Auto-Configuration

TCP/IP class

Jim Binkley

overview

- problem statement
- in the beginning (rarp/icmp)
- bootp
- dhcp (bootp++)
- tftp
- what else?

definition - **auto-configuration**

- limited sense: - at boot, end system can determine its own network address dynamically, not manually via net. admin.
  - plus for Novell ipx, minus for tcp/ip
- general: can we obtain all magic numbers and names (even news server or info about local bathrooms) dynamically ?

basic problem

- tcp/ip client must have at least the following bits of information:
  - **ip address**, **subnet mask**, **broadcast address**
  - former two must be set, latter can be determined in sw(but may not be)
  - **router address** (RIP or IGP might be used to dynamically discover , but many PCs can’t do that)
  - **local DNS servers** (1, better 2)

but it’s not really that simple

- we may need a DNS name (servers)
- we may want a proxy HTTP server address
- and NTP time, or news, or printers?
- and a mail server or pop/imap, etc.
- the cheapest latte in Portland (general info), the nearest bathroom (right now!)?
- RSN - key servers and electronic cash machines?

ISO/Novell ipx situation

- ISO address is variable length <= 20 bytes,
  - address is roughly (network, host)
- router advertises with “router hello” the router address, host can learn that
- host uses MAC address as host portion
- Novell/ipx client can broadcast to learn net
- uses MAC address for host portion
- thus we have (net, host), no manual admin

rarp - reverse arp (1st attempt)

- instead of knowing ip, and broadcast for destination mac, we know mac and broadcast for our ip address at boot
- uses arp header. above link layer, two commands
  - broadcast: rarp request, here's my mac, give me my ip
  - unicast: rarp reply from rarp server
- rarp server on link, contains tables of (mac, ip)

rarp continued

- rarp servers should be redundant in case one crashes
- used by simple pcs, X-terminals for boot and download
  - use rarp to get ip address
  - use tftp to download operating system
- **our motto: “have mac, need ip & brains”**

rarp continued

- pros: simple (too simple...)
- cons:
  - only 1 piece of info. not enough
  - servers must interface directly to link layer
  - rarp broadcasts at “link layer”, not ip, therefore routers may not easily forward them
  - therefore pain with multiple IP networks
  - arp is not routable (duh ...)

icmp (odds and ends)

- icmp has some boot-time info
  - router advertisement/solicitation - learn router
  - timestamp query - get the time
  - subnet mask query - get the subnet mask
- router advert/solicitation appeared in 4.4 BSD, some routers/hosts may know how to do it, others may not

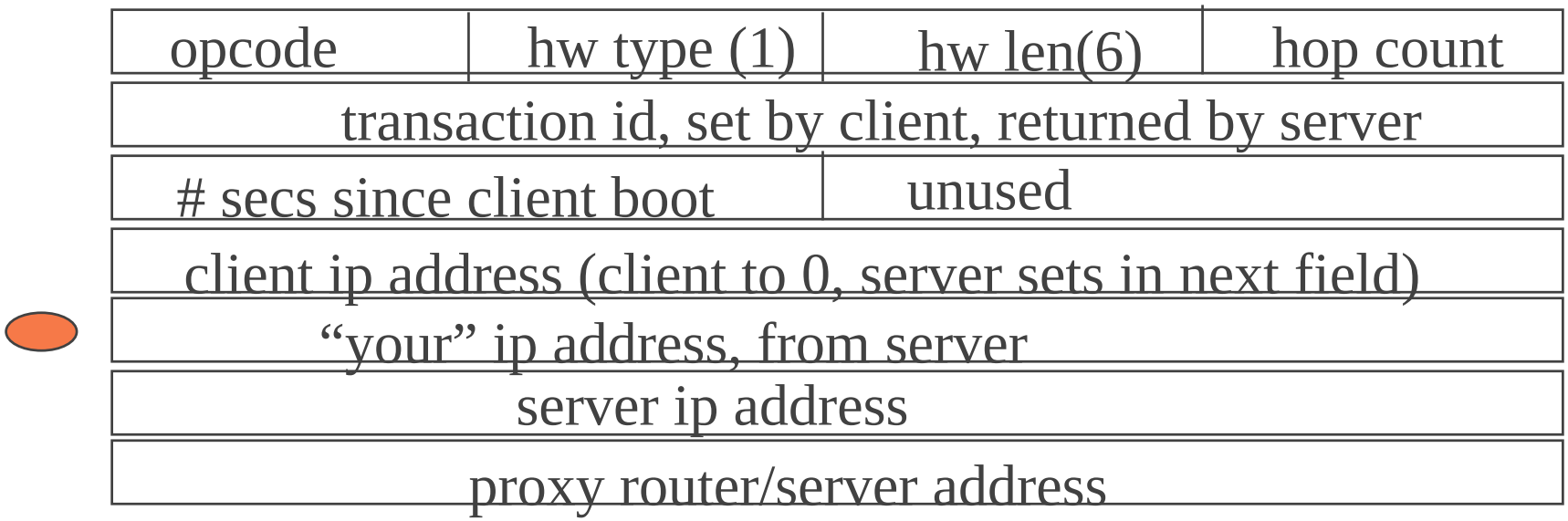
bootp - bootstrap protocol

- rfcs include: 951(1985)/1533/1542
- built on top of UDP (routable)
  - broadcast request
  - unicast reply from bootp server
- uses ports 67 for server and 68 for client
- since broadcast is IP address, router can support “bootp relay agent” to bridge nets
- bootp often paired with tftp for download

bootp

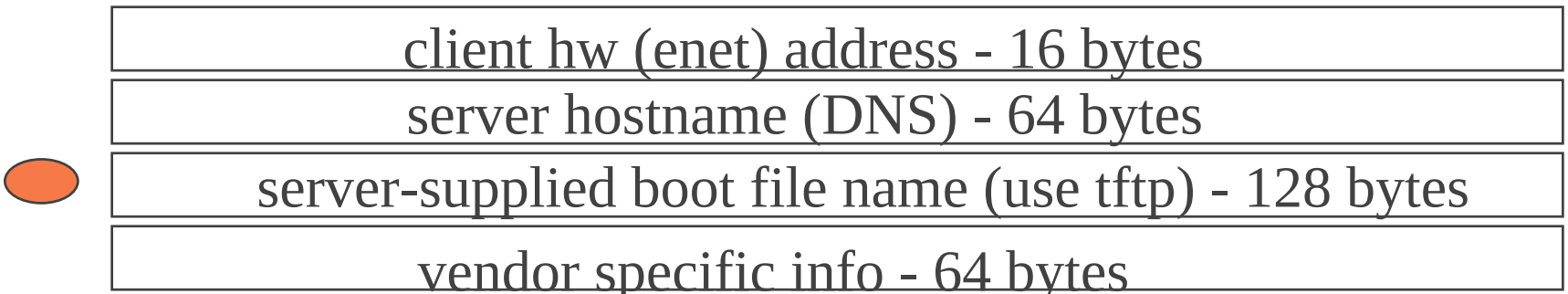
- dest ip: 255.255.255.255
- src ip for client: 0.0.0.0
- problem with reply is that server doesn’t have IP, mac address in arp table and can’t arp for client
- if server can insert entry into own arp cache, can unicast reply
- else must broadcast

bootp header - 1st part



300 bytes maximum length
hw type is enet,
hop length for client = 0, used by proxy router

bootp header - 2nd part



vendor specific info - RFC 1533 defines format

magic cookie: 1st 4 bytes has ip address 99.130.83.99

means info exists

rest of area is list of items in Tag, Length, Value format

Tag - 1 byte, denotes type, 255 means end tag

length - 1 byte, length of value field

bootp/vendor-specific TLVs

- **subnet mask**
  - host requirements RFC “deprecates” use of ICMP subnet mask request in favor of bootp
- time offset since UTC, Jan 1 1900, UTC
- **router**
- **DNS server** (!)
- print server, and more but field size is limited

bootp pros/cons

- pros
  - uses ip broadcast, can be forwarded across router
  - server is udp server therefore simpler
  - more information, can get most basic info needed
- cons
  - packet size is fixed, thus limits on info passed
  - you can assign ip address but can’t get it back
  - you need to collect mac addresses of course to do mapping, mac address A gets IP address B

dhcp - bootp++

- dhcp - dynamic host configuration protocol
- 1. a better “bootp”, more options plus
- 2. ip addresses can be leased for a certain time, thus they can be reclaimed
- can use bootp relay agent
- useful for mobility?
  - can move laptop to new subnet, get new ip

current rfcs

- 2131, Dynamic Host Configuration Protocol, R. Droms, March 1997
- 2132, DHCP Options and BOOTP Vendor Extensions. S. Alexander, R. Droms, March 1997
- 2489, Procedure for Defining New DHCP Options, R. Droms, January 1999

more options

- ip layer might want
  - ip forward enable/disable
  - max datagram reassembly size
  - default ip ttl
  - path mtu aging timeout
  - static routes
- arp
  - cache timeout value

etc - more options on heaven/earth...

- tcp
  - default ttl
  - tcp windowsize and keepalive interval
- nis
  - domain
  - servers
- X font server, etc., etc.
- printers/time servers, blah, blah

ip address allocation

- can be static (like bootp)
  - dhcp should be able to do bootp like stuff
  - and more ... e.g.,
- dynamic including automatic allocation from IP address pool
  - pro there can be no mac addresses stored
  - or map them to IP addresses for yet another variation

protocol:

- client can send DHCPDISCOVER (broadcast) to server port 67, UDP
  - 4 retries, 2 second interval, 5 minute try again
- server/s sends DHCPOFFER,
  - default 1 hour lease
- client sends DHCPREQUEST to particular server - may have > 1 server, choose one
- DHCPACK from server, or DHCPNACK

protocol:

- client can use IP address
- time down to 50% left, renegotiate
- server can try (should ...) and return same IP address, so client connections not adversely affected
  - if new IP address, TCP connections are toast
- client may use ARP to check for IP address owned by another client
- server may use ping to check for IP address usage as well

dhcp protocol: cont,

- other messages exist:
- client may send DHCPRELEASE to give up IP address (i'm done ...)
  - of course, if you turned it off it may not have a chance
- DHCPINFORM used by client to get other info from server

pros/cons

- more info is a pro
- time leases MIGHT be a pro
  - reuse of scarce IP addresses, few IP addresses mapped to many part-time hosts
- cons: what about DNS name for box?
  - (dns name dhcp25.foo.com, IP address) (both change)
  - can't fix fact that DNS entries are static, therefore if we move and change ip addresses,
  - **we might not want to change our DNS name!**

other possibilities

- **Mobile-IP**: assumption that you keep the same IP address when you move
  - routing protocol
- not reasonable to assume that mobile hosts will never have servers ... (why not a web cam/http server on your laptop?)
- dynamic DNS mapping possible, but by definition world-wide server caching is a problem **- security here very important**

DHCP security

- 2131 says there is none. DHCP packets have no authentication
- some possible security threats
  - fake dhcp server (maybe just DOS, or maybe works, but gives you itself as router in order
  - to inspect all your packets (man in the middle attack)
  - arp spoofing of course possible
  - MAC addresses are harder to spoof, but it is often possible for an attacker to spoof your MAC address

tftp - trivial file xfer protocol

- used for bootstrapping - download “os brains” to X terminal
- tftp uses UDP, makes it smaller for boot rom storage (traditional POV ...)
- rfc 1350 (1992)
- see Steven’s Network Programming book for source

tftp protocol

- client requests file by name
- server send data with block # (<=512 bytes)
- block # is sequence number
- client ACKS block #
- stop and wait protocol or “ping-pong”
- tftp assumes UDP checksum (oops)
- server assigns one UDP dest per client

tftp security

- ha ha ha ha! (oh...)
- no password or username
- on UNIX server typically have
  - /tftpboot directory only place tftpd can access
  - tftp server is user nobody or some other user so files must be world-readable and server can't get at root files
  - try not to leave "important" files there
  - **block tftp access across external routers**

what else? - past and future attempts

- **anycasting** - you have a magic ip address that the network (routers) will use to help you find the nearest XYZZY server
- Novell SAPS - broadcast service info around net (not very scalable)
- Service Location Protocol, RFC 2608, June 1999
  - basic idea: i want TIME, here's IP, use NTP ...
  - dhcp ++ ?! :-)
- directory services might provide such info

research areas

- mobile systems - need to (re) discover available resources when they move from site to site (bathroom OR dns server)
- general problem of information retrieval on Internet (aka **devil IS the details**, thanks)
  - agents that periodically search for X and send it to you
    - » not loved due to “too much email now, thanks” problem

research, cont.

- scalability: how to make these services deal with ever more hosts
  - ip addresses, MAC addresses, whatever
- desire to deal with smaller HOST, with less pieces of puzzle, make infinitely dynamic
  - HOST says: I am incomplete, may I have more soup please ? (DLLs, IP addresses, server addresses, dynamic code, you name it)
  - network deity: here you go ...