tr>
<tr>
<td>Server Host Name</td>
<td></td>
</tr>
<tr>
<td>Bootfile Name</td>
<td></td>
</tr>
<tr>
<td>Vendor Specific Area</td>
<td></td>
</tr>
</tbody>
</table>
BOOTP Message (Continued)

- Operation: 1 = Request, 2 = Reply
- H/w type: 1 = Ethernet
- H/w Address Length
- Hops: Initialized to zero. Incremented by BOOTP relays (routers)

**Diagram:**

- **BOOTP Client:** Please tell me my address
- **BOOTP Relay:** My client needs an address
- **BOOTP Server:** Your client’s address is...

Your address is ...
**BOOTP Message**

- **Boot File name:** Generic name like "unix" in the request. Full name in response.
- **Vendor specific area:** Misnomer. Also used for general purpose info.
- **Magic cookie:** First 4 octets = 99.130.83.99
- **Type-length-value:** describes the option

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Time of Day</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>End</td>
<td>255</td>
<td>-</td>
</tr>
</tbody>
</table>
Method 2a: DHCP

- **BOOTP limitation:** cannot dynamically assign IP address
- Dynamic Host Configuration Protocol (DHCP)
  - BOOTP + Dynamic allocation of IP addresses
    - => compatible with BOOTP.
  - No new fields in header.
  - Addresses can be **leased** for a period. Reallocated to the same or other nodes after lease expiry.
## DHCP Message Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>1 B</td>
</tr>
<tr>
<td>H/W Type</td>
<td>1 B</td>
</tr>
<tr>
<td>H/W Length</td>
<td>1 B</td>
</tr>
<tr>
<td>Transaction Identifier</td>
<td>6 B</td>
</tr>
<tr>
<td>Seconds elapsed</td>
<td>4 B</td>
</tr>
<tr>
<td>Flags</td>
<td>4 B</td>
</tr>
<tr>
<td>Client IP Address</td>
<td>4 B</td>
</tr>
<tr>
<td>Your IP Address</td>
<td>4 B</td>
</tr>
<tr>
<td>Server IP Address</td>
<td>4 B</td>
</tr>
<tr>
<td>Router IP Address</td>
<td>4 B</td>
</tr>
<tr>
<td>Client H/W address</td>
<td>6 B</td>
</tr>
<tr>
<td>Server Host Name</td>
<td>16 B</td>
</tr>
<tr>
<td>Bootfile Name</td>
<td>64 B</td>
</tr>
<tr>
<td>Options (Variable)</td>
<td>128 B</td>
</tr>
</tbody>
</table>
DHCP State Diagram

Host Boots → Initialize

Select → Offer → Request → Bound → Ack

Rebind

Nack or lease expires → Offer

Nack

Lease expires 87.5%. Request → Ack

Renew

Lease expires 50%. Request → Ack

Release

Request → Select offer/ Request → Select

Rensselaer Polytechnic Institute

Shivkumar Kalyanaraman
DHCP States

- Boots => INITIALIZE state
- DHCPDISCOVER: broadcast request to servers
  => SELECT state
- DHCPOFFER (from server) => remain in SELECT
- DHCPREQUEST => select one of the offers and notify server (goto REQUEST state) about the lease
DHCP States (Continued)

- **DHCPACK** => server Oks request to lease => go to the **BOUND** state
- Renewal: after 50% of lease go to **RENEW** state
- Rebind: after 87.5% of time, if server has not responded, try again and go to **REBIND**.
- If server NACKs or lease expires, or client sends **DCHPRELEASE**, go to **INITIALIZE**, else come back to **BOUND** state
Answer to 7 config problems…

1. End systems: Layer 3 address, names, masks: DHCP
2. Router finds Layer 3 addresses of end systems: Same network ID (i.e. IP prefix)
3. Router finds Layer 2 addresses of end systems: ARP
4. End systems find a default router, name server: DHCP
5. End nodes on the same LAN discover that they can send directly to each other: Same network ID + ARP
6. End systems find the best router for exit traffic: ICMP Router Redirect
7. End systems communicate on a router-less LAN: need a DHCP server at least. Same prefix => same LAN; ARP

Zeroconf IETF WG: networking without server-based configuration in certain scenarios…

http://www.ietf.org/html.charters/zeroconf-charter.html
NAT: translate addresses, without changing the application

10.0.1.2 192.1.2.3

Private addresses

Global Internet

128.96.41.1
Address Management: Private Addresses

- Since IPv4 addresses are scarce, enterprises may use private addresses within their “realms”

<table>
<thead>
<tr>
<th>Class</th>
<th>Private Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.0.0.0 … 10.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>172.16.0.0 … 172.16.255.255</td>
</tr>
<tr>
<td>C</td>
<td>192.168.0.0 … 192.168.255.255</td>
</tr>
</tbody>
</table>

- Need to get “globally unique” public addresses for external use.

- Mapping between public & private addresses done by NAT (Network Address Translator)
Simple NAT operation
Dynamic NAT = NAT + DHCP
Network Address Port Translation (NAPT)
NAPT (contd)

- Also known as IP masquerading. Allows many hosts to share a single IP address differentiated by port numbers.
- Eg: Suppose private hosts 192.168.0.2 and 192.168.0.3 send packets from source port 1108.
- NAPT translates these to a single public IP address 206.245.160.1 and two different source ports, say 61001 and 61002.
  - Response traffic received for port 61001 is routed back to 192.168.0.2:1108,
  - Traffic for port 61002 traffic is routed back to 192.168.0.3:1108.
Realm-Specific IP (RSIP)

- NAT (and NAPT) have to mess with several transport/application level fields.
- NAT breaks IPSec.... Solution: RSIP
- RSIP leases public IP addresses and ports to RSIP hosts => not transparent like NAT.
  - RSIP does not operate in stealth mode and does not translate addresses on the fly.
  - RSIP allows hosts to directly participate concurrently in several addressing realms.
- Avoids violating the end-to-end nature of the Internet => allows IPSec
RSIP “external address server”, eliminating side effects of NATs

But the applications must be upgraded...
network address translators and app-layer gateways

- inevitable loss of some semantics
- hard to diagnose and remedy end-to-end problems
- stateful gateways inhibit dynamic routing around failures
- no global addressability => brokered with NATs
- new Internet devices more numerous, and may not be adequately handled by NATs (e.g., mobile nodes)
Argument against NATs

- End-to-end vs Optimizations
- Short term problem
  - Connect many computers,
  - IP address are expensive
- Short term optimization
  - Use a NAT box,
  - Hide many computers behind one address
  - Works well for web clients…
- Addresses are the key…
  - Scarcity: the user is a “client”
  - Plethora: the user is a “peer”

Qn: Today’s optimizations, tomorrow’s roadblocks?
Can’t We Make NATs Better?

- we could keep adding more protocols and features to try to alleviate some of their shortcomings
  - might improve their functionality, but will increase their complexity, fragility, obscurity, unmanageability,...
  - new problems will arise when we start needing inter-ISP NAT

- **Anti-NAT suggestion**: moving to IPv6 will avoid the need to continue doing many other things to keep the Internet working and growing
  - IPv6 is not the only possible solution, but the most mature, feasible, and widely agreed-upon one
Network Management

- Management = Initialization, Monitoring, Control
  - Today: automated, reliable diagnosis, and automatic control are still in a primitive stage
- **Architecture**: Manager, Agents & Management Information Base (MIB)
- Observe that management-plane has a new interface to the network distinct from data-, and control-plane
SNMP History

- Early: based upon ICMP messages (eg: ping, source routing, record routing)
- A lot of *informal* network debugging is done using tcpdump, netstat, ifconfig etc
- When the internet grew, Simple Gateway Management Protocol (SGMP) was developed (1987)
- Build single protocol to manage OSI and IP
  - CMIP (an OSI protocol) over TCP/IP {called CMOT}
  - Goal: Keep object level same for both OSI and IP
  - CMOT progressed very sluggishly
- SNMP: parallel effort. Very simple => grabbed the market.
SNMP

- Based on SGMP
- Simple: only five commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-request</td>
<td>Fetch a value</td>
</tr>
<tr>
<td>get-next-request</td>
<td>Fetch the next value</td>
</tr>
<tr>
<td>get-response</td>
<td>Reply to a fetch operation</td>
</tr>
<tr>
<td>set-request</td>
<td>Set (store) a value</td>
</tr>
<tr>
<td>trap</td>
<td>Agent notifies manager</td>
</tr>
</tbody>
</table>

- Simple: handles only scalars. “get-next-request” used successively to get array values etc
- SNMP is bare-bones protocol to support monitoring & management

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
SNMP (Continued)

- Stateless => one management station can handle hundreds of agents
- Simple: Works as an application protocol running over UDP
- Agent and manager apps work on top of SNMP
- **Proxy-SNMP** can be used to manage a variety of devices (serial lines, bridges, modems etc).
  - Proxy (similar to bridge) is needed because these devices may not run UDP/IP
  - For *each new device define a new MIB*. 
Management Information Base (MIB)

- Specifies *what variables the agents maintain*
- Only a **limited number of data types** are used to define these variables
- MIBs follow a fixed naming and structuring convention called “Structure of Management Information” (SMI). See next slide.
Management Information Base (MIB) (Continued)

- Variables are identified by “object identifiers”
  - Hierarchical naming scheme (a long string of numbers like 1.3.6.1.2.1.4.3 which is assigned by a standards authority)
  - Eg:
    iso.org.dod.internet.mgmt.mib.ip.ipInReceives
    1.3.6.1.2.1.4.3
Global Naming Hierarchy

- ccitt(0)
  - iso (1)
    - joint-iso-ccitt (2)
      - standard (0)
      - dod (6)
        - internet (1)
          - directory (1)
          - mgmt (2)
            - experimental (3)
              - private (4)
                - mib (1)
                - fddi (8)
                  - fddimib (73)
                - fddi (15)
          - transmission (10)
        - interfaces (2)
          - system (1)
    - org (3)
      - iso9314 (9314)

Internet SMI is this subtree
<table>
<thead>
<tr>
<th>Variable</th>
<th>Category</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysUpTime</td>
<td>system</td>
<td>Time since last reboot</td>
</tr>
<tr>
<td>ifNumber</td>
<td>interfaces</td>
<td># of Interfaces</td>
</tr>
<tr>
<td>ifMTU</td>
<td>interfaces</td>
<td>MTU</td>
</tr>
<tr>
<td>ipDefaultTTL</td>
<td>ip</td>
<td>Default TTL</td>
</tr>
<tr>
<td>ipInReceives</td>
<td>ip</td>
<td># of datagrams received</td>
</tr>
<tr>
<td>ipForwDatagrams</td>
<td>ip</td>
<td># of datagrams forwarded</td>
</tr>
<tr>
<td>icmpInEchos</td>
<td>icmp</td>
<td># of Echo requests received</td>
</tr>
<tr>
<td>tcpRtoMin</td>
<td>tcp</td>
<td>Min retrans time</td>
</tr>
<tr>
<td>tcpMaxConn</td>
<td>tcp</td>
<td>Max connections allowed</td>
</tr>
</tbody>
</table>
Instance Identification

- How does the manager refer to a variable?
  - **Simple variables**: append “.0” to variable’s object identifier
    - Eg: udpInDatagrams.0 = 1.3.6.1.2.1.7.1.0
    - Only leaf nodes can be referred (since SNMP can only transfer scalars)
MIB (Continued)

- All names are specified using a subset of Abstract Syntax Notation (ASN.1)
- **Types**: INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL
- **Constructors**: SEQUENCE (like struct in C), SEQUENCE OF (table i.e. vector of structs), CHOICE (one of many choices)
- ASN.1 provides more types and constructors, but they are not used to define MIBs.
RMON

- Remote Network Monitoring
- Defines remote monitoring MIB that supplements MIB-II and is a step towards internetwork management
- It extends SNMP functionality though it is simply a specification of a MIB
- Problem w/ MIB-II
  - Can obtain info that is purely local to individual devices
  - Cannot easily learn about LAN traffic as a whole (eg like LANanalyzers or “remote monitors”)
RMON (Continued)

- Functionality added: Promiscuously count, filter and store packets
- System that implements RMON MIB is called an RMON probe (or less frequently, an RMON agent).
  - No changes to SNMP protocol.
  - Enhance the manager and agents only.
- RMON MIB organization:
  - Control table: read-write. Configures what parameters should be logged and how often.
  - Data table: read-only (statistics etc logged)
- Other issues: shared probes, ownership of tables, concurrent table access ...
Summary

- **Error reporting** is a separate protocol in IP: ICMP
  - Features help build neat tools: ping, traceroute etc
- **Configuration:**
  - 7 basic configuration problems
  - Internet solution: ARP, DHCP (server-based)
  - Earlier attempts: RARP, BOOTP
- **Address Mgmt:** Private addresses, NAT, NAPT, RSIP
- **Management** = Initialization, Monitoring, and Control
  - SNMP = Only 5 commands (simple polled transfer of management information)
  - MIB: labeling of mgmt info using ASN.1 encoding
  - Standard MIBs defined for each object
  - RMON extends management functionality through definition of a new MIB (no protocol changes)