Post 1

Thursday, January 24, 2019 11:43 PM

WCNA CH1 NOTES

Protocols	Set of communications standards dealing with machines/networks
TCP/IP	Protocol suite that provides services in use on Internet • **TCP: Transmission Control Protocol** 1. Handles reliable message delivery • **IP: Internet Protocol** 1. Manages routing of network transmissions between senders/receivers
Protocol suite	Family of networking protocols
Packets	Generic PDU term at any layer in a networking model: Applied PDU's at L3

History of TCP/IP: 1969: **DoD: ARPA: Advanced Research Projects Agency**

- Funded academic research project involving long-distance network [packet-switched]

Packet-switched network: Packets take any usable path between sender/receiver

- I**D by unique network address:** Packets not required to follow same path (even though they usually do)
- **ARPANET**: Networks built as a result of this project

Design: Based on gov't needs:

- Ability to withstand potential nuclear strike
- Communication between different kinds of systems
- Interconnection across long distances: 1960's "Big iron" era: Systems used terminals/were expensive

Original ARPANET linked:

- Stanford Research Institute | University of Utah | campuses in University of California system: Los Angeles/Santa Barbara

TCP/IP: 1970's: Early researchers: Data had to move across different network types/locations

- Necessary to permit LANs w/Ethernet to use long-haul networks
- TCP/IP: Began 1973 || IPv4: 1978: Developed
- **Original Internet:** Established a model for a network made of other networks

Internetwork: A single logical network made of multiple physical networks

1983: DCA: Defense Communications Agency (**DISA: Defense Info Sys Agency**) took over**ARPANET** from **DARPA**

- **DARPA: Defense Advanced Research Projects Agency**
- More widespread Internet use: More schools/agencies/facilities/contractors began to rely on it for data/email exchange

Same year: DoD instituted requirement that all computers on Internet switch to TCP/IP

Berkeley Distribution UNIX: 4.2 BSD: Included support for TCP/IP: Helps explain proliferation of Internet protocols

MILNET: All-military net: Split off from ARPANET: Divided infrastructure into military-only/public side of Internet

University of Wisconsin: Dev name server technology: Allowed users to locate/ID network addresses anywhere on the Internet

Groups that Oversee TCP/IP: Most important IETF: Responsible for RFC's

ISOC	**Internet Society** • Parent org for all Internet boards/task forces • Non-profit/non-gov't/int'l/membership org funded by dues • Corporate contributions/occasional gov support
IAB	**Internet Architecture Board: AKA: Internet Activities Board** • Arm of ISOC: Standards-making/research groups that handle current/future tech/protocols/research • Oversight for architecture of protocols/procedures • Editorial oversight for **RFC's: Request for Comments** (where Internet Standards are stated)
IETF	**Internet Engineering Task Force:** Drafting/testing/proposing/maintaining Internet Standards (RFC)

	• Participation of multiple working groups **IETF/IAB**: Process of **rough consensus** to create standards • Participants in standards-making process must agree before proposed/drafted/approved
IRTF	**Internet Research Task Force:** Fwd-looking activities of ISOC • Research/dev work for topics too far out for implementation but may play role someday
ICANN	**Internet Corporation for Assigned Names/Numbers** • Ultimate responsibility for managing domains/network addresses/protocol parameters/behavior • Delegates mgmt of customer interaction/money/DB maintenance to other authorities

IPv4/IPv6

IPv4	**32-bit addresses** || Roughly 4.3*10^9 || 4 billion addresses: 3 billion usable **1990s:** Obvious address space limited • **9/2015:** America ran out of IPv4 addresses: IPv6 adoption: 21.31% • **02/2011:** ICANN dispensed last few unallocated Class C address blocks
IPv6	**128-bit addresses** || Roughly 3.4*10^38 || 8*10^28 larger than IPv4 • **2 to power of 128:** 340,282,366,920,938,000,000, 000,000,000,000,000,000 unique addresses

Standards/RFC: Old vers of RFC's replaced by newer/more up-to-date ones: ID by number
2/more RFC's cover same topic? Share same title: Highest number: Current: Older obsolete
RFC: Internet Official Protocol Standards: Snapshot of current standards/best practices
RFC 2026: How an RFC created/processes it must go through to be adopted by IETF **||** 7 phases while becoming standard
Potential Standard RFC 3 phases:

1. **Proposed Standard:** Initial review: 2/more reference implementations to demonstrate interoperability
2. **Draft Standard:** Approved from proposed
3. **Internet Standard:** Specifies rules/structure/behavior
4. **Historic Standard:** RFC overridden by newer vers

BCP	**Best Current Practice**: Another category of RFC • Philosophy/approach design/implementation recommended • Not standards: Worth reading/applying

OSI Model Overview: International Org for Standardization Open Systems Interconnection:
- Standards for info tech/networking equip/protocols/comm tech
- Describe how networks operate from HW to SW lvl

ISO Standard 7498: OSI model: 7 layers: AKA reference model
- Dev as part of international standards initiative: 80s
- Intended to usher new/improved suites of protocols to replace TCP/IP

Networking in Layers: Divide and conquer
- **Divide/Conquer:** Deconstructing complex problems into smaller/less complex/interrelated problems
- Can be solved more/less independently of others

Key points about networking: Challenges easier to overcome when big tasks broken into smaller ones
- Layers operate independently of 1/another: Modular design of specific HW/SW components: Perform individual functions
- Individual layers encapsulate traffic: Independent functions changing layers need not affect others
- Layers work together: Sending machine performs ops on 1 layer: Reversed by ops at same layer of receiver
    - **Peer layers:** Analogous: Receiving layer reverses operations sending layer performs
- Different expertise needed: Implement solutions for networking functions at each layer
- Layers in network work together to create general solution
- Network protocols usually map to 1/more layers of model
- TCP/IP is designed around layered model

OSI Model Layers:
- Application
- Presentation
- Session
- Transport
- Network

- Data Link
- Physical

How Protocol Layers Behave: Layers encapsulate/isolate specific types of functionally

- Layers exist to provide services to layer above: Deliver outbound traffic
- Layers exist to provide services to accept data for inbound traffic: From layer below

PDU	**Protocol Data Units:** SW handles packages of data: Packets irrespective of layer addressed **Header:** Layer-specific label for whatever PDU precedes **Trailer:** May include error-detection/correction info/end of data flag/other data

Layers: More detail:

Physical	**Physical transmission medium** (cables) network must use to send/receive signals of comm • Activate/maintain/deactivate connections ○ **Senders:** Attempt to establish connection to transmit data across medium ○ **Receivers**: Respond to attempts by accepting/rejecting them • Conversion of outgoing data from bits machines use to signals networks use • Physical layer reverses/converts incoming signals from networking medium into bits to be sent through NIC **PDU's at layer:** Signals correspond to bit patterns for frames at Data Link
Data Link	Reliable transmission of data through physical layer at sending end • Checks reliability: Receiving end • Manages point-to-point transmission across medium ○ From 1 machine to other on single logical/physical cable segment **Point-to-point transmission:** Pairs of devices establish comm link to exchange data: Machine/ISP **Cable segment:** Media/devices that fit on single cable/hub/virtual equip/LAN emulation env on switch • Handles sequencing of data from sender/receiver • Patterns of bits must be mapped to corresponding signals for transmission: Process reversed when receiving • **Data Link**: Can control pace at which data transmitted **Media flow control:** Responds to congestion: Keeps network from being swamped by traffic • Mgmt of data transmission rates bet devices across medium • Guarantees receiver can accept/process input before it arrives from sender • Requests data transfers to occur when outgoing PDU's reader to be transmitted • Handles accepting/constructing incoming PDU's for incoming data • PDU's at L2 must fit into bit patterns that map carrying capacity of medium: Fmt/structure/max data size **Responsible for brokering certain connection types** Example: When a connection uses circuit switching technology (telephone system) **Circuit switching:** Establishes dedicated channel for duration of transmission between 2 end points • TCP/IP can use such comm links: Treats them no differently than other point-to-point links **PDU's at layer:** Called frames/data frames
Network	Where network locations addressed: Intricacies involved in directing PDU from sender/receiver handled • Handles logical addresses w/machines: Permits human-readable names w/numeric addresses • Uses addressing info to determine how to send PDU when source/des doesn't reside on same segment **Primary function:** Provides globally unique address to every host on Internet/paths to/from hosts • Multiple simultaneous connections bet diff IP's so numerous apps can maintain connections at same time **Protocol level:** ID which connection belongs to process/app • Not only directs traffic to proper receiver from sender • Explains how you can have a browser open at same time you're reading email • Also carried out by port number/assignments in Transport **TCP/IP and UDP networks:** Logical connections called ports: Specifies processes on computer • Many ports have preassigned functions: FTP/HTTP/POP3 • Flexible: Recognize/use multiple routes bet sender/receiver while ongoing comm underway **Packet switching:** Used to fwd/relay individual PDU's from sender/receiver

	• Sensitive to delays: Can manage traffic sent across while forwarding data from sender/receiver **Congestion control:** TCP mechanism: Available for other protocols: • Permits hosts to exchange info about ability to handle traffic • Senders decrease/increase frequency/size of upcoming comms
Transport	Ensure reliable end-to-end transmission of PDU's from sender/receiver **To enable this function to occur:** • Includes end-to-end error detection/recovery data • Data packaged as part of trailer for Transport-layer PDU: Checksums calculated before/after delivery **MTU: Max Transmission Unit:** Containers that transport data from end to end that have a fixed max size • Handles segmentation/reassembly **Segmentation:** Cutting up a big msg into a numbered sequence of chunks called segments • Each chunk represents the max data payload network media can carry bet sender/receiver **Reassembly:** Process of chunks put into original order/used to re-create data as it was before segmentation **Fragmentation:** Sender end: Data payload for TCP broken into sequence of fixed-size payloads called segments • Reassembled by TCP at receiving end **ONLY exception: Fragmentation:** • When TCP segment must travel across link where MTU is smaller than packet size for TCP segment • Won't be reassembled until arrived at receiving host • Host handles reassembly of fragments & original segments as incoming packet structures need Equipped to request retransmission of erroneous/missing PDU's when reassembly underway: Guarantees delivery **PDU's at layer**: Called segments/data segments
Session	Ongoing comm bet sender/receiver set up/maintained/terminated/torn down as needed • Mechanisms to permit senders/receivers to request conversation start/stop • Mechanisms to keep conversation going even when traffic may not be flowing bet 2 parties • Provides mechanisms called **checkpoints** **Checkpoints:** • Point where system state/info captured/saved: After failure ops can resume at point in time w/no further loss • **Primary job**: Handle comm bet 2 parties where sequence of msgs/PDU's exchanged Example: User logs into DB (setup)/Enters queries (exchange)/logs off (teardown) **PDU's at layer:** Variety of types [30+]: Session PDU's/SPDU's
Presentation	Manages data presented to network on its way down stack: Way presented to machine on its way up stack • Handles transforming data from network-oriented to platform-oriented forms • **OS driver:** Said to reside at layer: Called redirector/network shell • Driver's job? Distinguish resource requests/redirect them to appropriate remote/local subsystem • Easier for devs to build apps that can access local/remote resources • Can supply special data-handling functions for apps ○ Protocol conversions, encryption (outgoing), decryption (incoming) **PDU's at layer:** Variety of types: Presentation PDU's
Application	Defines interface apps can use to request network services • Kinds of services apps can request from network • Stipulates forms data must take when accepting/delivering msgs to such apps • Defines set of access controls over network **PDU's at layer:** Application PDU's

TCP/IP Layers

Network	**TCP/IP Network Access Layer:** AKA: Network Interface layer

Access	• Where LAN media (Ethernet/wireless) and devices come in • Layer where WAN/connection-management protocols (PPP): Point-to-Point Protocol come in • X.25 packet-switched WAN protocol: Replaced by IP: Still used in some legacy apps **IEEE: Institute of Electrical/Electronic Engineers standards apply** **Includes 802 family of standards:** • **802.1 Internetworking:** How data works for 802 family • **802.2 Logical Link Control:** Links bet 2 devices: Physical: Established/managed • **802.2 MAC**: How media ints ID/accessed on network: MAC addresses ints • **802.3 CSMA/CD:** How Ethernet ops/behaves **Ethernet:** Shared medium that supports multiple access • Signal called carrier sense to detect when medium in use • Circuitry detects when 2 transmissions run into each other (collision) **CSMA/CD: Carrier Sense Multiple Access/Collision Detection** • Gigabit Ethernet (802.3z)/10/100/1000Mbps/10-Gbps varieties • Latest 802.3 standards cover 100G/40G Ethernet • **802.5 Token Ring:** ○ How IBM dev token ring ops/behaves ○ Fast Ethernet/Gigabit Ethernet largely replaced token ring ○ 802.5 standards dev at standstill • **802.11 Wi-Fi:** ○ Speeds from 1-1300Mbps (theoretical max) ○ Most common members: 11Mbps 802.11a/802.11b standards ○ 54Mbps 802.11g/72-150Mbps 802.11n multichannel ○ 802.11ac w/BW rated up to 450Mbps on 2.4 GHz band: 1,300Mbps on 5GHz band **PPP: Point-to-Point Protocol:** • L2: Permits client/server established comm links that accommodate higher-layer protocols (IP) **Datagrams:** Used by connectionless protocols at Transport • Adds header to PDU: Supplied from L7 protocol/service: UDP: Connectionless **PPP: Most impt Network Access layer protocol** • Used to establish direct connection bet pair of networked devices • Can provide connection auth to establish ID's/encrypt transmissions/compression • Variety of PPP: **PPPoE: Point-to-Point over Ethernet** **HDLC: High-Level Data Link Control protocol:** Based on IBM's SDLC: Syncs Data Link Control protocol: • Uses data frames to manage network links/transmission **Frame relay:** • Telecom service designed to support intermittent data transmission bet LANs/WAN end points • Uses data frames to manage network links/data transmission • Slowly being phased out by most ISPs: Still used in rural areas **ATM: Asynchronous Transfer Mode** • High-speed, cell-oriented connection-switching tech • Used for digital voice/data comm: Backbone/infrastructure **PPTP: Point-to-Point Tunneling Protocol:** L2 Network Interface layer protocol • Allows client/server to establish secure/encrypted comms link for PPP traffic **VPN: Virtual Private Network**: Connection bet specific sender/receiver: Info sent often encrypted • Uses public networks: Internet: To deliver secure, private info from sender/receiver **PDUs at layer: Called datagrams**
Internet	Handles routing bet machines across multiple networks: Manages names/addresses • Handles moving data from sender/receiver: Repackages data: Handles issues • Defines how to get here to there **3 Primary tasks:** 1. MTU Fragmentation 2. Addressing 3. Routing

	MTU fragmentation: • Data from 1 network to another via routers carry diff MTU varies • When 1 medium that supports larger MTU's and another doesn't: Size must be reduced • Only needs to be 1 way: Must be performed while data in transit **Addressing:** All network ints associated w/unique patterns to ID each int/network **Routing:** Forwards packets from sender/receiver • Numerous relays may be involved in achieving delivery • Not only the processes involved in delivery but methods to track performance **Protocols:** **IP: Internet Protocol:** Routes packets from sender/receiver **ICMP: Internet Control Message Protocol:** Handles info about IP routing/behavior: Traffic conditions/errors **PING: Packet Internetwork Groper:** Checks accessibility/round-trip time bet sender/receiver **ARP: Address Resolution Protocol:** Converts bet numeric IP/MAC's on specific cable segment: Eliminated under IPv6 **RARP: Reverse Address Resolution Protocol:** Converts MAC address into numeric IP • ARP/RARP have to bridge L2/3 b/c they work w/MAC/IP • They are considered L2 protocols: Functions w/in DLL code modules: RARP eliminated in IPv6 **BOOTP: Bootstrap Protocol:** Precursor to DHCP • Manages network allocation of IP's/other config data • Some portions of BOOTP replaced by DHCP: Other elements used to provide service to it **RIP: Routing Info Protocol:** Original distance-vector/basic routing for local regions w/local internetworks **OSPF: Open Shortest Path First:** Defines widely used link-state: Local/interior routing w/in internetworks **BGP: Border Gateway Protocol (BGP):** Connects common backbones w/in Internet: • Multiple parties share responsibility for managing traffic.
Transport	**Sometimes called host-to-host layer:** Helps move data from 1 host/other: Reliable delivery • Necessary segmentation of outgoing msgs prior to transmission • Reassembling msgs prior to delivery to App layer for further processing **2 protocols:** 1. **TCP:** Connection-oriented: Negotiates/maintains connection prior to sending data 2. **UDP:** Connectionless: Transmits data in best-effort delivery: No checking/follow-up
Application	**AKA: Process layer:** Where stack ints w/processes on host machine • Some services like NFS: Network FS/VoIP/Various streaming media (H.323) often use UDP • Most use TCP **Depend on 2 elements to operate:** 1. Daemons 2. Port addresses **Daemons:** Special listener process: Ops on server to handle incoming requests for specific services • **WinServer 12/16:** INETINFO.EXE in Task Manager whenever IIS/FTP server running • **UNIX:** FTP associated w/ftpd: Internet services run under inetd **Port addresses:** Services have associated port: Uses 16-bit number to ID: Addresses in 0-1024 range: Well-known

Protocols/Services/Sockets/Ports

- **TCP/IP inclusion in 4.2 BSD:** Milestone in history: Point where research/academics began working w/TCP/IP

Multiplexing: Combining various sources of outgoing data into single output data stream
Demultiplexing: Breaking up incoming data stream so separate portions may be delivered to correct apps

- Typically handled at Transport: Outgoing msgs broken into chunks sized for networks
- Incoming msgs reassembled in correct order from a stream of incoming chunks

To make this easier: TCP/IP uses protocol #'s to ID App layer protocols/services
Numerous port numbers reserved to ID well-known protocols

- An assigned series of numbers represent a sizable collection of TCP/IP-based services
- IP datagram header: Protocol number appears in 10th byte

- This 8-bit value indicates which Transport layer protocol should accept delivery of incoming data

TCP/IP Numbers

Number	Acronym	Protocol Name
0	**HOPOPT**	IPv6 Hop-by-Hop Option
1	**ICMP**	Internet Control Message Protocol
2	**IGMP**	Internet Group Management Protocol
3	**GGP**	Gateway-to-Gateway Protocol
4	**IPv4**	Internet Protocol v4 (encapsulation)
5	**ST**	Internet Stream Protocol
6	**TCP**	Transmission Control Protocol
7	**CBT**	Core Based Trees
8	**EGP**	Exterior Gateway Protocol
9	**IGP**	Interior Gateway Protocol
10	**BBN-RCC-MON**	BBN RCC Monitoring
11	**NVP-II**	Network Voice Protocol
12	**PUP**	PARC Universal Packet
13	**ARGUS**	ARGUS Protocol
14	**EMCON**	Emission Control Protocol
15	**XNET**	Cross Net Debugger Protocol
16	**CHAOS**	CHAOS Protocol
17	**UDP**	User Datagram Protocol
18	**MUX**	Multiplexing
19	**DCN-MEAS**	DCN Measurement Subsystems
20	**HMP**	Host Monitoring Protocol

UNIX: Contents of /etc/protocols don't need to contain every entry in Assigned Numbers RFC
TCP/IP Ports: IP passes incoming data to TCP/UDP at Transport: Protocol must do its work

- Passes data to intended app process: AKA **Network services**: ID'd by ports
- Source/Destination ports: ID process that send/receive data
- **Values:** 2-byte (16-bit): 1st header word of TCP segment/UDP packet
- **Range:** 0-65535

Traditionally: Below 256: Well-known services [Telnet/FTP] || **256-1024:** UNIX-specific services
Today: Below 1024: Well-known services: Many registered ports w/specific app services from 1024-65535

Sockets	**Well-known/registered ports:** Preassigned w/particular network services: Simplifies client/server connection **Dynamically allocated port number:** Not preassigned: Used as needed for temp connections: Limited data exchange • After client/server use well-known port: Session created: Temp pair of socket addresses: Send/receive ports for more comm **Socket address:** Combo of IP/dynamically assigned port

Data Encapsulation: At each layer: Data packaged/ID'd for delivery to layer underneath

- Incoming data: Has encapsulating info from underlying layer stripped before delivered to upper-layer

Header: Opening component: ID's protocol/sender/receiver/other info
Trailer: Closing component: Provides data integrity checks: AKA: **payload**
Encapsulation: Enclosure of payload bet header/trailer

- Data from upper layer gets manipulated/enclosed w/header/trailer before passing to layer below

Protocol Analysis: AKA Network analysis:

- Process of tapping into network comms: Capturing packets: Gathering statistics: Decoding packets

Protocol analyzer: Eavesdrops on network

Useful Roles for Protocol Analysis: Troubleshooting network comms: Placed/config to capture problems

- Reading packets can ID faults/errors
- Also used to test networks
- Passive: Listening to unusual comm
- Active: Transmitting packets onto network
- Monitoring traffic: Network utilization/packets-per-second rate/packet size distribution/protocols in use

Analyzer Elements:

Promiscuous mode card/driver:

- Packet filters
- Trace buffer
- Decodes
- Alarms
- Statistics

Promiscuous Mode Card/Driver	Packets enter analyzer from network using NIC **NIC/driver used must support promiscuous mode op** • Broadcast/anycast/multicast/unicast/error packets sent to other devices • Example: Collision fragments, oversized/undersized packets (runts) • Or packets on an illegal boundary • Collision/over-undersized packets reflect errors normally ignored by NICs when not in promiscuous **Ethernet collision fragment:** • Appears when 2 packets transmitted at same time run into 1/another • Produces random hash of signals • Increase in frequency as traffic volumes go up **Oversized packets:** Exceed MTU for type of network in use: NIC/driver problem **Undersized packets:** Don't meet reqs for min sizes: Indicate HW/driver problems **End on an illegal boundary**: Don't close properly: May have been truncated by HW/driver issues **Wireshark:** Shows errors only used w/WinPcap packet capture driver: AKA: **pcap**/compatible NIC **Pcap: Packet capture:** Based on well-known UNIX app programming int (API): AKA: **libpcap** **WinPcap:** Windows-compatible ver same code: Built into installation process **Pcapng: Packet Capture Next Generation:** Dev to overcome limitations in **libpcap** • Began using pcapng in version 1.8.
Packet Filters	**Defines type of packets analyzer wants to capture** • When filters applied to incoming packets: Referred to as capture filters/pre-filters • Can apply to set of packets after capture • Enables creation of subsets of interesting packets: Easier to view **Can be based on a variety of packet characteristics:** • Source/destination data link address • Source/destination IP address • Application/process
Trace Buffer	**Packets flow into analyzer's trace buffer: Holding area for packets copied off network** • Typically: An area of mem set aside on analyzer • Some analyzers allow you to config a "direct to disk" save option • Can be viewed immediately after capture • Buffer robust: Can fill quickly on high-speed connections unless filter applied • Usually a default trace buffer of 4 MB

Decodes	**Packet-translation tools:** Applied to packets that captured in trace buffer: • Enables you to see packets in readable fmt w/packet fields/values interpreted • Can separate all fields of header w/in a packet, defining source/dest. IP/purpose of packet.

Wireshark interface panes
Packet list pane: Top: Decoded view
Packet details pane: Middle
Packet bytes pane: Bottom: Encoded view/byte-level data

Alarms	Indicate unusual network events/errors **Some alarms usually included w/analyzer:** • Excessive broadcasts • Utilization threshold exceeded • Request denied • Server down
Statistics	Display network performance: Current packet-per-second rate/network utilization rates • Admins use these to ID gradual changes in ops/sudden spikes in patterns • WS int can provide many statistical displays • Also offers: Summary page/protocol hierarchy listing/protocol-specific info

Placing a Protocol Analyzer on a Network

- Analyzer can only capture packets it can see
- Network connected w/hubs: Can place analyzer anywhere: All traffic fwded out all ports
- Connected w/switches: Sees multicast/broadcast/directed/initial packets sent to addresses: No ports ID'd yet

3 methods for analyzing switched networks:

1. Hubbing out/tap device
2. Port redirection
3. Remote Monitoring (RMON)

Hubbing Out	Place hub bet device of interest/switch: Connecting analyzer to hub: • Can view all traffic to/from server • Tap device splits signal from a single switch port: All traffic gets copied into 2 ports • One for target device/one for analyzer **Tap required to analyze full-duplex comm:** • Duplicates all RX (receive)|TX (transmit) in single RX channel into analyzer
Port Redirection	Switches can be config to redirect [copy] packets traveling through ports **Port spanning/mirroring:** • Placing analyzer on destination port: Can listen to conversations through port of interest
Remote Monitoring	**RMON: Remote Monitoring: Uses SNMP: Simple Network Management Protocol:** • Collects traffic at remote switch/sends data to another device • Management device: Decodes traffic/data and can display entire packet decodes

Post 2

Thursday, January 24, 2019 11:44 PM

WCNA: IP ADDRESSING: CH 2

IP Addressing Basics
Uses 3-part addressing scheme:

- **Symbolic name**
- **Logical numeric address**
- Physical numeric address

Symbolic name	Human-recognizable name: www.name.com: Domain names
Domain name	Must correspond w/at least 1 unique numeric IP **Domain:** Any collection of computing devices on network • Only point to numeric addresses

IPv4: Logical numeric address: Set of 4 #'s separated by periods: 172.16.1.10
- Must be smaller than 256 to be represented in bin
- **Range: 0 – 255:** Lowest/highest values: 8-bit string
- Bytes expressed as octets in networking

IPv6: Address consist of 128 bits: IPv4: 32: Hex values: Base16: 21da:00d3:0000:2f3b:02aa:00ff:f328:9c5a
- Divided into 8 blocks/words: 4 chars each
- Separated by colon
- Group of continuous 0's can be 'compressed' by leaving separators
  - Example: 21da:d3**:0**:2f3b:2aa:ff:fe28:9c5a

IPv4/6: Function at network layer: Assigns unique #'s to each network interface: IP also ops on network layer
Physical address: 6-bytes: Burned into firmware

1st 3 bytes	**OUI: Organizationally Unique Identifier** • ID manufacturer/whatever int in use
Final 3 bytes	Assigned by manufacturer: Give int on network address • **Ops as subset of Data Link Layer** ○ **MAC: Media Access Control** ○ MAC address • **LLC: Logical Link Controller:** Other subset of Data Link: ○ SW enables int to create point-to-point connection w/other network int ○ Same physical cable segment

ARP/RARP

ARP	**Address Resolution Protocol:** Translates IP's to MAC addresses
RARP	**Reverse Address Resolution Protocol:** Translates MAC layer addresses into IP's
Intermediate host	Networked device relays traffic from origin/destination • Data moves through intermediate hosts: Moves bet pairs of network ints • Each source/destination pair reside on same physical network
Hop	Data frame crossing a router

IPv4 Classes: 5 classes:
1. Class A: n.h.h.h
2. Class B: n.n.h.h
3. Class C: n.n.n.h
4. Class D: Multicast comm
5. Class E: Experimental use

n	Portion of address: ID's networks by #

h	Portion of address: ID's hosts by #

More than 1 octet part of network/host portion: Bits concatenated to determine address: 10.12.120.2 valid Class A

- **Network portion:** 10
- **Host portion:** 12.120.2: 3 octet number

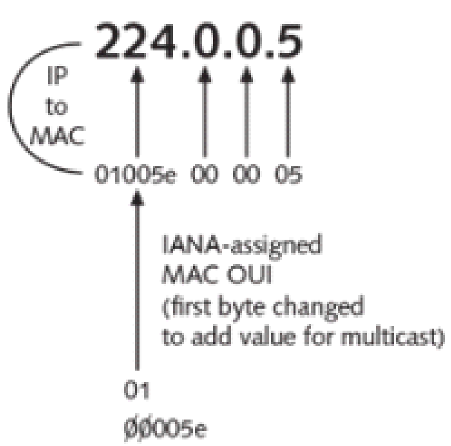
Class D/E:

- D: Multicast comm: Single address may be associated w/more than 1 network host machine
- Useful when info broadcast to selected group of recipients
- Videoconferencing/teleconferencing

Multicast addresses	Good when class of devices (routers) must be updated w/same info on regular basis • Some routing protocols use multicast to propagate routing table updates **Class E:** Reserved: Experimental use

2 addresses reserved from each IPv4 range

Broadcast	**Address all hosts on network must read** • Originated: Era w/limited scope networks • Traffic seldom fwded from 1 network/other • Routers separating networks filter as way of managing traffic/BW consumption • Usually purely local form of traffic **Packet Structure: 2 destination address fields:** **1. DLL destination** address field **2. Destination network** address field
Multicast	**Address Structures:** Host uses service: Employs multicast: RIPv2: router updates • Registers to listen on it: Also own unique host address • Host: Informs gateway: Router/device will fwd traffic to host network ○ Registering for service: Device will fwd multicast traffic **Registration:** Informs NIC to pass packets sent to IP stack: Contents read • Tells gateway to fwd onto network: Where listening int resides • No registration? Traffic ignored/unavailable **ICANN: Internet Corporation for Assigned Names/Numbers:** • Allocates multicasts on controlled basis • Formerly under **IANA: Internet Assigned Numbers Authority** DLL destination address based on network layer multicast **Address 0x01-00-5e-00-00 obtained w/calculation** 1. Replace 1st byte: Corresponding 3-byte OUI: 224 replaced w/0x00-00-5e 2. Change 1st byte: Odd value: 0x00 to 0x01 3. Replace 2nd-4th bytes w/dec equivalents: 224.0.0.5



IPv4 Networks/Subnet Masks

If 2 network ints on same network: Can comm directly w/1 another at MAC layer through subnet mask

Subnet mask	Special bit pattern that 'blocks off' network portion of IPv4 w/all-1's pattern

Default subnet masks for IPv4:

Address Type	Subnet Mask
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

255: 11111111: Thus each 255 masks off 1 on of the octets that makes up the network portion of the address

- Subnet mask: ID's network portion of the IP address

Class C Subnet

Network Bits	Subnet Mask	Number of Subnets	Number of Hosts
/24	255.255.255.0	0	254
/25	255.255.255.128	2(0)	126
/26	255.255.255.192	4(2)	62
/27	255.255.255.224	8(6)	30
/28	255.255.255.240	16(14)	14
/29	255.255.255.248	32(30)	6
/30	255.255.255.252	64(62)	2

Subnets/Supernets:

Subnetting	Represents stealing/borrowing bits from host portion: Using them to create multiple regions w/in single address
Subnet mask	Larger than default mask for address in use: Divides single network into multiple subnetworks

Class B: 255.255.0.0 subnet mask of 255.255.192.0: Indicates stealing 2 bits from host portion to use for subnet ID

- 192 dec: 11000000 bin: Shows upper 2 bits used for network portion

Network prefix	ID's # of bits in IP: From left: Add'l 2 bits of subnetting represent bits borrowed from host portion of IP
Extended network prefix	Entire network address including prefix/subnetting bits

Machine on 1 subnet wishes to comm w/machine/other subnet: Traffic fwded from sender to nearby gateway to send msg

Supernetting	Takes opposite approach: Combines contiguous network addresses • Steals bits from network portion: Uses to create single larger address space for host

7 varieties of subnet masks: Depends on how you want to implement address segmentation scheme

CLSM	**Constant-Length Subnet Masking** • Subnet includes same # stations: Represents simple division of address space • Subnetting into multiple equal segments **Good if:** All segments must support roughly same # of devices
VLSM	**Variable-Length Subnet Masking** • Permits single address to be subdivided into multiple subnets • Subnets don't need to all be same size Good if: 1/2 segments require larger #'s of users: Others require lesser # • Different subnets may have different extended network prefixes reflecting varied layouts/capacities

CIDR: Classless Inter-Domain Routing: IPv4

- Sets network-host ID boundary where it wants in way that simplifies routing across resulting address spaces
- Allows IPv4 addresses from Class A/B/C to be combined/treated as larger space/subdivided as needed
- Reduction in # of individual Class C addresses that must be recognized

CIDR: Limitations:

All addresses must be contiguous

- Use of standard network prefix notation for addresses makes it tidy/efficient

When multiple addresses aggregated:

- Requires them to be in numerical order: Boundary bet network/host portion can move to reflect aggregation

When address aggregation occurs:

- Blocks work best when they come in sets >1/= to some lower-order bit patterns that corresponds to all 1's
  - Namely groups 3/7/15/31/etc.
    - Makes possible to borrow corresponding # of bits (2/3/4/5) from network portion of block
    - Uses them to extend host portion

To use CIDR address on any network: All routers in routing domain must understand CIDR notation

- Not a problem for most routers built after August 2006

Prefix notation:

Class A	/8
Class B	/16
Class C	/24

192.168.5.0/27: Class C: /24 +3 added bits to make /27: Mask: 255.255.255.224

Public vs. Private IPv4 Addresses:

Private IPv4 Address Info:

Class	Address Range	Networks	Total Private Hosts
A	10.0.0.0-10.255.255.255	1	16,777,214
B	172.16.0.0-172.31.255.255	16	1,048,544
C	192.168.0.0-192.168.168.255.255	256	65,024

NAT	**Network Address Translation:** Converts public to private addresses
End-to-end	Connection extends all the way from sender/receiver while active

Organizations need public IP's only for 2 classes of equipment:

- **Devices permit orgs to attach networks to Internet**
    - Include: External ints on boundary devices (routers/proxy servers/firewalls)
    - Help maintain network perimeter bet outside/inside
- **Servers designed to be accessible to Internet:**
  - Public web/email/FTP/whatever app layer services an org may want on Internet

Reverse proxying	Permits proxy server to front for servers inside boundary • Advertises only server's address to outside world • Fwds legitimate requests to internal servers for further processing

L3 routing decisions: Typically SW: Slow compared to decisions made by L2 switches

- **ASICS: Application-Specific Integrated Circuits:** Switches make decisions w/this HW
- **L3 switch:** Implements logic from SW into own ASICs: Faster routing
- L3 switching: Allows partition on a large network into many smaller subnets: No loss of performance

Constraints:

- Min size of routing tables/time required for network to converge: Update table for changes in topography
- Max flexibility/facilitate management/troubleshooting

I**Pv6 Network/Host Portions:** Shorthanded notation

- /dec # after address: Left most contiguous bits of address part of network prefix: 1090:: /60
  - 0's: Representation of single 16-bit group: Can't be omitted
  - If < 4 hex digits in 16-bit group: Assumed to be 0's

Scope Identifier: IPv6 multicasts use scope identifier:

- 4 bit field: Limits valid range for multicast to define portion of Internet to which multicast group pertains

Interface Identifier: IPv6 requires every network int to have its own identifier

- EUI-64 format: Unique 64-bit int: HW vendors tend to use modified EUI-64 fmt

EUI-64 format:

- 1st 24-bits: Name of card's manufacturer: Maybe also individual production run
- 2nd 24-bits: Chosen by manufacturer to ensure uniqueness among cards
    - Required to create: Unique int ID: Padding #
    - Specific 16-bit pattern: 0xfffe (:fffe:) bet 2 halves of MAC to create 64 bit #

Global/Local Individual/Group bits in IPv6 Int ID

Bit 6	Bit 7	Meaning
0	0	Locally unique, individual
0	1	Locally unique, group
1	0	Globally unique, individual
1	1	Globally unique, group

Special Addresses: 2 addresses reserved for special use

1. Unspecified address
2. Loopback address

<table>
<tr><td>Unspecified</td><td>All 0's: Can be represented as :: in notation
• Address that is no address
• Can't be used as destination address</td></tr>
<tr><td>Loopback</td><td>Allows host on network to check operation of its TCP/IP
• IPv6: All 0's except last bit set to 1 – ::1
• Diagnostic tool for local use only
• Can't be routed/used as either source/destination address for packets actually on a network
When packet sent w/loopback as destination:
• Stack on sending host sends msgs to self</td></tr>
<tr><td>Multicast</td><td>Send identical msg to multiple hosts: Subscription based
• Nodes must announce they want to receive multicast traffic
Format:
<table><tr><td>8</td><td>4</td><td>4</td><td>112 bits</td></tr><tr><td>11111111</td><td>FLAGS</td><td>SCOPE</td><td>GROUP ID</td></tr></table>• 1st byte: 8 bits: Set to all 1's: 0xFF indicating multicast
• 2nd byte: Divided into 2 fields
○ Flags: 4 bits long
○ Scope: 4 bits long
• Remaining 112 bits: Define identifier for multicast group
Flags field: Treated as set of 4 individual 1-bit flags:
• Reserved: Future use/must all be set to 0
• 4th flag: Set to 1 when multicast address temp/transient address
• If address well-know: Flag set to 0
Scope field: Limits range of addresses over which multicast subscriber group is valid
Transient/temp: Established then abandoned
• Analogous to way TCP might use unassigned port for temp session
• Group ID: Meaningless outside scope
2 groups w/Identical group ID's but different scopes:
• Completely unrelated when T flag set to 1
• Well-known: T bit set to 0: Assigned to entities as all routers/DHCP servers
• In combo w/scope field:
○ Allows multicast to define all routers on local link/all DHCP servers on global Internet
Although last 112 bits assigned to Group ID:
• 1st 80 bits set to all 0's
• Remaining 32 bits: Must contain whole non-zero part of Group ID</td></tr>
<tr><td>Neighbor Solicitation</td><td>Special multicast called solicited node address used to support NS</td></tr>
</table>

Anycast	Packets go to nearest single instance of address • Near in terms of router's view of distance • Addresses functions commonly deployed on Internet at multiple locations ○ Examples: DHCP servers/routers Rather than using multicasts to send packets to all NTP (Network Time) servers on local link: • Node can send packet to anycast for all NTP servers • Same fmt as unicast
Unicast Addresses	Sent to 1 network int: Basic address in IPv6 space • 64-bit int ID: In least significant bits • 64-bit network portion: Most significant bits

Aggregatable Global Unicast Address: Aid in routing/admin of addresses: Particular kind of unicast
- Can be combined w/other addresses into single entry in routing table
- Leftmost 64 bits of address (FP/SLA ID): Explicit fields for easier routing

3	13	8	24	16	64 bits
FP	TLA ID	RES	NLA ID	SLA ID	INTERFACE ID

FP: Format Prefix: 3 bit identifier to show which part of IPv6 address space address belongs to
- All aggregatable addresses: Must have 001 in this field
- **TLA ID: Top-Level Aggregation ID:** 13 bit field: Allows 2^13 top-level routes
  - 8K highest-level groups of addresses
- **RES:** 8 bits: Reserved for future use
- **NLA ID: Next-Level Aggregation:** 24 bits: Allows entities controlling any 1 of TLA's to divide address blocks
  - Likely ISP's/Large Internet entities
  - Smaller ISP's subdivide blocks further: If enough space in NLA field permits
- **SLA ID: Site-Level Aggregation ID:** 16 bits: Permits creation of 65,535 addresses as flat address space
  - Users can set up hierarchically/use portion of address to create 255 subnets w/255 addresses each
- **Interface ID field:** Same EUI-64 fmt int ID

Link-Local/Site-Local Addresses

IPv6 Link-Local Address Format

10 bits	54 bits	64 bits
1111111010	0	INTERFACE ID

IPv6 Site-Local Address Format

10 bits	38 bits	16 bits	64 bits
1111111011	0	SUBNET ID	INTERFACE ID

Link-local address: 1st 10 (leftmost) bits set to 11111010:
- **Next 54 bits:** Set to 0's
- **Final 64 bits:** Represent normal int ID
- Router sees link-local prefix in packet: Knows it can ignore it

Site-local address: 1st 10 (leftmost) bits: 11111011:
- **Next 38 bits**: Set to 0's
- **Next 16 bits**: Contain subnet ID: Defines the site to which this address is local
- F**inal:** 64 bits represent standard int ID
- Allows packets to be fwded internally w/in site: Prevents packets from being visible on public Internet
- Site/link-local addresses are each assigned a 1/1024 portion of the IPv6 address space

NSAP Allocation: Network Service Access Point: 1/128 of IPv6 address space set aside for NSAP

- ATM, X.25, etc.. Typically set up point-to-point links bet hosts

Methods/tech that will allow IPv4/IPv6 hosts/networks to exist:

Teredo Tunneling	Designed to allow full IPv6 connectivity bet hosts on IPv4 • IPv6 datagrams encapsulated in IPv4 UDP packets • Then fwded even through NAT devices/routed on IPv4 Once packets received: • Encapsulation stripped: Runs on native IPv6 • Win7/10 native support
ISATAP	**Intra-Site Automatic Tunnel Addressing Protocol:** • Generates link-local IPv6 from IPv4 • Neighbor discovery on top of IPv4 • Uses IPv4 as virtual **NBMA: Non-Broadcast Multi-Access** network on DLL • Allows use of multicast packets w/out IPv4 needing to support msging • XP/Win/Linux/Cisco IOS
6to4 tunneling	Facilitate IPv4 to IPv6 migration: • IPv6 packets sent over IPv4 networks: Internet w/out explicitly config tunneling • Can be used by individual IPv6 node/network When used by host: Computer must have global IPv4 connected • Must provide encapsulation of any IPv6 packets transmitted/decapsulation of IPv6 packets it receives Older tunneling tech: Meant to be transitional method/not perm
Rapid Deployment	Builds on mechanisms described for 6to4: • Enables SP's to deploy IPv6 unicast service to IPv4 sites • Uses stateless IPv6 in IPv4 encapsulation to transmit IPv4-only network traffic • Unlike 6to4: 6rd provider uses IPv6 prefix in place of fixed 6to4 prefix
NAT-PT	**Network Address Translation-Protocol Translation:** • Meant to be used for IPv4-to-IPv6 transition • Allowing IPv6 packets to be sent back/forth across IPv4 networks • Now considered "historic" **Network Address Port Translation-Protocol Translation (NAPT-PT):** Similar to NAT-PT

Post 3

Thursday, January 24, 2019 11:45 PM

WCNA WIRESHARK NOTES CH 1 TO 6

How WS Captures Traffic:
GTK: GIMP Graphical Toolkit [Dissectors/plugins/display filters]: Core Engine

dumpcap [Capture engine]	**Wiretap Lib**
libpcap/WinPcap [Capture filter]	Drive
Network	

Capture process: Link Layers: Rely on NIC/LL drivers to send/receive packets
2 link-layer drivers commonly used:

1. **WinPcap:** Windows hosts
2. **libpcap:** Linux/OS X hosts

dumpcap	Launches actual capturing: Frames pass from network to LL driver into WS's engine • Defines how process runs/stop conditions **.pcapng: Packet Capture Next Generation:** Default trace file format • Save metadata w/file: Comment inside trace files Capture filters based on **BPF: Berkeley Packet Filtering** syntax
Core engine	Capture engine passes frames to core engine: • Tons of dissector support: Translates bytes: Human-readable frames • Dissectors: Break apart fields in frames/perform analysis on contents
GIMP graphical toolkit	**GNU Img Manip Program graphical toolkit:** • Cross-platform int for WS: Moves seamlessly bet diff platforms
Wiretap Lib	Opens Saved Trace Files: I/O functions for saved ones: Delivers frames to core engine

Analysis Session: Steps

1. Who is talking in trace file?
2. What apps in use?
3. Filter convo of interest
4. Graph IO: Look for drops in throughput
5. Expert to look for problems
6. Determine RTT to ID path latency

Packets vs. Frames

Frame	Comm from MAC header-trailer: All comms bet devices use frames **eth0 headers:** Well implemented tech: Not lots to analyze/troubleshoot usually • WS: Adds frame section to provide extra info **Packet Details Pane**: Frame section at top: **Expand:** Time/coloring/other info added to actual frame **Frame starts 2nd line: Ethernet II:** Frame section only contains info about it not contents of (metadata)
Packet	Sits inside a MAC frame: Begins at IP header | Ends before MAC trailer **Packet analysis:** Network analysis
Segment	Follows TCP header: May include HTTP header/just data During establishment of TCP connection: Each peer shares MSS: Max Segment Size value

Follow HTTP Packet Through Network:
What would you see at client?

Capture Point 1	**MAC:** A -> B | 10.1.0.1 -> 74.125.224.143 TTL: 64 TCP Header GET / eth0 trailer • Devices: Can only send to HW address of local machines in MAC headers • MAC header: Stripped off by 1st rtr along path: Used to get packet to next hop • IP header: Addressed from 10.1.01 (client) – 74.125.224.143 (server) **Analyst view:** Ethernet header of client's GET request addressed to local router's MAC address

Capture Point 2	**MAC:** A -> B | 10.1.0.1 -> 74.125.224.143 TTL: 64 | **True switches:** Don't affect contents of frame Switch 1: Would look at dest MAC: Determine if host connected to 1 of switch ports • When switch finds port associated w/MAC address B: Fwds frame Analyst view: Frame that matches the frame we saw at point 1
Capture Point 3	**MAC:** C -> D | 10.1.0.1 -> 74.125.224.143 TTL: 63 • On receipt of frame after checking for corruption: Addressed to MAC's address • Rtr strips eth0 header: Examines dest IP: Packet now/not frame • Consults r-tables to see if knows what to do If router doesn't know how to get dest. address: • Drops packet: Msgs originator indicating problem • Capture error msgs w/WS: Detect rtr unable to fwd packets to dest. If router has info required: • Decrements IP header TTL field value by 1 (hop) • New eth0 header to packet before sending to rtr/NAT device **Analyst view:** New eth0 header: IP header TTL value decreased by 1
Capture Point 4	**MAC:** E -> F | 67.2.0.1 -> 74.125.224.143 TTL 62 • Rtr/NAT device: Same routing as before: Fwd packet ○ Device changes source IP address (NAT)/source port # ○ Makes note of original source IP/port # • Device associates info w/newly assigned outbound IP/port # **Analyst view**: eth0/IP header TTL value decreased by 1: Source IP/port # changed
Capture Point 5	**MAC:** E -> F | 67.2.0.1 -> 74.125.224.143 TTL: 62 • Same frame as CP 4: Switches shouldn't alter contents of it

Where Traffic Matters:

- Point 1/2/3: Can't determine MAC of server
- Point 3/4/5: Can't determine MAC of client
- Point 5: Can't tell actual IP of client

Beware of Default Switch Fwding: Switches fwd frames based on MAC

- If you connect WS to either switches: Wouldn't see traffic bet client/server
  - Switches: Only fwd broadcast/multicast/traffic: Don't alter MAC/IP of traffic

Resources	wiki.wireshark.org | ask.wireshark.org

Open Trace File: File > Open > .pcapng file

Main Menu

File	Export HTTP objects
Edit	Clear all marked/ignored packets/time references
Analyze	Create display filter macros/see enabled protocols/save forced decodes
Statistics	Build graphs/open statistics window for protocols
Tools	Build FW rules from packet contents: Access Lua scripting tool
Internals	View dissector tables/list of supported protocols

Summarize Traffic: Packet List Pane

WS has 3 panes:

1. Packet List
2. Packet Details
3. Packet Bytes

Columns

No.	Each frame assigned #: Traffic sorted on column from low-high
Time	When each frame arrived compared to 1st
Source/Destination	Highest layer address avail in each frame: Some only have MAC (ARP)
Protocol	Displays last dissector applied in frame: See which apps in use

Re-order columns: Right-click column headings: Hide/Display/Rename/Rem
Use packet coloring: ID types of traffic/spot problems faster
Status Bar:

Expert info	Colored: Show highest lvl in window: Can alert to concerns in trace file/packet comments
Annotation	Add comments to trace file
1st Column	Field/Capture/Trace file info: Changes depending on what's highlighted/running live
2nd Column	Packet Counts (Total/Displayed)
3rd Column	Profile: Created to customize environment

Add Columns to Packet List Pane:

1. Right-click: Apply as column
2. Edit > Preferences > Columns > Add > **ip.ttl**/field name > Occurrence # 0 to view all occurrences in field
    - Title entry to type new heading

Export Column Data: File > Export Packet Dissections > CSV > Export only Summary info
Dissect WS Dissectors: Core engine: Glue code that holds other blocks together

Frame	Examines/displays trace file basic info: Timestamp on each frame: Hands frame to eth0 dissector
eth0	Decodes/displays fields: Ethernet II header: Based on contents of Type field: Hands to next dissector • Example: Type field value 0x0800: IPv4 header • When we rem eth0 frame from dissection: Packet
IPv4	Decodes fields of IPv4 header: Based on contents of protocol field: Hands packet to next dissector • Example: Protocol: TCP (6) – 6 indicates TCP
TCP	Decodes fields of TCP header: Based on contents of port fields: Hands to next dissector
HTTP	Decodes fields of HTTP packet: No embedded protocol/app inside: Last dissector applied to frame

Analyze Traffic: Non-Standard Ports: Apps running non-standard ports: Always concern:

- Traffic runs over non-standard port # WS recognizes by other app? May apply wrong dissector

Manually force dissector traffic: 2 reasons why: Manually force dissector onto traffic

1. WS applies wrong dissector b/c non-standard ports associated w/it already
2. WS doesn't have heuristic dissector for traffic type

Right-click on undissected/incorrectly dissected packet in Packet List pane > Decode As > Transport tab: Choose dissector
Port # not recognized? Uses heuristic dissectors to decode data into recognizable protocol/app
How Heuristic Dissectors Work: Each looks for recognizable patterns in data to figure type of com contained in packet
If dissector doesn't recognize anything:
Returns failure indication to WS: Hands off data to next heuristic dissector: Continues hand offs until:

1. Returns indicator of success/decodes traffic
2. WS runs out of dissectors

Adjust Dissections w/App Preference Settings:
If you know certain traffic runs over non-standard port: Add port to protocol pref settings

- Edit > Preferences > + Protocols > HTTP/w/e) > Add port to list

Determine if WS unable to apply dissector to some frames: Statistics > Protocol Hierarchy > Data under TCP/UDP sections
Change how WS displays certain traffic types:

User Int Settings	Edit > Preferences > User Int | Basic preferences for int
Name Resolution Settings	Edit > Preferences > Name Resolution | View/change way WS deals w/MAC/port/IP address resolution **manuf file:** MAC name resolution: Resolves 1st 3 bytes: MAC/OUI **services file:** Transport name resolution: Info column of Packet List Payne Host name resolution: Enable network resolution • Enabling w/out creating hosts file to use • Can cause WS to send DNS PTR queries to obtain host names • Extra work for DNS server • View > Name Resolution (temp)

Define Filter Expression Buttons: Edit > Preferences > Filter Expressions | Save fav display filters as

buttons
Set Protocol/App Settings: Edit > Preferences + Protocols: View all protocols/apps w/editable settings: Right-click faster

Allow subdissector to reassemble TCP streams	Default: Enabled: Can cause problems when analyzing HTTP traffic If HTTP server answers client request w/response code (ex. 200 OK) • Includes some requested file in packet: WS doesn't display response code • Can disable TCP reassembly preference until exporting files
Track # of bytes in flight	**Bytes in flight:** Bytes sent across TCP connection but not acknowledged yet • If # seems to hit 'ceiling': TCP setting may limit data flow capabilities When enabled: New section appended to TCP header in Packet Details Pane • New field not displayed until after connection established
Calculate convo timestamps	Tracks time values w/in each separate TCP convo • Obtain values on 1st frame in single/previous frame in TCP convo • New section appended to TCP header in Packet Details pane

Profiles: Dirs contain WS config/support files loaded: Coloring rules: Highlight suspicious traffic w/known sigs
Create new profile: Right-click Profile column (Status Bar) > New | Edit > Config Profiles
Locate Key WS Config Files: Help > About WS > Folders
Global Config Dir

preferences	Name resolution/filter expression buttons/protocol settings
dfilters	Display filters for profile
cfilters	Capture filters for profile
colorfilters	Coloring rules for profile

Time Columns: Spot Latency Problems:
Latency: Measurement used to define time delay

- Always some latency: Excessive can be from problems along path/endpoints

Time/Info column: Used to detect 3 types of latency:

1. Path
2. Client
3. Server

Path	**AKA: RTT: Round Trip Time:** • Often measures how long it takes for packet to be transmitted/response received • Can't tell if slow performance inbound/outbound • Can be infrastructure: Enterprise rtr prioritizing QoS/BW bottleneck on network
Client	**Users:** Apps/Lack sufficient resources: Natural 'human-induced' latency • Seen least often: Most apps put load on server side of comms
Server	Slowly replying incoming requests: Lack of processing power/faulty/poorly written app • Consulting another server to get response info (multi-tiered/middleware arch)

Detect Latency Problems: Change Time Column Setting:

- Default: Sec since beginning of capture
- WS: 1st packet arrival as 0.00000000
- Time column value for each after: How much later arrived
- Spot high delta times: Time from end of 1 packet to end of next:
    - View > Time Display Fmt > Seconds Since Previous Displayed Packet

Detect Latency Problems w/New TCP Delta Column: Edit > Preferences > + Protocols > TCP

- Add column for TCP delta time value: Expand TCP header
- Right-click Time since previous frame in TCP stream: Apply as column

Rename column: Edit > Column Details
Usually normal

.ico file requests	By browser to put icon on tab

SYN packets	Sent to establish new connection w/TCP peer: May begin capturing: Ask usr to connect to web server • Delay before 1st packet of TCP connection
FIN, FIN/ACK, RST RST/ACK	Sent implicitly/explicitly to term connection: Browsers send when click on new tab • No recent activity to site: Browsing session config auto close after page loaded
GET requests	Can be generated when usr clicks link to req next exchange: May be launched by BG processes: No priority
DNS queries	Sent various times during web browsing: Page hit numerous hyperlinks loads at client
TLSv1 Encrypted Alerts	Often before connection close process (TCP RST): Alert likely TLS Close request

Capture Options:

Int List	Select 1/more ints [multi-adapter capture]
Manage Ints	Add new local/remote ints
Capture Filters	Save to multiple files/set ring buffer/set auto-stop condition based on # of files
Stop Capture	Auto-stop condition based on # of packets/quantity of data/elapsed time
Green WS Icon	Live capture (otherwise blue)

Ideal Starting Point: Capture traffic at/near host experiencing performance issue
- See traffic from host's perspective: Can detect round trip latency times/packet loss/errors/etc..

Options

Directly on complaining host	Good if allowed to install packet capture SW on host
Span host's switch port	If switch above usr supports port spanning/you can config switch: • Setup switch to cp all traffic to/from usr's switch port down your WS port • Won't fwd LL error packets
TAP	**Setup Test Access Port:** • Full-duplex devices installed in path bet host of interest/switch • Default: Fwd all network traffic: Including LL errors

Capture Traffic on Wireless Network: AirPcap adapters:
- Designed to capture all types of WLAN traffic: Apply WLAN decryption keys: Add metadata about captured frames
- 802.11 control/mgmt/data frames: Monitor mode (RFMON): Enables adapter to capture all traffic w/out AP association

Can be config to affix **PPI: Per-Packet Info**/RadioTap header to each WLAN frame
- Headers contain freq. info/signal-noise lvl/strength/moment-loc of capture/etc.

Multi-Adapter Capture: 2/more ints at a time
Capture to file set: File > File Set > List Files : Individually linked files that can be examined
Capture > Options > Check box next to int > Enter path/file name for file set in capture file > Use Multiple Files
Cascade Pilot: Handles large trace files: Graphing/reporting capabilities missing in WS: Integrates tightly
- File Sets/Ring Buffer use to capture/save minimally sized files

Detect when WS can't keep up: Dropped: x will appear on Status Bar's center column
- Trace file will contain numerous **ACKed Lost Segment/Previous Segment Not Captured indications**

Capture filter: Capture > Options > Capture Filter column > Double click int line
- **Red background:** Filter can't be processed

Protocol Filters

arp	ARP traffic including gratuitous ARP's/requests/replies
ip	IPv4 traffic: Packets that have IPv4 headers embedded in them (ICMP destination unreachable)
ipv6	IPv6 traffic including IPv4 packets that have IPv6 headers embedded in them [Teredo/6to4/ISATAP]
tcp	TCP-based connections

Application Filters

bootp	DHCP traffic

dns	DNS traffic including TCP-based zone xfers/standard UDP-based DNS requests/responses
tftp	TFTP: Trivial FTP traffic
http	HTTP cmds/responses/data xfer packets: • Doesn't display TCP handshake packets/TCP ACK/TCP connection teardown packets
icmp	ICMP traffic

Field Existence Filters

bootp.option.hostname	DHCP traffic that contains a host name
http.host	HTTP packets w/HTTP host name field: Sent by clients when they send req to web server
ftp.request.command	FTP traffic that contains a cmd: USER/PASS/RETR cmds

Characteristic Filters

tcp.analysis.flags	All packets that have any TCP analysis flags associated w/them • Includes: Indications of packet loss/retransmissions/zero window conditions
tcp.analysis.zero_window	Packets that are flagged to indicate sender has run out of receive buffer space

Display Filter Error Detection Mechanism:

- **Red BG:** Filter won't work
- **Yellow BG:** Warning filter may not work as desired
- **Green:** Filter properly formed: Be careful: WS doesn't do a logic test

Filter Comparison Operators

== or eq	IPv4traffic from 10.2.2.2 • Example: ip.src == 10.2.2.2
!= or ne	TCP traffic from any port except 80 • Example: tcp.srcport !=80
> or gt	Packets that arrive more than 1s after previous one in trace file • Example: frame.time_relative >1
< or lt	Display when TCP receive window size less than 1460 bytes • Example: tcp.window_size < 1460
>= or ge	DNS response packets that contain at least 10 answers • Example: dns.count.answers >=10
<= or lt	Packets less than 10 in IP TTL field • Example: ip.ttl < 10
contains	All HTTP client GET requests sent to HTTP servers • Example: https contains "GET"

WS: 15 default display filters: Can use as reference to make new ones

- **Filter button > Display Filter button:** Watch out for default values: May mismatch w/network

2 methods to filter HTTP traffic

1. http
2. tcp.port==xx [xx = HTTP port in use] -> more effective
   - Displays all web browsing traffic/TCP connection setup/mgmt frames

Determine why DHCP Display filter doesn't work: Use bootp syntax

Filter Traffic to/from Single IP/Host

ip.addr==10.3.1.1	Frames that have 10.3.1.1 in IP source/dest. address field
!ip.addr==10.3.1.1	All frames except
ipv6.addr==2406:da00:ff00:6b16:f02d	All frames to/from
ip.src==10.3.1.1	Traffic from
ip.dst==10.3.1.1	Traffic to
ip.host==www.wireshark.org	Traffic to/from IP that resolves to address

Filter Traffic to/from ranges

ip.addr > 10.3.0.1 && ip.addr < 10.3.0.5	Traffic to/from 10.3.0.1/3/4

(ip.addr >=10.3.0.1 && ip.addr <= 10.3.0.6) && !ip.addr==10.3.0.3	To/from 10.3.0.1/2/4/5/6 10.3.0.3 excluded
ipv6.addr >= fe80:: && ipv6.addr < fec0::	IPv6 addresses start 0xfe80-0xfec0

Quickly Filter on a Field in a Packet: Right-click > Apply as Filter/Prepare a Filter
Prepare a filter: When you want to change filter/check syntax before applied

- Right-click Request URI line > Prepare a Filter > Selected
- WS places http.request.uri==”/prod/scripts/mbox.js” in display area
    - Doesn’t apply filter to traffic
    - Change to http.request.uri contains “jpg” > Apply

Right-click again to use “…” Filter Enhancements

- Options that begin w/… append to existing display filter

Example: Right-click Request Method

GET > Selected	**http.request.method == “GET”** • Replaces current display filter/displays all HTTP packets that contain GET
GET > Not Selected	**!(http.request.method == “GET”)** • Will replace current display filter: Display packets EXCEPT HTTP packets w/GET request
GET > …and Selected	**(tcp.port==80) && (http.request.method ==”GET”)** • Displays packets to/from port 80 that contain GET
GET > …or Selected	**(tcp.port==80) || (http.request.method ==”GET”)** • Displays packets to/from port 80 & HTTP GET
GET > …and Not Selected	**(tcp.port==80) && !(http.request.method ==”GET”)** • Displays all traffic to/from port 80: Not any HTTP packets that contain GET
GET > …or Not Selected	**(tcp.port==80) || !(ip.src==10.2.2.2)** • Displays packets to/from port 80: Any traffic not 10.2.2.2

Filter on Single TCP/UDP Convo:

- Right-click TCP/UDP packet in Packet List pane > Conversation Filter > TCP/UDP > Follow TCP/UDP stream
- **Extract convo w/WS:** Statistics > Conversations
    - Extract based on Stream index # (in TCP header)

Filter convo: Right click > Conversation Filter > TCP
Follow Stream: Right-click > Follow TCP/UDP Stream
Filter convo from WS statistics: Right click > Apply as Filter/Prepare a filter/Find a packet/Colorize conversation

- Statistics > Conversations > Sort column
- Under TCP/UDP tab: A: Refers to both columns labeled Address A/Port A
- Also Statistics > Endpoints

Logical Operators:

&& or and	View all IPv4 traffic from x that is to/from port 80 • Example: **ip.src.==10.2.2.2 && tcp.port==80**
|| or or	View all TCP traffic to/from ports 80-443 • Example: **tcp.port==80 || tcp.port ==443**
! or not	View all traffic except ARP • Example: **!arp**
!= or ne	View TCP frames that don’t have TCP SYN flag set to 1 • Example: **tcp.flags.syn !=1**

Use Regex w/”.”: Can use regex w/matches of the operator to represent a str w/vars

- “.” represents any char except line break/carriage return: Must esc with \

Filters to spot time delays: Filter on Large Delta Times: (frame.time_delta)

- Loc in Frame section of each packet
- Can create filter for large values
- To set filter for delays over 1s: frame.time_delta > 1
- Looks at ALL packets in trace file to display time from 1 packet to end of next

Troubleshooting UDP connection? File > Export > Specified Packets > Save new trace file > frame.time_delta

Create Filter Expression Button: Type display filter in area > Save No limits to # of Filter Expression Buttons you can create
Edit Filter Expression Area in preferences file: Help > About WS > Folders > Personal Configs link > preferences file just txt file
Color/Export Interesting Packets
ID Applied Config Rules:

- WS auto colors packets based on default set of rules: Can quickly ID packet types on colors
- Expand Frame section of packet > Coloring Rule Name/Coloring Rule Strings
- Maintained in textfile called colorfilters: Can be edited

Turn off Checksum Error Coloring Rule:

- If you have TCP/UDP/IP checksum validation enabled: Capturing on host that uses task offload
    - Checksum Error coloring rule will create false+ coloring on trace file
    - Valid checksums applied by network int card before frame sent on network
    - WS captures copy of packets before this is appended to frames
    - Disable it

Disable Individual Coloring Rules: Coloring Rules button > Click coloring rule > Disable
Disable All Packet Coloring: View > Colorize Packet List or Colorize Packet List button on toolbar
Build coloring rule to highlight delays: View > Coloring Rules > New > T-Delays > frame.time_delta > 1

- Click BG button/Foreground button > OK

Right-Click method to create coloring rule: Colorize w/Filter > New Coloring Rule
Colorize a convo temp: Right click convo in Packet List pane > Colorize Conversation > TCP > Color 1
Remove temp coloring: View > Reset Coloring
Export Packets that Interest you: File > Export Specified Packets

- Packets that don't match neatly in display filter: Filter > Export Specified Packets > Toggle Mark Packet
- Marked packets or First to last marked
- Not visible due to display filter? Captured radio button

Export Packet Dissections: File > Export Packet Dissections: 6 diff export options: Plain text/CSV fmts
IO Graph Interface

Graph area: X axis	Defaults to seconds
Graph area: Y axis	Set to logarithmic scale
Graph buttons	Enable/Disable graph lines
Filter area	Recall saved display filters w/Filter button
Graph Style	Line/impulse/fbar [floating]/dot formats

Network Conversation: Statistics > Conversations: Tabs: Bytes column (ex)

- WS refers to services file to replace port #'s w/app names: Uncheck Name Resolution to turn off

Relative Start	Rel Start: When convo started in trace file
Duration	How much time passed from 1st packet of convo to last packet in trace file

- Can uncheck Limit to display filter if you have a filter
- Follow Stream (TCP/UDP) to reassemble selected convo

Filter on Convo: Apply as Filter/Prepare a Filter > Options
Most Active Host: Statistics > Endpoints > IPv4/6 > Click twice on Bytes column > Tx Bytes from high to low
View Protocol Hierarchy: Statistics Protocol Hierarchy

- Right Click Filter/Colorize any Listed Protocol/App

Look for Suspicious Protocols/Apps/Data:
Hierarchy Statistics:

1. Distributed Computing Environment/Remote Procedure Call (DCE/RPC) under TCP
2. IRC: Internet Relay Chat traffic
3. TFTP traffic

Data listed under TCP/UDP window indicates WS couldn't apply dissector

- Doesn't recognize #/no heuristic dissector matched packets
- Allow subdissector to reassemble TCP streams before opening protocol

Decipher Protocol Hierarchy Percentages

- % Packets/Bytes column: Total traffic
- Numerical rounding may make sum slightly off

Export App/Host Traffic before Graphing: Statistics > IO Graph > bits/tick: Each tick 1 second on X axis setting

ip.addr	Graph button > Filter
ip.src	Unidirectional traffic: ip.src/ip.dst/ipv6.src/ipv6.dst
tcp.port/udp.port	Can click on graph buttons to disable them

Expert Infos Button on Status Bar:

- Red: Errors
- Yellow: Warnings
- Cyan: Notes
- Blue: Chats
- Grey: Details
- Green: Packet comments

Unreassembled **Indications in Expert:** Warnings tab > Edit > Preferences > TCP > Allow subdissector to reassemble..

Packet Loss/Recovery/Fault Trace Files

Previous Segment Not Captured	Indicates WS didn't see previous packets in TCP comm • WS tracks packet ordering based on TCP Sequence #'s • Typically occurs at internetwork device (switch/rtr)
ACKed Lost Packet	WS saw a TCP ACK: Data packet being acknowledged • Shouldn't be used for analysis
Duplicate ACK	TCP host receiving data from another host believes packet is missing
Retransmission	Sees 2 data packets w/same sequence #
Fast Retransmission	Data packet requested through duplicate ACKs within 20 ms of duplicate ACK

Async/Multiple Path Indications

Out-of-Order	Packet has lower TCP sequence # than previous packet • May indicate traffic flowed along diff paths to reach target
Keep-Alive	TCP host hasn't received any comm for a time
Keep-Alive ACK	Response to Keep-Alive packet

Receive Buffer Congestion Indications

Window Full	WS calc packet will fill avail receive buffer space of target: Can be last packet before 0 window condition
Zero Window	Sender is advertising TCP window size value of 0 • No receive buffer space available • App running on host that sent zero window packet not picking up data from receive buffer • Faulty app/overloaded host
Zero Window Probe	Host is trying to determine if target has any receive buffer space avail
Window Update	Sender advertising more TCP receive space than previous packet: Common recover from above

TCP Connection Port Reuse

Reused Ports	Should be investigated: Same port # as previous connection in trace file: Some apps may reuse ports • But usually sec scanning tools do as well

Possible Rtr Problem

4 NOPs in a Row	TCP value 0x01: No Op seen 4x in a packet: • Used to pad TCP header to end on 4-byte boundary: Should never see • Typically misbehaving rtr along path

Misconfig/ARP Poisoning

Duplicate IP Address Config	2/more ARP response packets offer diff HW addresses for same IP • Very unusual/can indicate ARP poisoning

Reassemble Traffic for Faster Analysis:
File > Export Objects > HTTP/DICOM/SMB to reassemble > Follow UDP Stream > Follow SSL Stream
Reassemble web browsing sessions: Right-click HTTP packet > Follow TCP Stream

Post 4

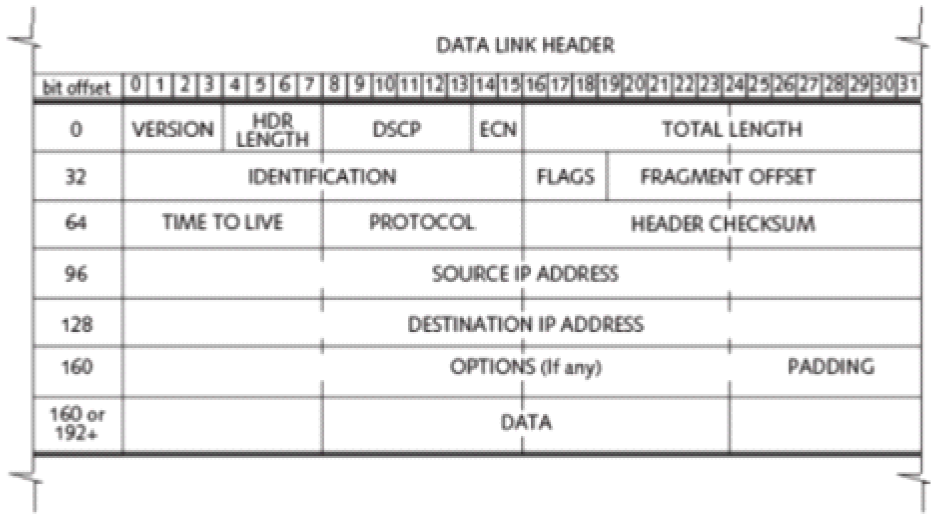
Thursday, January 24, 2019 11:45 PM

WCNA/NETWORK TRAFFIC ANALYSIS: CH 3 NOTES

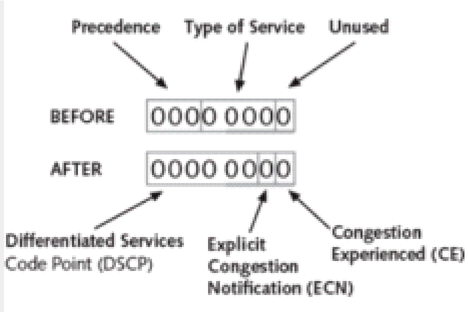
IP Packets/Packet Structures:
- IP works to transmit/deliver data bet devices
- Information is encapsulated into units called packets/datagrams
- Each packet contains portion of info being transmitted
- Each contains header to get packet where it needs to be

IPv4 Headers: RFC 791: 1981
- Each packet contains header followed by data field
- Can be bet 20-60 bytes in length w/total packet size: 65,535 bytes
- Most networks can't handle max sized packets: Usually 576 bytes

14 fields possible in header: 13 are required: 14th optional: Options
- Each field value provided in multiples of 4 bytes
- Size of header may vary
- Byte order is big endian: Most significant byte precedes least significant ones
- MSB: Most Significant Bit come 1st w/in individual bytes
- MSB 0 bit numbering
- Version field: Occurs in 4 most significant (leftmost bits) of 1st byte in header



Version Field	Value of 4: Indicates IP version 4
Header Length Field	**AKA: IHL field**: Length of IP header only • IP header can support options: May vary in length • Includes offset to data to make it fall on a 32-bit boundary value • **Min IHL: 5** [5*32 = 160 bits or 20 bytes] • **4 bit field:** Max value it can represent is 16*1=15 • Max length IHL 15*32=480 bits or 60 bytes
DSCP Field	**Differentiated Services Field Code Point** • **RFC 2474:** Method for differentiating services for network traffic • 6 high-order bits of the byte: Formerly 3 bit Precedence field • 1st bit of TOS field



Special marker: DSCP identifier:
- Traffic can be prioritized by end (node)/boundary (rtr) device
- Queued/fwded according to this value
- Rtrs that support DS tech: Handle traffic according to DSCP ident

Value may be assigned on flow of data by rtr/w/in data packets
- VoIP vs. e-mail traffic
- **RFC 2597/3246**
  - Referred to as AF: Assured Forwarding classes
  - Assigned by drop probability
  - Based on chance may be dropped in high traffic
  - # assignments indicate packet assured of being handled
  - Handled at each hop: per-hop behavior

Convert PHB AF to DSCP/vice versa
- **Afxy = (8*x) + (2*y) = DSCP**
- x/8 where x is DSCP value, y = remainder /2 or x/8 (r/2)
- 26 = 26 28/8 w/r of 2.2/2 -> AF 31

EF: Expedited Forwarding per-hop behavior

DB PHB: Delay Bound Forwarding:
- Strict bound on delay for packets through hop
- Traffic must be managed at source edge

Process of rtr dropping packet: Helps prevent DoS attacks
- If 2 negotiated DB traffic rate: All traffic from domain marked 0
- Dropped

Traffic bet domains must be negotiated/upstreams
- Must shape DB packets as per negotiated rate
- Overflow incident could be indicative of service attack

EF: Corresponds to DSCP 46: Premium service connection:
- Appears as virtual lease line bet end points
- If source sends packet w/value 101110 in DSCP field
- Rtrs that support DS functionality must expedite packet fwding
- Not change DSCP field value to lower priority

RTA: Real-Time Apps: Any app that must function w/in time frame
- Immediate/continual basis: Little/no delay

WCET: Worse-Case Execution Time:
- Depends on max amt of time app task requires to execute

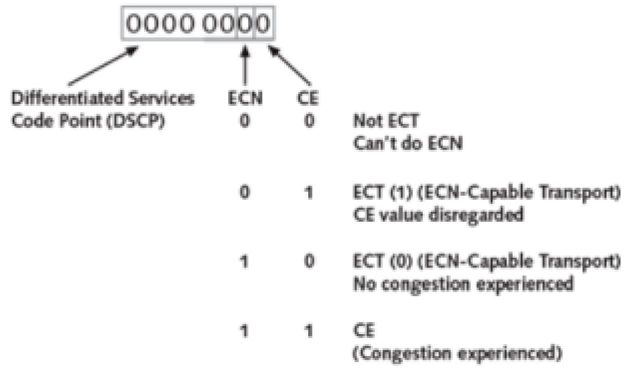
VoIP: Uses IP: RTA: Benefits from DSCP EF handling
- Intolerant of delay/time-sensitive

Other RTA types:
- Chat/Comm storage solutions/IM/Gaming/Media/Videoconf
- E-Mail/Web traffic: Can still function w/delays

ECN

Explicit Congestion Notification:
- Designed to provide devices w/method for notifying each other
- Link is experiencing lag before rtrs drop packets
- Both sides of congested link must support ECN
- Tries to reduce # of dropped packets

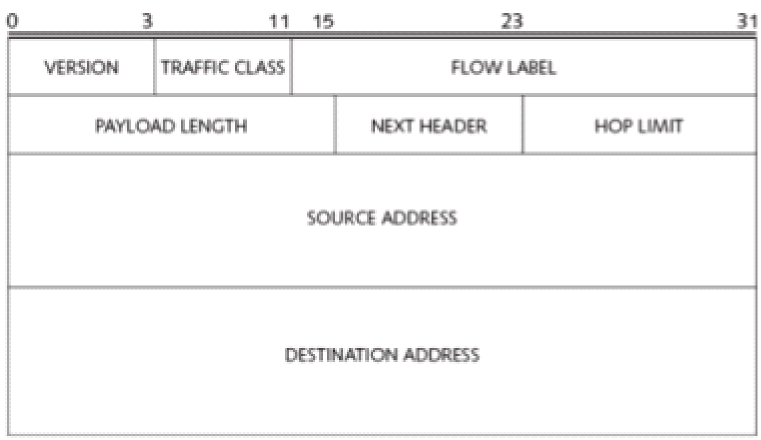
How: When packet sent bet 2 ECN-capable rtrs:
- Marked **ECT(0)/ECT(1) for ECT: ECN-Capable Transport**
- If packet crosses queue bet 2 rtrs: Receiving rtr may change

	○ **CE: Congestion Encountered** ○ Instead of dropping packet **ECN/CE requires 2 bits w/in IP header** • Combo of 2 bits provides 7 possible interpretations  **Must look at values combined in both bits to interpret ECN value** • **ECN/CE bits: 01 or 10:** Sending rtr supports it: ECT • **Value set to 00:** Sender not ECT • **Value set to 11:** Sender ECT and congestion on link
Total Length Field	Defines length of IP header/any valid data (doesn't include link padding) • Total length 60 bytes • IP header: 1st 20 bytes • Remaining packet length: 40 bytes
ID Field	Each packet given unique ID value when sent • Fragmented? Supports smaller size: Same ID placed in each • Helps ID fragments part of same data set Detects/removes duplicate datagrams from congested rtrs • Used in some diagnostic tools **MDL: Max Datagram Lifetime:** Longest period of time a datagram may exist
Flags Field	**3 bits long:** Bit value assignments • **Bit 0:** Reserved Set to 0 • **Bit 1: DF: Don't Fragment** bit: 0: May Fragment • **Bit 2: MF: More Fragments:** 0: Last fragment: 1: More to come **If fragmentation must be done to cross network w/smaller MTU used** • IPv4 rtrs continually fragment traffic from 1 hop/next • Depending on MTU size tolerated by link • IPv6: Source nodes use PMTU Discovery to determine smallest link MTU
Fragment Offset Field	**If packet fragment:** Shows where to place packet's data when reassembled • Gives offset in 8-bye values • Example: 1st fragment may have offset of 0/contain 1,400 bytes data • 2nd fragment would have offset value of 175 (175*8-1,400) Field only used if packet a fragment
TTL Field	**Time to Live:** Remaining distance packet can travel • Defined in seconds • Implemented as # of hops a packet can travel before being discarded by rtr • Typical: 32/64/128 • Max TTL: 255
Protocol Field	**Indicates what's coming next:** • **1:** ICMP | **2:** IGMP | **6:** TCP | **8:** EGP | **9:** IGP | **17:**UDP | **45:** IDRP • **253/254:** Experimentation • **143-252:** Unassigned • **255:** Reserved by IANA
Header Checksum Field	**Provides error detection on contents of IP header only:** • Doesn't cover other contents of packet

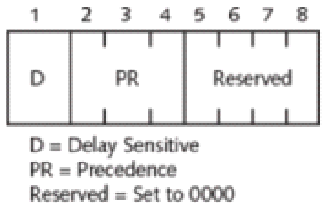
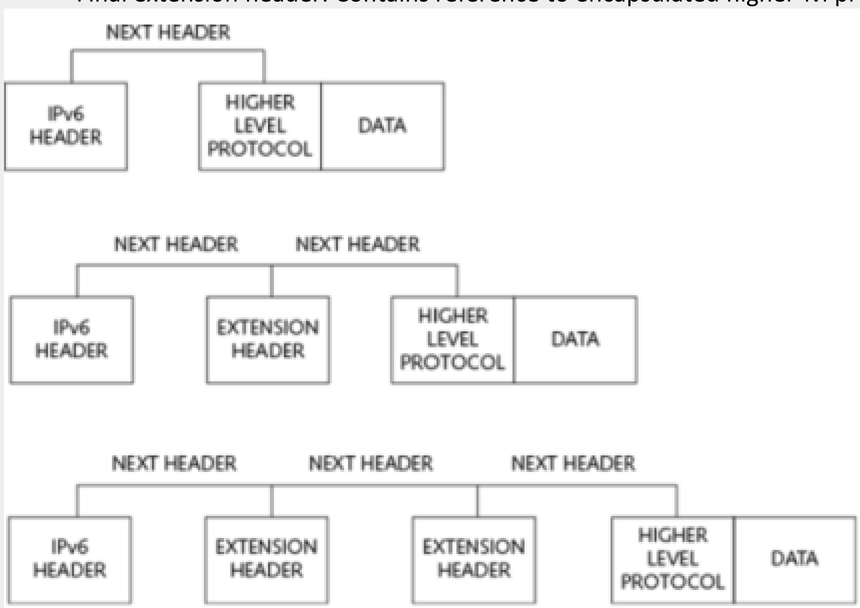
	• Doesn't include Header Checksum field in calculation Error-detection mechanism in addition to data link CRC required to pass through rtrs Example: • Ethernet packet arrives at rtr: Performs CRC: Cyclic Redundancy Check • Ensures packet wasn't corrupted • After considered good: Rtr strips data link header • Leaves behind unencapsulated network layer packet Required to detect/defeat rtr packet corruption
Source Address Field	IP of host that sent packet: • Some cases like DHCP: Host may not know IP so uses 0.0.0.0 • Can't contain multicast/broadcast address • Source must originate from specific network int w/IP assigned to it
Destination Address Field	Can include unicast/multicast/broadcast address: Final destination of packet
Options Fields	**Can be extended by 7 options:** Not often used: Must end on 4-byte boundary • IHL field defines header length in 4-byte boundaries • 0: End of Options List • 1: No Operations • 2: Security • 3: Loose Source Route • 4: Time Stamp • 5: Extended Security • 6: Record Route • 8: Stream ID • 9: Strict Source Route Useful when testing/debugging code/specific connections • Can contain 0/1/more options
Padding	Used to make sure header ends at 32-bit boundary • Consists of w/e number of 0-filled bytes required to make IPv4 header end at • Makes sure header length is always multiple of 32-bits

IPv6 Header Fields/Functions

- Adds improvements over IPv4: Enhanced support for extensions/options
- More efficient packet forwarding/labeling for specific traffic flows
- **RFC 1883**
- **40 octets: 320 bits of packet**



Version Field	4-bit IP version #: Will always be 6: 0110
Traffic Class Field	8-bit field used by source network hosts/forwarding routers • Distinguishes classes/priorities in IPv6 packets • Network node's IPv6 int must provide method for upper-layer protocol to offer value ○ for bits in any packet from the protocol w/default being 0 Allowed to change bit values if source/destination node/fwding packet • Otherwise nodes should ignore traffic class field bits they don't support

	 **1st bit:** Whether traffic is delay sensitive **Set to 1:** Sensitive **Precedence:** Used by rtr to prioritize traffic through queue: Buffering system to hold packets **Last 4 bits:** Reserved
Flow Label Field	**Flow:** Set of packets for which source requires special handling by intervening rtrs • **20-bit:** Used by source node to request special handling [RTA's/nondefault QoS) • **Value of Flow Label:** 0 for packets that haven't been labeled • Packet classifiers use FL/source/dest. address to ID • Once set to nonzero value: Expected to remain unchanged bet source/dest. • Forwarding node must either leave nonzero flow value unchanged ○ Change it only for operational sec reasons
Payload Length	**16-bit length:** Describes size of payload in octets: Including extension header (optional info) • Located bet fixed header/upper-layer protocol header • Length: 0: Hop-by-Hop Options extension header possesses jumbogram options
Next Header Field	**8-bit:** Specifies header type of header following IPv6: Extension ones • Field points to 1st extension header • Contains w/in own Next Header field identifier for following extension header • Final extension header: Contains reference to encapsulated higher-lvl protocol 

Next Header Field Values

Decimal	Hexadecimal	Extension header/protocol name
0	00	Hop-by-Hop Options extension header
1	01	ICMPv4
2	02	IGMPv4
4	04	IPv4 Encapsulation
6	06	TCP
8	08	EGP
17	11	UDP
41	29	IPv6
43	2B	Routing extension header

44	2C	IPv6 fragmentation header

IGMP	**Internet Group Management Protocol:** Network protocol used by hosts/adjacent rtrs **IGMPv3:** Used in IPv4 networks to report IP multicast group memberships from 1/other • Ability to use source filtering • Allows network nodes to report having interest in receiving only particular src addr • Sent to a specific multicast addr • Prevents multicasts from being delivered to networks not interested in info
Hop Limit Field	**8-bit:** Decrements by 1 each time fwded by network node: Discarded if reaches 0 • Max value of 255/max hop value
Source Address Field	128-bit address of source packet
Destination Address Field	128-bit address of recipient of packet: May not be final recipient if extension header avail

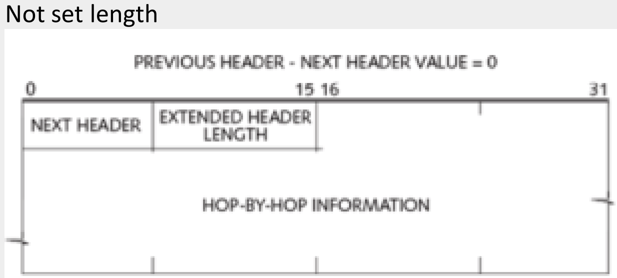
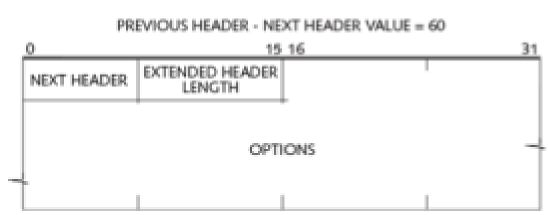
IPv6 Extension Headers: Additional functionality implemented: Used only for specific purposes
Order for extension headers:

1. Hop-by-Hop Options
2. Destination Options (normal op)
3. Routing
4. Fragment
5. Authentication

- **ESP: Encapsulating Security Payload**

1. Destination Option (optional duplicate)

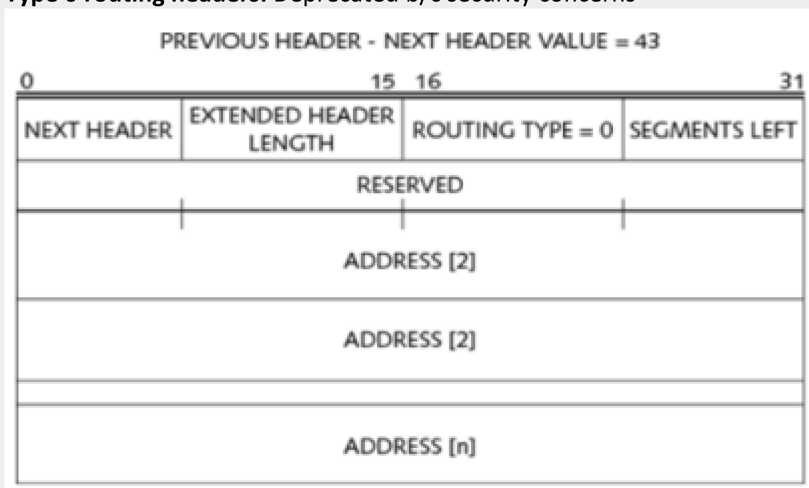
Can contain a Transport layer header: UDP/TCP/ICMP: Chained:

Hop-by-Hop Extension Header	Designed to carry info that affects rtrs along path: **2 fields:** **1. Next Header field: NH value** **2. Extended Header Length field:** Length of header excluding 8-byte min required Not set length 
Destination Options Extension Header	If appears after ESP extension header: Sent through encryption process • Makes room for future standards • May appear in more than 1 loc (only header to do this) • Immediately before routing extension header • Last header before actual higher-layer protocol data 
Routing Extension Header	**Supports strict/loose source routing for IPv6:** • Includes fields for intermediary addresses which packet should be fwded **1st byte:** Indicates NH that follows rtr extension header

Extended Header Length field: Defines length excluding min 8-bits required
Routing Type = 0: Sender calcs path among all rtrs it wishes packet to visit

- Places their addresses in ordered list in Hop-by-Hop Options EH
- Final destination rtr at end of list
- Then places address of 1st rtr in Destination addr field
- When packet arrives at 1st destination: Rtr examine/finds header
- If correct: Places address of next rtr in list
- Places its own address at bottom of list
- Continues until final destination
- 255 rtrs may be included in such a list
- **Segments Left field:** Defines # of remaining route segments packet must visit

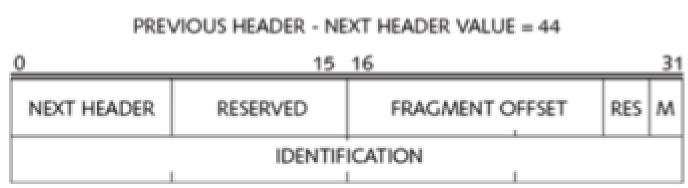
Type 0 routing headers: Deprecated b/c security concerns



Fragment Extension Header

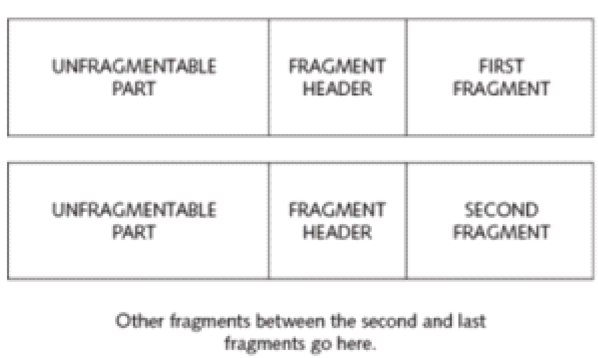
IPv6: Doesn't support fragmentation at fwding rtrs: Source of a packet can still fragment

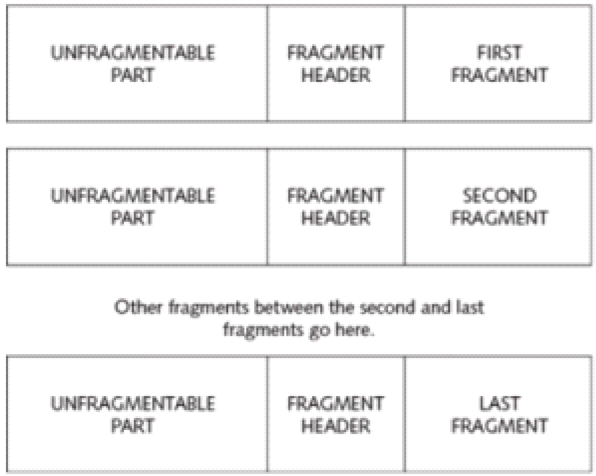
- All packets treated as implicit Do Not Fragment bit set
- PMTU Discovery process used to provide source stations w/max fragment size



Source node may fragment packet in order to meet reqs of smallest link MTU in path

- Before fragmented by source node: AKA Original packet
- Made of 2 segments
    1. **Unfragmentable:** Header/any EH may exist to/routing if present HBH
    2. **Fragmentable:** Rest packet: EH: Processed by dest node/upper-layer data



<table>
<tr>
<td></td>
<td>


Payload length: Unfragmentable:

Changed from size of original packet to length of fragment packet – IPv6 header
Value of NH field of last header changed to 44

Fragment Header:

Contains next header value
Identifies 1st header of fragmentable part of original header
Contains a Fragment Offset: Expressed in 8-bit units

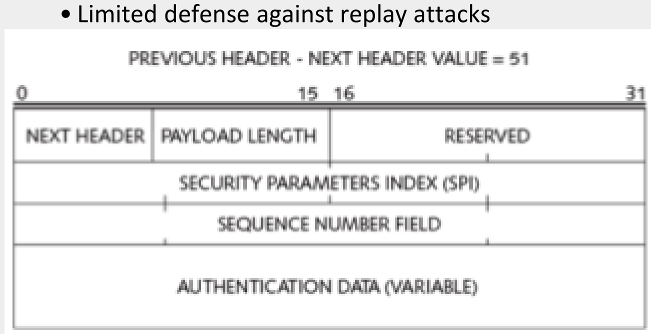
1st fragment: Set to value of 0
M flag: Also set to value of 0 for last fragment
Otherwise: M set to 1
Last element: Identification value: Generated for original packet

Fragment length: Established to accommodate smallest link MTU size in Path MTU

Ensures it will arrive at destination node

</td>
</tr>
<tr>
<td>Authentication Extension Header</td>
<td>
Designed to specify true origin of packet by preventing spoofing/connection theft

Integrity check on parts of packet that don’t change in transit
Limited defense against replay attacks



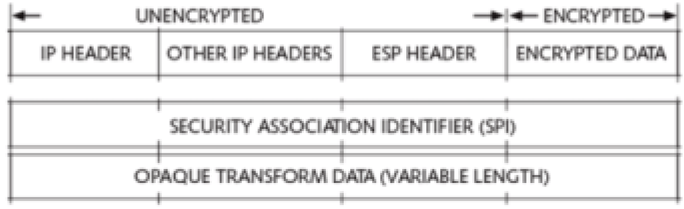
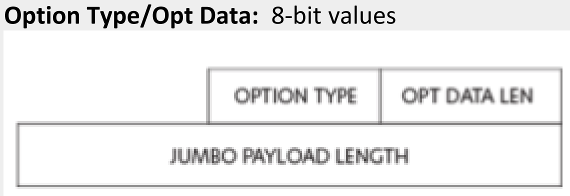
1-byte Next Header field: Indicates NH in chain
1-byte payload length: 4-byte words following **SPI: Security Parameters Index** field

Bits in Reserved should be set to all 0’s
SPI contains values: May point to index/table of sec params
or SA: Sec Association
Always a pointer to sec details on its partner

Sequence Number field: Used to ensure receivers recognize old packets
Auth Data field: Based on cryptographic checksum on payload data
</td>
</tr>
<tr>
<td>Encapsulating Sec Payload Extension Header/Trailer</td>
<td>
Auth process defined by Auth extension header:

Doesn’t encrypt/protect data from sniffing attacks
Still in native transmission fmt
ESP EH should be used to encrypt data
Must always be last header of IP header chain

</td>
</tr>
</table>

	 **AH/ESP/IPSec** **IPSec: IP Security:** Suite of add-in sec protocols for networks: • Access control • Connectionless integrity • Data origin auth • Protection against replay attacks (etc) **AH/ESP: Part of IPSec suite:** • AH specifies true origin of packet: Preventing spoofing/connection theft • Integrity checking/limited defense **ESP: Encryption services under IPSec:** • AH protects payload/all header fields of IP datagram w/exception of unauth • IPv6: A protects extension header itself/destination options extension header ○ IP Payload/fixed IPv6 header/extension headers before AH ○ Exceptions: DSCP/ECN/Flow Label/Hop Limit
Jumbograms	**Very large packet:** Use HBH options EH to add alternate Packet Length field of 32 bytes • Allows packet to carry single chunk of data larger than 64 KB [over 4billion bytes] • Use of large MTUs: Fewer packets that require processing by backbones **Option Type/Opt Data:** 8-bit values  **Jumbo Payload Length Field:** 32-bit value: Length of IPv6 packet in octets – header • Must be greater than 65,535 **Will process Jumbo Payload as jumbogram if:** 1. Packet header's payload length field set to 0 2. NH field set to 0: Hop-by-Hop Options extension comes next 3. Link-layer framing indicates addl octets exist beyond IPv6 header

Post 5

Thursday, January 24, 2019 11:45 PM

CH 5 NOTES: WCNA ICMPV4 P1

Reachability	Ability to find at least 1 transmission path bet pair of hosts: So can exchange datagrams across network • ICMP provides possible way to return info about routes

ICMP: Ability to report errors/congestion/other conditions: Nothing but specially formatted IP diagrams: 8 byte header

- Subject to same conditions as other packets in general traffic
- Up to IP host that receives incoming ICMP msgs to act on content of those msgs

Network congestion	Occurs when traffic starts to exceed handling capabilities

ICMP Msg Types/Uses

Echo/Echo Reply	Functionality for reachability utilities: Ping/Tracert • Essential install/config/troubleshoot
Destination Unreachable	Docs when routing/delivery errors prevent datagrams from reaching dest • Code values extremely impt: Also PMTU Discovery bet pairs of hosts
Source Quench	Permits GW to instruct sending host to adjust (lower) sending rate: Ease congestion problems
Redirect	Permits GW on nonoptimal route bet sender/receiver to redirect traffic: More optimal path
Router Advertisement	Permits hosts to req. info about local rtrs: Permits rtrs to advertise their existence on a network
Time Exceeded	Indicates a datagram's TTL/fragmented reassembly timer: Expired: Too-short TTL • Presence of routing loop on a network
Parameter Problem	Some error occurred while processing header of inc datagram: Causes it to be discarded • Catchall for ambiguous/miscellaneous errors • Further investigation needed

ICMPv4: Core protocol in IP suite: 1981 RFC 777: Obsoleted by RFC 792:
Primary use: Send certain error msgs to other networked nodes

- **ping** used widely to test connection bet 1 machine/other
- Differs from TCP/UDP: No payload carried: Not used by other apps: Supports series of testing/error msgs

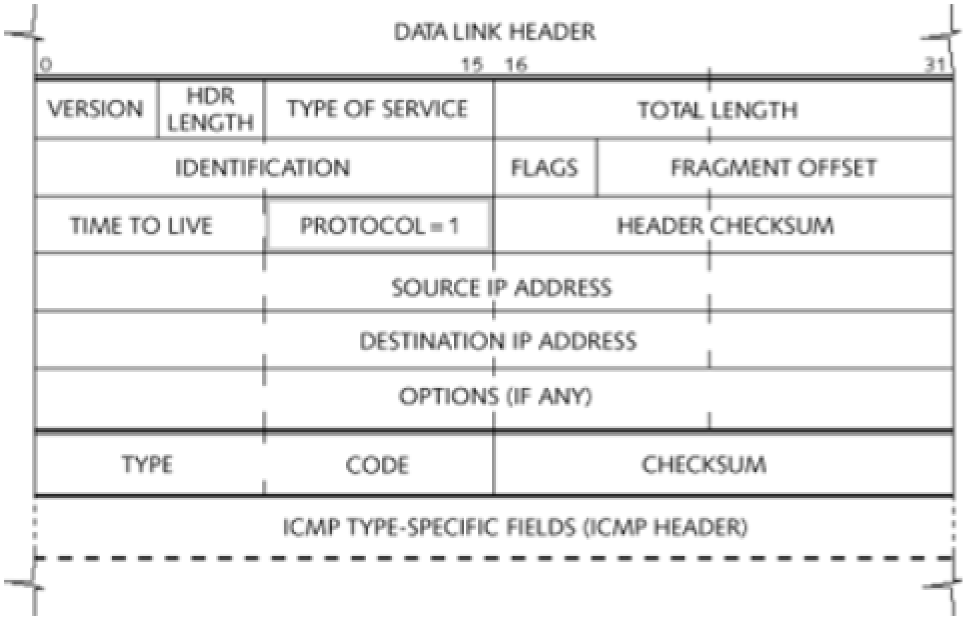
Supports ping/tracert/traceroute:

- Traces packet from source to destination counting # of hope made in transit/time req for each hop

RFC 792: Defines basics for all valid ICMP msgs/defines kind of info/services ICMP can deliver

- Provides mechanism for GW's (routers)/destination hosts to comm w/source hosts
- Msgs take form of specially formatted datagrams: Specific types/codes
- Required element in some implementations of TCP/IP: Essential part of support
- Report errors only about processing non-ICMP IP datagrams
    - Conveys no msgs about itself/provides info
    - Only about 1st fragment in any seq. of fragmented datagrams

Header:



Value 1 in IP header: Denotes ICMP header follows IP header
Consists of 2 portions:

1. Constant portion
2. Variable portion

Constant ICMP Fields	**Packets only contain 3 required fields after IP header:** 1. Type 2. Code 3. Checksum **Some ICMP packets:** Addl fields provide info/details about msg
Type Field	ID's types of ICMP msgs sent on network: Type #s correspond to types of msgs

ICMPv4 Types/Names/Reference

Type Number	Name	Reference
0	Echo Reply	RFC 792
1	Unassigned	
2	Unassigned	
3	Destination Unreachable	RFC 792
5	Redirect	RFC 792
6	Alternate Host Address	JBP
8	Echo	RFC 792
9	Router Advertisement	RFC 1256
10	Router Solicitation	RFC 1256
11	Time Exceeded	RFC 792
12	Parameter Problem	RFC 792
13	Timestamp	RFC 792
14	Timestamp Reply	RFC 792
19	Reserved (Security)	Solo
20-29	Reserved (Robustness Experiment)	Zsu
40	Photuris	RFC2521
41	Msgs utilized by experimental mobility protocols (Seamoby)	RFC 4065

42-252	Unassigned	JBP
253	RFC3692-style Experiment 1	RFC 4727
254	RFC3692-style Experiment 2	RFC 4727

JBP: Jon B. Postel: A dev of the IP suite:
Code Field: Many ICMP packet types have one
Type 3: Destination Unreachable Codes

Code	Definition
0	Net Unreachable
1	Host Unreachable
2	Protocol Unreachable
3	Unreachable
4	Fragmentation Needed and Don’t Fragment was Set
5	Source Route Failed
6	Destination Network Unknown
7	Destination Host Unknown
8	Source Host Isolated
9	Comm w/Destination Network Is Administratively Prohibited
10	Comm w/Destination Host Is Administratively Prohibited
11	Destination Network Unreachable for Type of Service
12	Destination Host Unreachable for Type of Service
13	Comm Administratively Prohibited
14	Host Precedence Violation
15	Precedence Cutoff in Effect

Type 5: Redirect Codes

Code	Definition
0	Redirect Datagram for Network/Subnet
1	Redirect Datagram for Host
2	Redirect Datagram for the Type of Service/Network
3	Redirect Datagram for the Type of Service/Host

Type 6: Alternate Host Address Code

Code	Definition
0	Alternate Address for Host

Type 11: Time Exceeded Codes

Code	Definition
0	Time to live Exceeded in Transit
1	Fragment Reassembly Time Exceeded

Type 12: Parameter Problem Codes

Code	Definition
0	Pointer Indicates the Error
1	Missing at Required Option
2	Bad Length

Type 40: Photuris Codes: Session-key management protocol: RFC 2522

Code	Definition

0	Bad SPI
1	Authentication Failed
2	Decompression Failed
3	Decryption Failed
4	Need Authentication
5	Need Authorization

Checksum Field: Provides error detection for ICMP header only: Fields that follow Checksum vary: Depends on msg sent

Destination Unreachable	Returned to source node when packet sent not delivered to dest addr Why a packet might fail to be delivered: • Erroneous parameters (invalid IP addresses) • Router unable to reach network where destination is located • IPv4 doesn't guarantee that sent packets always reach dest
Source Quench	Tells source node: Reduce rate of speed sending packets to dest node • Network nodes: Tend to buffer packets in a 'last moment' scenario – When traffic coming too fast to be processed/size of buffer limited – When the buffer is full/receiving node can't process traffic fast enough • Source quench sends a message to the sending node • Reduce buffer volume • Responds by slowing down transmission rates until messages stop Limited: They only contain info that the destination is congested • Don't tell source node what it should do • No message sent to source when buffer clears **Receiving window: AKA: Sliding window acknowledgement system:** • More effective flow-control mechanism • Regulates transmission of packets between 2 network devices
Time Exceeded	**Sent in 2 circumstances:** • **When a packet's TTL (Time-To-Live) field is decremented to 0** ▪ Packet is caught in a routing loop: Convergence hasn't occurred ▪ Can't tell which route to take to the destination ▪ Packet goes through routers repeatedly until TTL drops to 0 • **When some fragments of a message don't reach destination node** ○ First fragment arrives at destination ○ Timer allots time the node will wait for remaining fragments ○ This is so fragments can reassemble the message ○ If the timer hits 0: All fragments discarded: This msg hits
Redirect	1st-hop router sends message to source when it receives a packet that: • Couldn't be managed more efficiently by that 1st-hop router • Not an error message: ICMPv4 classified this way • Provides a limited amount of routing information to hosts
Parameter Problem	**Generic error message:** Contains special pointer field used to tell source problem • Bad parameters in header fields means dropped packet
Echo Request/Reply	Tests connectivity
Timestamp Request/Reply	Synchronize system clocks for date/time: Think NTP

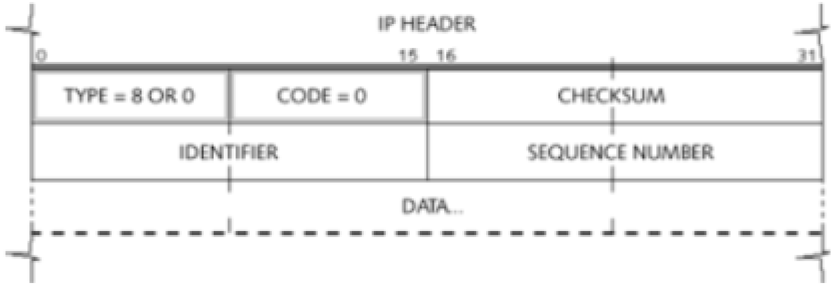
Traceroute: Msg type is similar to echo request/reply

- Instead of testing for basic connectivity: Traces exact sequences of rtrs used to send packet
- Hop-by-hop basis

Windows	Tracert
Linux	Traceroute

- Sent as a single packet containing special Traceroute IP option recognized by rtrs receiving packet
- Rtrs fwd the packet along the route to destination: Each rtr fwding msg responds to source
- Time in M/s

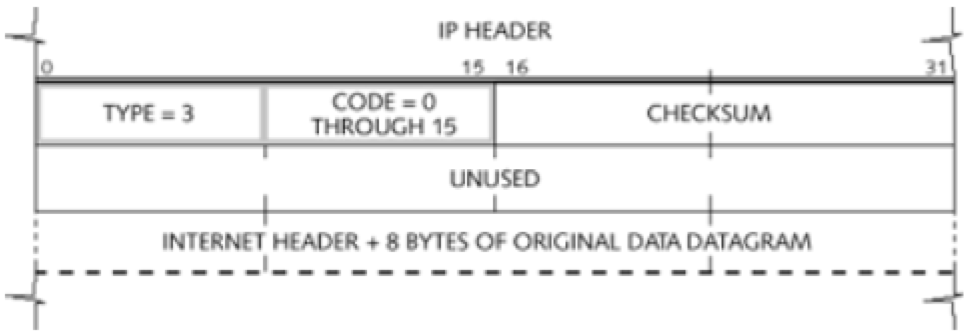
Type 0 and 8: Echo Reply/Request Packets:



Windows Server 2012/7/10 ping packets display the following chars:

- LE: Identifier field set to 256 dec (0x100)
- First echo sent: Sequence # (LE) field value set to a multiple of 256 dec 0x100
    - In each subsequent echo, field is incremented by 256 dec
- Data field contains value abcdefghijklmnopqrstuvwabcdefghi
- Identifier/Sequence # fields have both BE (big-endian) and LE (little-endian) entries
    - Which bytes are most significant (BE) and least significant (LE) in multibyte data types

Type 3: Destination Unreachable Packets:



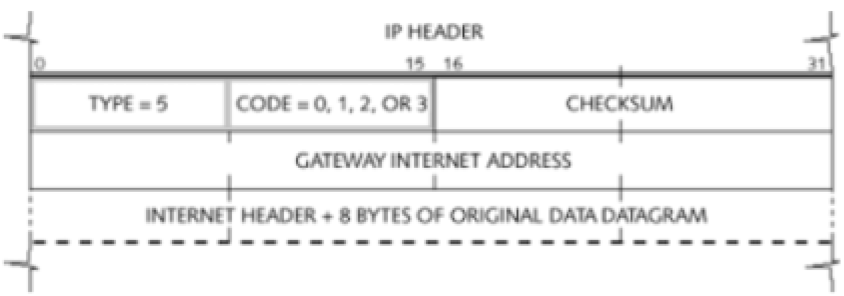
- Must return IP header/8 bytes of the original datagram that triggered this response
- Destination Unreachable reply: Contains IP header/1st 8 bytes of data in UDP header
    - From original DNS query

Total of 16 (0-15) codes currently assigned to ICMP Destination Unreachable type number: Not all used

Code 0	**Net Unreachable:** Sent by rtrs to indicate that the rtr knows about the network • # used in incoming packet says route isn't up at this time • Or too far away to reach
Code 1	**Host Unreachable:** Rtr sends reply to report it couldn't locate the destination host • Currently: Also sent when destination network unknown • Can occur when a host is offline/part of network down • Can occur w/switch/rtr failure/when host IP doesn't exist
Code 2	**Protocol Unreachable:** Host/rtr sends this msg to indicate that protocol can't be processed
Code 3	**Port Unreachable:** Host/rtr sends this to indicate sender doesn't support process/app
Code 4	**Fragmentation Needed/Don't Fragment Set:** 2 versions of this reply **1. Standard:** Simply states packet had the DF: Don't Fragment bit set **2. PMTU:** Includes information about restricting link **PMTU Discovery:** Place MTU (Max Trans. Unit) of restricting link into 4byte area
Code 5	**Source Route Failed:** • Rtr sends reply to indicate it can't use the strict/loose source routing path specified • If original packet defined strict source routing: ○ Rtr doesn't have access to next rtr indicated in strict path list • If original packet defined loose source routing: ○ Perhaps rtr knows no next-hop rtr that can fwd packet
Code 6	**Destination Network Unknown:** Obsolete: Rtrs send code 1 unreachable

Code 7	**Destination Host Unknown:** Indicates it can’t reach a directly connected link
Code 8	**Source Host Isolated:** Obsolete Rtrs sent this to indicate a host was isolated/packets unrouted
Code 9	**Communication w/Destination Network Is Administratively Prohibited:** • Indicates rtr config to block access to desired destination
Code 10	**Communication w/Destination Host Is Administratively Prohibited:** • Indicates desired host can’t be reached bc of block configs
Code 11	**Destination Network Unreachable for Type of Service** • Indicates the Type of Service (TOS) requested in an incoming IP header • Default TOS: 0: Not avail through this rtr/network • Only rtrs that support TOS can send this type
Code 12	**Destination Host Unreachable for Type of Service:** • Indicates TOS requested in incoming header not avail through this rtr for that host • Only routers that support TOS can send this type of ICMP message.
Code 13	**Communication Administratively Prohibited:** • Indicates rtr can’t fwd packet BC packet filtering prohibits it
Code 14	**Host Precedence Violation:** Indicates Precedence value defined in sender’s original header not allowed • Also results in a discarded packet
Code 15	**Precedence Cutoff in Effect:** Indicates an admin imposed a min lvl of precedence to obtain service from a rtr • A lower-precedence packet was received: Discarded

Type 5: Redirect

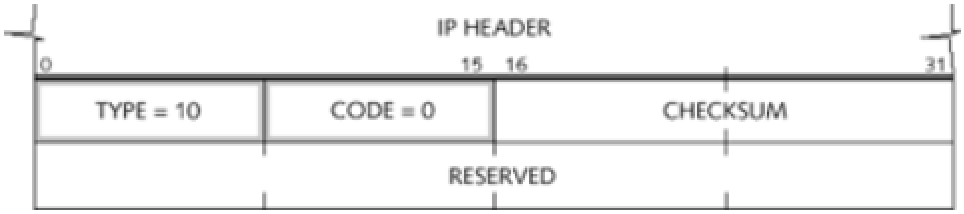


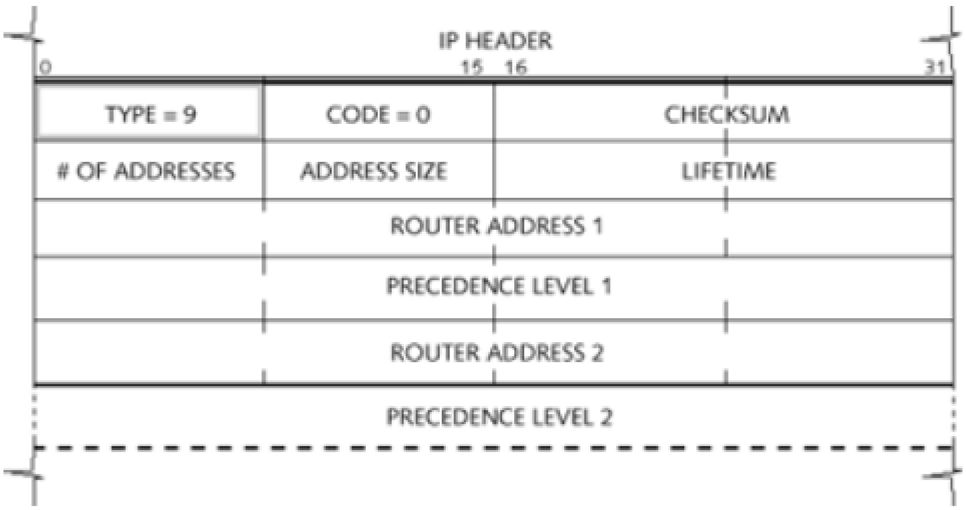
4 codes that define whether Redirect packet contains new route to full host address/network address

- Some redirection info points to path used for specific TOS

Code 0	**Redirect Datagram for Network/Subnet:** Rtr can send to indicate a better way to get to desired network • BC rtrs can’t determine which portion of a destination address is network/host: They use Code 1
Code 1	**Redirect Datagram for Host:** Rtr can send to indicate better way to get to desired host: Most common
Code 2	**Redirect Datagram for TOS/Network:** Rtr can send to indicate better way to get to desired network using TOS • BC rtrs can’t determine network/host portion of dest addr: Use Code 3
Code 3	Redirect Datagram for TOS/Host: Rtr can send to indicate better way to get to destination host using TOS

Types 9 and 10: Router Advertisement and Router Solicitation

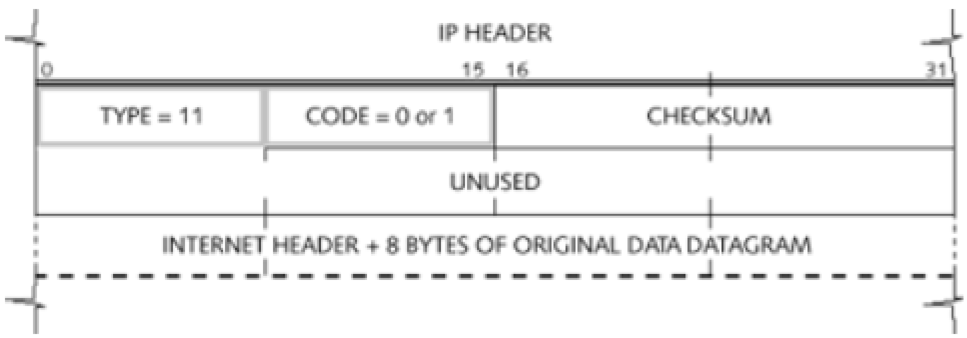




Router Advertisement packets include following fields after checksum field:

# of Addresses	# of rtr addresses advertised in packet
Address Size	# of 4-byte increments used to define each rtr address advertised • BC version includes 4-byte Precedence field • IP Address field: Address Size value 2 (2+2+4 bytes)
Lifetime	Max # of seconds rtr info may be considered valid
Rtr Addr 1	Sending rtr local IP address
Precedence Lvl 1	Preference value of each rtr address advertised • Higher values indicate greater preferences • May be config at rtr to ensure 1 more likely to become default GW
Rtr Address 2/Precedence Lvl2	If there are addl rtr values: Will follow w/precedence lvls

Type 11: Time Exceeded

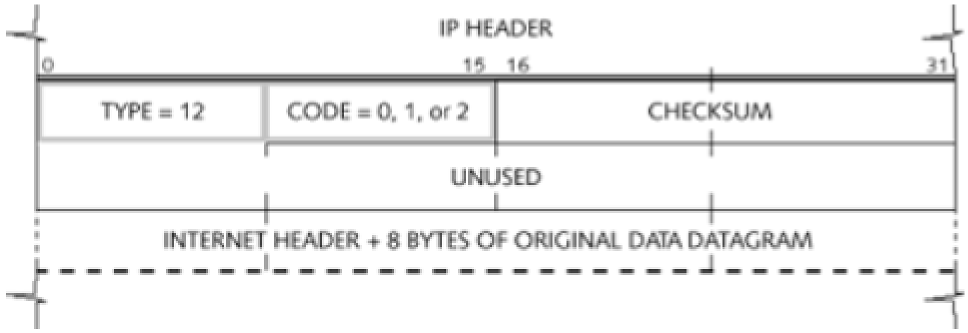


2 codes can be used in Time Exceeded packets

Code 0	**TTL Exceeded in Transit:** Rtr sends msg to indicate packet arrived w/TTL value of 1 • Rtrs can't decrement TTL value to 0/fwd: Must discard packet/send msg
Code 1	**Fragment Reassembly Time Exceeded:** Host sends when it doesn't receive all fragment parts before expiration • When 1st packet of fragment set arrives: TTL interpreted as "seconds of lifetime remaining"

	• Timer begins counting down in seconds • If all fragments of set don't arrive before timer expires: Entire set considered invalid • Receiver sends msg back to originator of set, causing it to be resent

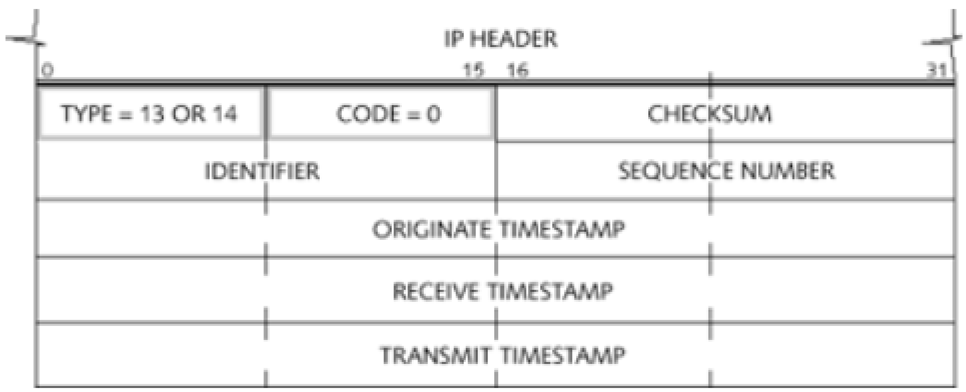
Type 12: Parameter Problem



3 codes can be used in Param Problem msgs

Code 0	Pointer Indicates Error: Includes a Pointer field that indicates where in returned header/datagram error occurred
Code 1	Missing Required Option: Indicates sender expected addl info in Option field of original packet
Code 2	Bad Length: Indicates original packet structure had invalid length

Types 13 and 14: Timestamp and Timestamp Reply: Defined as method to obtain time: Value returned in ms since midnight UT



Timestamp requester enters current send time in Originate Timestamp field

- Receiver enters time value in Receive Timestamp field as packet processed
- Receiver places current Timestamp in Transmit Timestamp field
- NTP more robust

Post 6

Thursday, January 24, 2019 11:46 PM

WCNA ICMP NOTES P2: ICMPV6

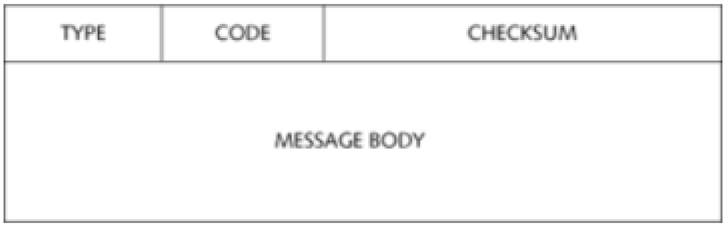
ICMPv6 Message Types

Type Name	Reference	Doc
0	Reserved	RFC 4443
1	Destination Unreachable	RFC 4443
2	Packet Too Big	RFC 4443
3	Time Exceeded	RFC 4443
4	Parameter Problem	RFC 4443
100	Private experimentation	RFC 4443
101	Private experimentation	RFC 4443
102–126	Unassigned	
127	Reserved	RFC 4443
128	Echo Request	RFC 4443
129	Echo Reply	RFC 4443
130	Multicast Listener Query	RFC 2710
131	Multicast Listener Report	RFC 2710
132	Multicast Listener Done	RFC 2710
133	Router Solicitation	RFC 4861
134	Router Advertisement	RFC 4861
135	Neighbor Solicitation	RFC 4861
136	Neighbor Advertisement	RFC 4861
137	Redirect Message	RFC 4861
138	Router Renumbering	RFC 2894
139	ICMP Node Information Query	RFC 4620
140	ICMP Node Information Response	RFC 4620
141	Inverse Neighbor Discovery Solicitation	RFC 3122
142	Inverse Neighbor Discovery Advertisement	RFC 3122
143	Version 2 Multicast Listener Report	RFC 3810
144	Home Agent Address Discovery Request	RFC 6275
145	Home Agent Address Discovery Reply	RFC 6275
146	Mobile Prefix Solicitation	RFC 6275
147	Mobile Prefix Advertisement	RFC 6275
148	Certification Path Solicitation	RFC 3971
149	Certification Path Advertisement	RFC 3971
150	Experimental mobility protocols	RFC 4065
151	Multicast Router Advertisement	RFC 4286
152	Multicast Router Solicitation	RFC 4286
153	Multicast Router Termination	RFC 4286

154	FMIPv6 Messages	RFC 5568
155	RPL Control Message	RFC-ietf-roll-rpl-19.txt
156–199	Unassigned	
200	Private experimentation	RFC 4443
201	Private experimentation	RFC 4443
255	Reserved for expansion of ICMPv6 info	RFC 4443

Specific message types may have unique fmting: IPv6 header/1/more extension headers come before msg

- Next Header value: 58



Type field: Type of msg: Value determines fmt of remaining data
Code field: Value of msg type
Example: Echo request/reply/Neighbor Ad: Code value 0

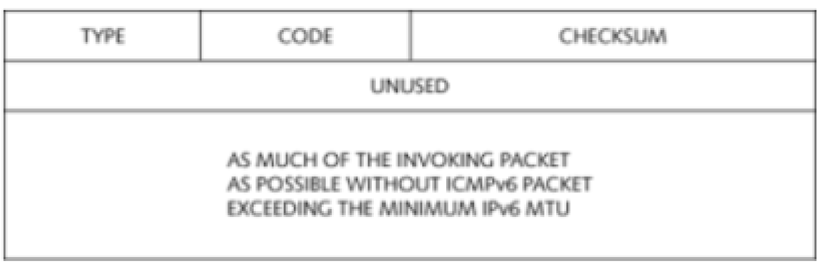
- Checksum detects data errors
- Contents of Message Body: Depend on message type

ICMPv6 msgs: 2 types:

1. Error msgs
2. Info msgs

Error Messages	Destination Unreachable | Packet Too Big | Time Exceeded | Parameter Problem • Other errors reserved/unassigned/set aside for experimentation

Destination Unreachable Messages



Field Name	Description
Type	1
0	No route to destination
1	Comm w/destination admin prohibited
2	Beyond scope of source address
3	Address unreachable
4	Port unreachable
5	Source address failed ingress/egress policy
6	Reject route to destination
Unused	No code values use this field: • Must be set to 0 by source node • Ignored by destination node

More about Msgs

Destination Unreachable	Produced by rtrs/layer in source node

messages	• Response to encountering packet that can't be delivered/congestion • Value in Code field: ○ Informs sending node reason ○ Destination node typically sends a Code 4
Packet Too Big Messages	**Required BC how IPv6 manages fragmentation/reassembly** • Doesn't fragment packets to accommodate link MTU to next hop • Source nodes req PMTU Discovery to find smallest MTU for all links in PMTU • If rtr receives a packet too large for link to next hop: Drops/sends this msg • Source node will mod MTU to fit smaller link MTU/resend msg

Format

Field Name	Description
Type	2
Code 0	Set to 0 by source node/ignored by destination node
MTU	MTU value of next-hop link

Packet Too Big msgs: Must be sent by rtrs that can't fwd a packet BC it exceeds MTU size limitation of outgoing link

- Unlike all other error types: Sent in response to msgs w/IPv6 multicast or LL multicast/broadcast address

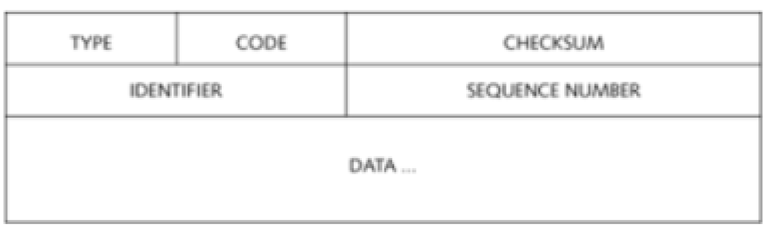
Time Exceeded	Similar to ICMPv4: Rtr that receives packet/decrements hop limit field to 0 will drop
Parameter Problem	Like ICMPv4: Generic: Triggered by any serious error in header: Special pointer used to indicate it

Parameter Problem

Field Name	Description
Type	4
0	Erroneous header field encountered
1	Unrecognized Next Header type encountered
2	Unrecognized IPv6 option encountered
Pointer	Extends beyond end of ICMPv6 packet if • Field w/beyond what can fit w/in max size of error msg

Info Msgs: Type codes in 128–255 range

Echo Request/Reply



All fields: Multiple of 4 bytes: 1st: Fields type, code, checksum: 2nd: Identifier, seq #: Last bytes: Data

Field Name	Description
128	Echo Request
129	Echo Reply
Code 0	Both types
Identifier	For Echo Request: • Identifier helps matching replies • May be 0 • For replies: Identifier from invoking request
Sequence Number	Requests: # to aid matching replies

	• May be 0

Data for Request: May consist of 0/more octets of data
Data for Reply: Data from invoking requests

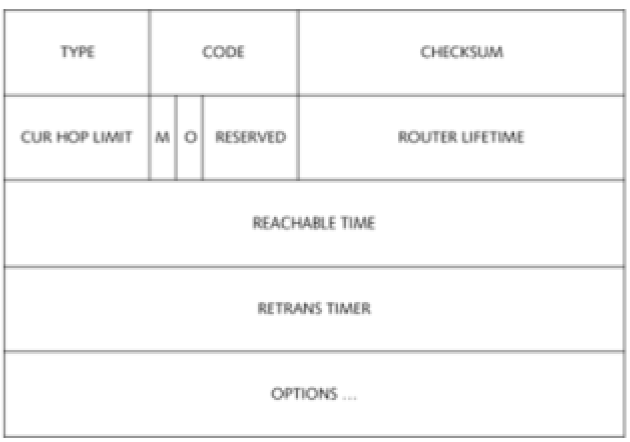
- Source address of a reply msg sent in response to a unicast echo must be the same as the dest. addr of request
- Reply msgs are also sent in response to request sent to multicast/anycast
  - Source addr of reply must contain unicast address of node responding

Router Advertisement/Solicitation Messages: Neighbor Discovery (ND) protocol IPv6
Message format

Fields in multiples of 4 bytes: 1st: Type, code, checksum: 2nd: Reserved: Last: Field options

Field Type	Description
Type	133
Code	0
Checksum	Checksum
Reserved	Unused field: Set to 0 by source/ignored by destination node
Options	RFC 4861 specifies source LL address as only valid option • Address not included if unspecified

Message format



Router Advertisement Message Format Fields

Field Name	Description
Type	134
Code	0
Checksum	Checksum
Cur Hop Limit	**Current Hop Limit:** Unsigned 8-bit int: Default value should be in packet's Hop Count field for all outgoing packets • Should be set to 0 if unspecified by rtr
M Flag	1-bit: Managed address config flag • Set to indicate addresses avail through DHCPv6 • If set: O flag becomes redundant/ignored BC DHCPv6 will provide config info
O Flag	1-bit: Other config flag • When set: Indicates other config info avail through DHCPv6 (like DNS)
Reserved	6-bit unused field: Must be set to 0

Rtr Lifetime	16-bit unsigned int: Indicates lifetime of default rtr in seconds • Can contain max value of 65535: Sending rules limit value to 9000 • Value of 0: Rtr isn't a default/shouldn't appear in default list • Rtr lifetime only applies to rtr's avail as a default
Reachable Time	32-bit unsigned int: Time, in ms that node assumes neighbor is reachable • If set to 0, reachable time unspecified by rtr
Retrans Timer	32-bit unsigned int: Time, in ms, bet retransmitted Neighbor Solicitation msgs • If field 0: Retrans time unspecified by rtr
Options	Include: Source LL address, MTU, Prefix info

MTU should be sent on links w/variable MTU: May be sent on other links

- Prefix Info options specify on-link prefixes and/or used for stateless address autoconfig

Redirect Messages
Message format

All fields in multiples of 4: 1st bytes: Fields Type, code, and checksum: 2nd set: Reserved: Last: Target/destination address/options

- Type value: 137: Code value 0
- Target Address: Default or 1st-hop rtr node is using to fwd traffic to destination address

Router Renumbering Messages Allows address prefixes on rtrs to be config/reconfig w/ease of ND/addr autoconfig

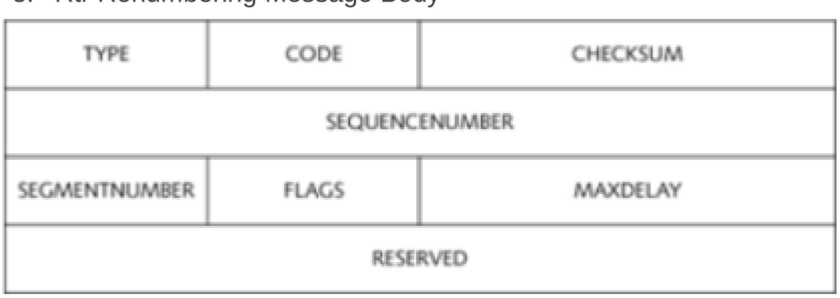
- Allows admins to update prefixes used/advertised by rtrs throughout a site
- Takes advantage of 128-bit address space: Allows admins to assign diff meanings to diff bits in addr structure

3 types of renumbering msgs:

1. Commands: Sent to rtrs: Code 0
2. Results: These are responses sent by rtrs: Code 1
3. Sequence # Reset: Used to sync a reset of seq #/cancel crypto keys: Code 255

Format:

1. Header/Extension Headers
2. Rtr Renumbering Header: 16 octets
3. Rtr Renumbering Message Body



Renumbering Message Header Fields

Field	Description

Name	
Type	138
Code 0	Rtr Renumbering Command
Code 1	Rtr Renumbering Result
Code 255	Seq # Reset
Checksum	Checksum
Seq #	Unsigned 32-bit seq # that must be non-decreasing bet seq # resets
Segment #	Unsigned 8-bit field: Contains values for diff types of rtr renumbering msgs having same SequenceNumber
Flags	**T: Test cmd:** • 0: rtr config to be modified • 1 test msg **R: Result requested:** • 0: Result mustn't be sent: • 1: rtr must send result after processing cmd **A: All ints:** • 0: cmd must not be applied to ints admin down down • 1: cmd must apply to all ints **S: Site specific:** • 0: cmd must be applied to ints regardless of site affiliation • 1: apply only to ints belong to same site **P: Processed previously:** • 0: Result msg contains complete report of processing cmd • 1: cmd msg previously processed
MaxDelay	Unsigned 16-bit field: Max amt of time in ms rtr must delay any reply to cmd msg
Reserved	Must be set to 0 by sender: Ignored by receivers

Comparison of ICMPv4/ICMPv6 Messages
Commonalities: Connectivity-checking | Error-checking | Info msgs | Fragmentation required
Only ICMPv6: Address Assignment/Resolution | Multicast Group Mgmt

PMTU Discovery in IPv4	rtrs can notify nodes via ICMPv4 if they need to change MTU size • Made up of a number of links bet source/destination node: Each link can have diff MTU size • PMTU: Size of smallest MTU for individual link • Packet size usually managed through fragmentation/PMTU Discovery • Originally: All packets set to MTU of 576 bytes • If packet MTU too large: Node receives msg Destination Unreachable
Changes to PMTU in IPv6	MTU sizing/fragmentation updated to improve the efficiency/quality of sending/receiving traffic • Default MTU packets: 1280 bytes • Eliminated fragmentation of packets once sent from source • IPv6 rtrs don't fragment packets in transit

Connectivity Testing w/Ping: Ping a form of ICMP echo comm
- An ICMP Echo packet consists of: Ethernet/IP/ICMP header and data
- Ping: Client transmits packet to target: Receipt: Target echoes back data
- Ping uses echo/reply requests: Most requests obtain an avg response time by sending out 7 requests
- Responses in ms following "time=,"
- Not evidence of round-trip time bet devices: Snapshot
- Ping lists: IP of device, bytes contained in response: Round-trip time: TTL in response packet

Ping in Server 2012/Win7/10 sends series of 4 ICMP echoes: 1 second reply timeout value
- Consist of 32 bytes of data in fragmentable packet

Most TCP/IP stacks don't allow ping to broadcast address BC all receiving hosts would respond to sender
- Typically don't response to requests sent to multicast/broadcast address

Flags for ping:

Flag	Description
-l	Size: # of data bytes to send
-f	Sets DF bit
-j	Sets value of TTL
-v	Sets TOS field value in header
-w	Sets timeout # of ms to wait for reply

Path Discovery w/Traceroute: Uses route tracing to ID path from sender to target

- Uses echo requests/some manip of TTL value in header
- List of rtrs along a path/round-trip latency time to each rtr
- Some versions try to resolve names of rtrs along path

Flags for Tracert: Windows Server 2012/7/10

Flag	Description
-d	Don't perform DNS reverse query on rtrs
-h	max_hops: Defines max TTL value
-w	Timeout indicates how long to wait for reply before displaying *

Pathping: CLI utility that uses echo packets to test rtr/link latency/packet loss
PMTU Discovery w/ICMP

- Using PMTU: Host always sets the DF bit in IP header to 1 header to 1: Packet can't be fragmented in path
- If packet to large: Receiver discards sends destination unreachable fragmentation needed/DF set to source
- After receipt: Reply indicates MTU size: PMTU host must reduce/retransmit or remove DF flag
- Process of PMTU discovery keeps going until end-to-end min MTU size discovered
- Host should be able to send the packet w/MTU that allows it not to be discarded

PMTU Discovery: Default all ver of Win since 2000
Can set 2 optional PMTU params

- **EnablePMTUDiscovery**
- **EnablePMTUBHDetect**

Neither default settings: Must be manually added to Registry
EnablePMTUDiscovery: Enables/disables PMTU Discovery on Win

EnablePMTUDiscovery	Registry Setting
Reg Info	Details
Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Data type	REG_DWORD
Valid range	0 or 1
Default value	1 (0 disables)
Present by default	No

EnablePMTUBHDetect: Defines if host should detect black hole routers: Silently discards packets w/out indicating cause

- Thwarts auto-recovery/auto-reconfig attempts
- PMTU host retries large MTU and if no response received: Auto sets PMTU to 576 bytes

EnablePMTUBHDetect	Registry setting
Reg Info	Details
Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Data type	REG_DWORD
Valid range	0 or 1

Default value	0
Present by default	No

Routing Sequences for ICMP

PerformRouterDiscovery	Registry Setting
Reg Info	Details
Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters \Interfaces\<interface>
Data type	REG_DWORD
Valid range	0 or 1
Default value	1 (changing to 0 disables rtr discovery)
Present by default	No
Windows Server 2012	Addl valid range option (2): Enable only if DHCP sends Perform Router Discovery

SolicitationAddressBCast: Can be config to use a subnet broadcast during Rtr Discovery of all-rtrs multicast address

SolicitationAddressBCast	Registry Setting
Reg Info	Details
Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters \Interfaces\<interface>
Data type	REG_DWORD
Valid range	0 or 1
Default value	0 (changing to 1 enables WinS2012/7/10 to use subnet broadcast to perform solicitation)
Present by default	No

Redirection to Better Router: ICMP can be used to point host to better router

Security Issues: ICMPv4

- ICMP: Info about network configs/connectivity status: Can use it to learn how network designed/config
- Recon: ICMP info-gathering tool
- Address scanning 1 method of obtaining list of active hosts
- Host probe performed by sending ping to each host w/in range: Noting responses
- Devices that reply considered valid targets

Port scanning: Once addresses of active devices known: Can be port scanned

- Many systems don't reply to pings sent to a broadcast address: Scans sent unicast to each possible address

ICMP Redirect Attack	ICMP can be used to manipulate traffic flow bet hosts • Attacker can redirect traffic to their machine: MiTM style attacks • Trust-based service exploitation: Connection hijacking/DoS/sniffing
ICMP Rtr Discovery	ICMP susceptible to attack on local network segment: Another MiTM • During discovery: Rtr Solicitation msg finds way to attacker's machine • Timing critical: Attacker must intercept solicitation: Stifle original response from rtr • Push out forged response before rtr does • Attacker spoofs response back to target host: ○ Machine is immediate rtr in question: Not actual rtr on segment ○ No auth performed ○ Recipient has no way of knowing response bogus
Inverse Mapping	When filtering device detected bet attacker/target: Interrogate routing device in unusual way • Intentionally sends packets to vacant network addresses • Receipt of packet destined for nonexistent host: Intermediary rtr will gladly

	pass it on anyway • ICMP: Stateless protocol • Packet reaches internal rtr? Replies w/Host Unreachable msg for every bogus entry • Attacker may logically deduce which addr correspond to live target
Firewalking	Walking a FW ACL/ruleset to determine what it filters/how **2 phase attack:** 1. Initial Traceroute to discover hop count to FW appliance: 2. Sending a packet w/TTL 1 greater than final hop count (bet attacker/FW) ○ Elicit a Time Exceeded response from beyond FW: Indicating live target

Sec Issues for ICMPv6

Sec features:

- Value in Hop Limit field set at 255
- Source addr of packets must be either LL/unspecified (::/128) for Advertisement/Solicitation messages
- No mech currently specified that would prevent an attacker on local from exploiting ICMPv6

Auth for ICMPv6 packet exchanges uses:

- IP Auth header: IPv6-AUTH
- IP Encapsulating Security Payload Header (IPv6-ESP)
- IPv6-ESP provides confidentiality for these exchanges

ICMPv6 is protected by Ipsec: Security bootstrap problem: Not avail when computer is at this state

- When booting: Network node sends rtr Solicitation req Advertisements from all local rtrs routers
- Rtr solicitations are completely insecure: ND msgs have same problem: No sec during booting
- Msgs totally dependent on IPsec Auth Header sec
- It runs on top of IPv6: Utilizes sec features like ARP: Insecure

Decoding ICMP Packets: WS can capture/decode packets

ICMPv4 Echo Request/Reply Msg Fmt Fields

Field Type	**Description**
Type 0	Echo reply: Type of msg
Code 0	Code for both request/reply msgs
Checksum	0xfb9c: No error found in header
Identifier	BE: Big-Endian/LE: Little-Endian each have separate entry • Both refer to which bytes most significant: BE/least significant: LE in multibyte data types • Describes how the sequence of bytes is stored in mem
Sequence #	BE: Big-Endian/LE: Little-Endian each have a separate entry • Both refer to which bytes are most significant: BE/least significant: LE in multibyte data types • Describes how seq of bytes stored in mem
Response Time	151.909 ms: Amt of time in ms responding host took to reply to echo
Data	Encapsulated data payload for reply expressed: 64 byte length

ICMPv6 echo request/reply msg Fmt Fields

Field Type	**Description**
Type 129	Echo reply
Code 0	Code for both request/reply
Checksum	0xcbe9: No error found
Identifier	Request/reply msgs use single identifier: WS doesn't provide BE/LE entries
Sequence number	164
Data	Encapsulated data payload for reply msg expressed: 32 byte length

Post 7

Thursday, January 24, 2019 11:46 PM

WCNA NOTES: CH 6: NEIGHBOR DISCOVERY

October 12, 2018 Moo Comments 0 Comment

Neighbor Discovery:

ND: Permit nodes to find out what link located on/LL prefixes/Where link's reside/neighbors/active neighbors

- Associates LL addr w/ipv6 addr: Info about node comm on network

Uses 5 msg types:

1. RS: Rtr Solicitation
2. RA: Rtr Advertisement
3. NS: Neighbor Solicitation
4. NA: Neighbor Advertisement
5. Redirect

Rtr Solicitation	When int becomes active: Node may send RS: Asks any rtrs connected to local link to ID themselves • Type: 133
Rtr Advertisement	Send out msgs: More LL address/network prefix/MTU for LL/hop limit values/params • Can have flagged params to indicate type of addr autoconfig process • Type: 134
Neighbor Solicitation	Can send to find/verify LL addr for local node/see if avail/check own addr not in use • **DAD: Duplicate Address Detection:** When checking to see if addr isn't in use by a node • Type: 135
Neighbor Advertisement	When LL address changes: Sends msg that includes IPv6/LL addr • Helps establish adjacency • Type: 136
Redirect	When rtr knows better 1st hop for dest addr: Sends: ID's it exists on same network segment • Traffic load balance on multiple ints • Type: 137

2 types of IPv6 nodes: routers/hosts:

1. Rtrs: Fwd IPv6 packets not addr to selves
2. Hosts: Any node not a rtr

Only sparing use of msgs: Although NS uses multicast: NA send in response to unicast

- ND: Use of multicasts w/LL scope: **ff02::2 ||** Addr w/LL scope **ff02::1**
- Solicited-node address: Multicast w/LL scope: Reduces # of multicast groups nodes must sub to
- Single node: May have multiple unicast/anycast addr
- Higher-order bits: Prefix: Each req to join solicited-node addr for each unicast/anycast addr assigned to it
- Solicited-node addr: **ff02::1:ff**

IPv6 ND vs. IPv4

- ND takes over functions ARP/RARP handled
- Performs many of the functions that ICMP Rtr Discovery/Redirect in IPv4: More compact/efficient

IPv6	**IPv4**
Neighbor Solicitation	ARP Request
Neighbor Advertisement	ARP Reply
Rtr Solicitation	Rtr Solicitation
Rtr Advertisement	Rtr Advertisement
Redirect	Redirect

Duplicate Addr Detection	Gratuitous ARP
Neighbor cache ARP cache	

Router Solicitation: When host's int initializes: May not wait for next advertisement: may send solicitation to find rtrs on network

- If so: Learn prefix/params

Composed of:

- Ethernet header: Source addr/MAC of host int
- Dest addr: 33:33:00:00:00:02.

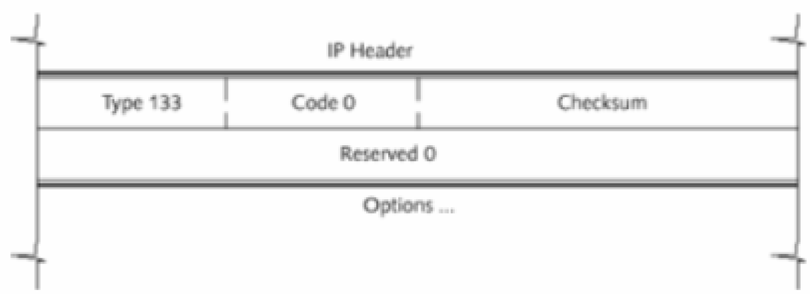
IPv6 header:

- Source addr: IPv6 addr of int/unspecified address
- Dest addr: LL scope all-rtrs multicast ff02::2

Hop Limit: 255

Msg Fmt

ICMP Field	Description
Type	133
Code	0
Checksum	
Reserved	Unused: 0 by source/ignored by dest
Options	RFC 4861: Source LL addr only valid for msg type if known • Addr not included if unspecified • Should be included on LL's that have addr



Router Advertisement: Inform hosts of link prefixes/addr autoconfig/link MTU/valid preferred lifetimes/options

- Reply to solicitations received by using advertisement

Composed of:

Ethernet header:

- Source addr: MAC for host int
- Dest addr: 33:33:00:00:00:01

IPv6 header:

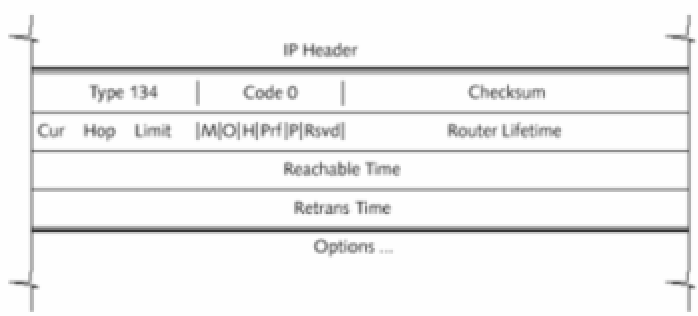
- Source addr: LL addr for int
- Dest addr: LL scope: All-nodes multicast address ff02::1/source addr for int

Hop Limit: 255

Msg Fmt

ICMP Field	Description
Type	134
Code	0
Checksum	
Cur Hop Limit	**Unsigned 8-bit int:** Default value should be in packet's Hop Count field for all outgoing packets • Should be set to 0 when unspecified by rtr
M Flag	**1-bit Managed Addr:** Set to indicate addr avail through DHCPv6 • If M flag set: O becomes redundant/can be ignore: DHCPv6 will provide all avail config info

O Flag	**1-bit Other:** Set? Indicates other config info avail through DHCPv6 [Ex. DNS]
H Flag	**1-bit Home Agent:** Indicates to host rtr also functioning as Mobile home agent
Prf Flag	**2-bit Default Rtr Preference:** Tells hosts to prefer this rtr over others • If Rtr Lifetime set to 0: This flag must be set to 00 • Valid values: ○ 11: Low ○ 00: Medium [default] ○ 01: High ○ 10: Reserved: If received must act as if 00
P Flag	**1-bit Proxy:** Experimental: Not req
Reserved	**2-bit unused:** Must be 0 by sending rtr/ignored by receiving node
Rtr Lifetime	**16-bit unsigned int:** Lifetime of default rtr in sec: Max value 65535: Sending rules limit to 9000 • 0: Rtr not default/shouldn't appear in default list: Only applies to avail as default
Reachable Time	**32-bit unsigned int:** Time in ms that node assumes neighbor reachable after reachability confirmation • 0: Unspecified by rtr
Retrans Timer	**32-bit unsigned int:** Time in ms bet retransmitted NS msgs: 0: Unspecified
Options	Source LL addr/MTU/Prefix info/Advertisement Interval/Home Agent/Route Info • Source LL addr: Addr of int advertisement sent: Only used on LL that has addr ○ Rtr may omit to enable inbound load sharing across multiple LL addr • MTU • Prefix Info: On-link prefixes/and/or used for stateless addr autoconfig ○ Should include all rtr on-links so multihomed hosts have complete prefix info • Advertisement Interval: Time in MS bet Advertisement msgs sent by rtr • Home Agent Info: 2 options for use by node if set • Route Info: Prefixes to node to be include in r-table



Neighbor Solicitation: Find/verify LL addr of local node/see if node still avail/check own address not in use: DAD

- When node resolving addr: Sends multicast
- When verifying reachability of neighbor: Sends unicast

Composed of:
Ethernet header:

- Source addr: MAC of host int
- Dest addr: MAC of solicited-node addr: Multicast NS/MAC of unicast addr: Unicast NS

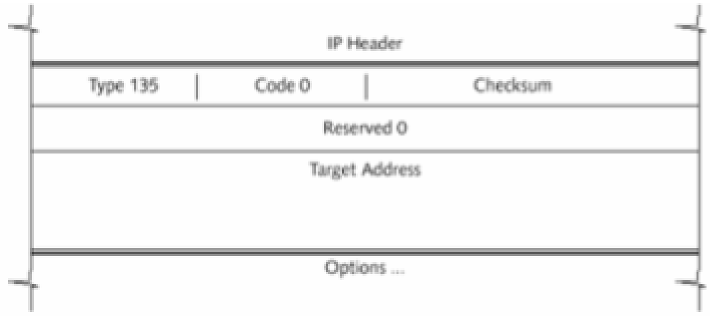
IPv6 header:

- Source addr: IPv6 of int/unspecified for DAD: Duplicate Addr Detection
- Dest addr: Either solicited-node addr: multicast/unicast NS

Hop Limit: 255
Msg Fmt

ICMP Field	Description
Type	135
Code	0

Checksum	
Reserved	Unused: 0 by source/ignored by dest node
Target Addr	Addr of target: Can't be multicast
Options	Source LL addr only valid if known



Neighbor Advertisement: Responds to solicitation when req: If own LL addr changes: Will send another
Composed of:
Ethernet header:

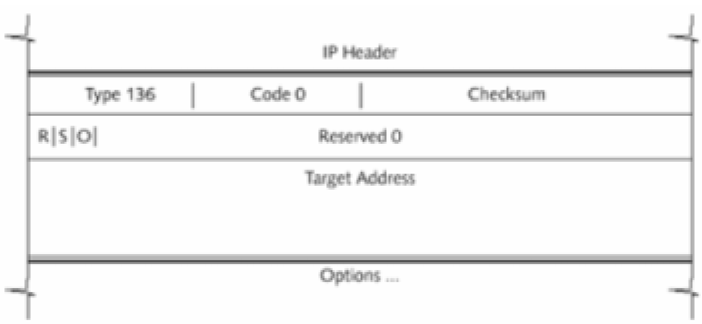
- Source addr: MAC of host int
- Dest addr: Either unicast MAC of NS/33:33:00:00:00:01 an unsolicited NA

IPv6 header:

- Source addr: IPv6 of int
- Dest addr: Source of NS/if source unspecified all-nodes multicast ff02::1

Hop Limit: 255
Msg Fmt

ICMP Field	Description
Type	136
Code	0
Checksum	
R Flag	**1-bit Rtr:** Informs nodes msg came from rtr: If rtr changes to host: • Uses flag in Neighbor Unreachability Detection
S Flag	**1-bit Solicited:** Msg reply to NS: Can't be set in unsolicited unicast/multicast advertisements
O Flag	**1-bit Override:** Informs node to update cached LL addr/existing cache entry • If node receives/doesn't have cache entry for LL: Updates cache even if flag not set • 1: Solicited/unsolicited msgs except for anycast/solicited proxy advertisements
Reserved	Unused: 0 by source/ignored by dest node
Target Addr	IPv6 of node sending NS: If unsolicited NA: Addr of LL hasn't changed
Options	Target LL addr: Source's node addr: Only valid option



Redirect: Informs host of better 1st-hop rtr for dest: Also informs host dest node on-link
Composed of:
Ethernet header:

- Source addr: MAC of host int
- Dest addr: Unicast MAC

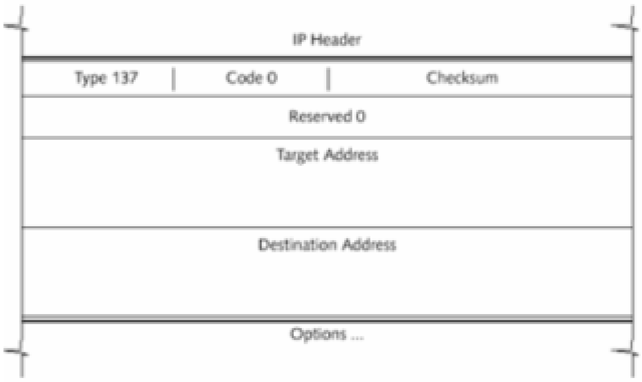
IPv6 header:

- Source addr: LL addr of int
- Dest addr: SA of node that triggered redirect

Hop Limit: 255

Msg Fmt

ICMP Field	Description
Type	137
Code	0
Checksum	
Reserved	Unused: Set to 0 by source/ignore by dest
Target Address	IPv6 of default/better 1st-hop rtr node is using to fwd traffic to dest: • Can have same addr as dest addr if endpoint • If rtr: Contains LL for rtr int directly connected to local link on where source loc
Dest Addr	Dest
Options	Target LL addr/Redirect Header: As much of original packet that triggered redirect: Doesn't exceed 1,280 bytes



Option Fmt

Type	Option Name	Reference
1	Source LL Addr	RFC 4861: ND
2	Target LL Addr	RFC 4861
3	Prefix Info	RFC 4861
4	Redirected Header	RFC 4861
5	MTU	RFC 4861
7	Advertisement Interval	RFC 6275: Mobility Support
8	Home Agent Info	RFC 6275
24	Route Info	RFC 4191

Source/Target Link-Layer Options

Source Options: Type 1: Length 1 (if eth0): Source LL Addr

Target Options: Type 2: Length 1 (if eth0): Target LL Addr

Prefix Info Option: Used in RA msgs: Contains info for on-link addr/prefixes for addr autoconfig

Option Fmt

Type	Description
Type	3
Length	4
Prefix Length	8-bit unsigned int: Indicates # of leading bits that constitute prefix addr: 0-128: • When combine w/L flag: Necessary prefix info for on-link determination
L	**1-bit On-Link flag:** Indicates prefix address can be used for on-link determination

	• When not set: Advertisement doesn't imply prefix is either on/off-link
A	**1-bit Autonomous Addr-Config:** Indicates prefix addr can be used for stateless addr autoconfig
R	**1-bit Rtr Addr:** Indicates prefix complete rtr addr: Used when rtr acting as Home Agent • Interpretation independent of On-Link (L)/Autonomous Address-Config
Reserved1	Unused: Set to 0 by source/ignored by dest
Valid Lifetime	32-bit unsigned int: Valid lifetime in sec of on-link prefix/for stateless addr config: All 1 bits would be infinity
Preferred Lifetime	32-bit unsigned int: Valid lifetime in sec of addr generated by using prefix/stateless addr autoconfig • Value must not exceed Valid Lifetime field: All 1 bits would be infinity
Reserved2	Unused: Set to 0 by source/ignored by dest
Prefix	IPv6 addr/prefix address of on-link segment used by nodes for stateless autoconfig • Bits in field combined w/bit value of Prefix Length make complete IPv6 prefix address • If combined value of 2 fields less than 128 bits: Remaining bits must be 0/ignored by node • Rtr must not send LL prefix/host should ignore prefix if received

Redirected Header Field Type: 4: Length: 8-byte blocks: Reserved: IP header/data: Doesn't exceed 1,280 bytes
MTU Option: Sent in RA msgs to provide common MTU value for nose on same segment
Fields: Type 5: Length 1: Reserved: MTU: 32-bit unsigned int: Recommended MTU for link [usually 1500]
Advertisement Interval Option: Mobile IPv6: RA for their movement detection alg
Fields: Type 7: Length 1: Reserved: Advert Interval: 32-bit unsigned int: Time in ms bet unsolicited RA's sent by rtr
Home Agent Info Option: May include in RA's
Fields: Type 8: Length 1: Reserved:

- HA Pref: 16-bit unsigned int: Determines pref order for avail HA's
- Lifetime: 16-bit unsigned int: Valid lifetime in sec of HA: Max value 18.2 hours: 0 not valid

Route Info Option: Sent in RA to specific individual routes for hosts to add to default Rtr list

Type	Description
Type	24
Length	1
Prefix Length	8-bit unsigned int: Indicates number of leading bits that make up prefix addr: 0-128 bits
Resvd	Unused: Set to 0 by source/ignored by dest
Prf	2-bit flag: Indicates hosts to prefer this rtr over others: If value 10 received: Node ignores rtr info option
Resvd	Unused: Set to 0 by source/ignored by dest
Route Lifetime	32-bit unsigned int: Valid lifetime in sec prefix valid for route determination: All 1 bits sets infinity
Prefix	IPv6 addr/prefix of on-link segment: Value less than 128 bits? Remaining bits must be set to 0/ignored

Conceptual Host Model: Primarily concerned w/op by hosts
Storing Neighbor Data on Host: For node to comm w/neighbor needs: LL addr/if host-or-rtr/recent comm/list/params
Node needs to store following:

Neighbor cache	Table of info containing on-link addr for each neighbor: • May include LL addr/state of reachability/whether neighbor host/or/rtr
Dest cache	Table of info containing data about dest traffic sent: On/Off-link nodes • Dest IPv6 mapped to next-hop addr of neighbor • Data not related to ND may be stored in dest cache: PMTU/RTT • May be updated Redirect msgs

Prefix list	Table of info containing data from RA msgs of on-link prefix addr • Each entry has invalidation timer: Can expire prefixes as they become invalid • LL prefixes have infinite invalidation timer regardless of whether RA msg received for it/not
Default rtr list	IP addr of rtrs that sent RA msgs: Each entry also includes invalidation timer value

Conceptual Sending Algorithm: For a node to comm w/a neighbor node:

- Needs to find IP of next-hop by examining dest cache to learn associated LL addr
- If node doesn't have addr avail: Invokes process called: Next-hop determination
    - Populates its caches/lists w/neighbor's addr info: Process AKA conceptual sending algorithm

ND Process:

Addr Resolution	Discovering on-link neighbors using NS/NA msgs
NUD	Neighbor Unreachability Detection: Determine whether/not neighbor previously comm w/remains avail
DAD	Duplicate Addr Detection: Determine if assigned IPv6 in use
Rtr Discovery	Nodes discovering default GW's/prefixes if avail
Redirect Function	Rtrs informing hosts better 1st-hop

NUD: Neighbor Unreachability Detection:

Used for node-to-neighbor-node verification of on-link comm: Host-to-host/-rtr/rtr-to-host

Neighbors reachable if: Recent comm by upper-layer protocol (TCP)/node recently sent NS msg

5 states for neighbor cache entry:

INCOMPLETE	Addr resolution being performed on entry: NS msgs sent to solicited-node multicast addr of target: • Corresponding NA msg hasn't been received • If NA msgs not received after **MAX_MULTICAST_SOLICIT** (default value of 3 transmissions): ○ Resolution failed: Neighbor entry rem from cache
REACHABLE	Neighbor considered reachable when either solicited NA msg received or: • Upper-layer protocol comm fwd progress received w/in **REACHABLE_TIME** var
STALE	After **REACHABLE_TIME** of 30,000 ms in comm w/neighbor: entry changed to **STALE** • If unsolicited ND msg also advertises LL received: Entry changed to**STALE** • Ensures proper addr resolution process when the node needs to send packet to neighbor
DELAY	After **STALE**: Node sends 1st packet to neighbor: State changes to **DELAY** • **DELAY_FIRST_PROBE_TIME** var set to default of 5 sec • If reachability msg received w/in timer: State changes to **REACHABLE** • Otherwise: State changed to **PROBE** • **DELAY** allows upper-layer protocols time to provide reachability confirmation
PROBE	Node sends unicast NS msgs to cached LL addr of neighbor based on**MAX_UNICAST_SOLICIT** var • Default value of 3 transmissions • **RETANS_TIMER**: Default value of 1,000 ms • Neighbor rem from table if max # of retransmissions/time exceeded/no response received

Post 8

Thursday, January 24, 2019 11:46 PM

WCNA CH 7 NOTES: IP ADDR AUTOCONFIGURATION

DHCP: Dynamic Host Config Protocol: Way for client that lack IP to request them: Static addr allocation

- Manages addr allocations: Centralized config

Origins: BOOTP: Bootstrap Protocol: 70's: Diskless workstations for startup across network

- DHCP packets: Similar msg fmt as BOOTP: Backward compatibility to

All DHCP servers on same broadcast domain receive req/send back unicast

Role of Leases: addr loans for specific time: Lengths vary: Avg bet 1-3 days

- 4-8 hrs common on ISP networks where clients come/go all time

3 pieces of SW work together:

Client	Enabled at client when selecting "Obtain IP addr auto" in TCP/IP • SW broadcasts req for service/lease renewal on behalf of clients/handles address config
Server	Listens/responds to client/relay agents for addr services: Manages addr pools/related config data
Relay agent	Broadcast addr req to network segments: Not fwded through rtrs: Relay agent intercepts req locally • Repackages as unicast to 1/more DHCP servers: Uses MAC to fwd reply to client

Lease Types: 2 Types of addr leases:

1. Manual
2. Dynamic

Manual	Assigned: Manually associates client's HW w/specific IP leased
Dynamic	DHCP server assigns addr for specific time

DNS isn't dynamic: All addr updates entered manually

2 types of IPv4 autoconfig on host int:

1. DHCP

- **APIPA: Automatic Private IP Addressing**

APIPA: Auto Private IP Addr: Dynamic config of IPv4 LL addresses: Initially MS: Win98: Adopted by manufacturers

- Used by ints as failover mech to self-assign an IPv4 addr if initial DHCP req not answered
- Network int will continue to send DHCP req every 5 min
- If server subsequently replies w/IP addr assignment for host: APIPA released from int in favor of DHCP
- Ops only ints config for DHCP: If manual IP assigned to int: APIPA disabled
- Addr assignment: Pseudo-RNG
- Allows hosts to comm on local link of network, although won't be routed comm to hosts on other networks

IPv6: 2 approaches:

1. **Stateless:** Presents req rtr config info to all comers
2. **Stateful:** DHCPv6: Considered stateful autoconfig: Server must maintain awareness of pool of avail addr

Stateless Autoconfig	Segments/nodes that support multicasting: Many tools to support stateless autoconfig • ND allows rtrs configs to present min info host needs when joining network link • Includes: Prefix of segment/rtr's own addr/MTU/Preferred # of max hops for various routes When int initializes network segment/link: 1st configs own link-local addr • Calcs its own 64-bit int ID: IID: Either EUI-64/Privacy • It forms a link-local addr by appending IID to well-known link-local prefix of fe80::/64

	• While performing duplicate addr check for LL addr:
	  ○ Host will send RS to prompt any attached rtrs to send their RA
	  ○ Rtr can provide prefix in order for host to add IID to create a global unicast
	Spoofing attacks: Hosts use default value of 2 hr valid lifetime when encountering any RA
	• When an RA auth nodes update valid lifetime of addr as directed

Stateful Autoconfig	Rtr may be config to not supply network prefix in RA but flags for host obtain addr via DHCPv6 **Significant diff in DHCPv6:** • Clients fully functioning hosts/able to search actively for DHCPv6 server using multicast solicitations • Clients can discover whether DHCPv6 servers are on local link • Can use a relay server on local segment to receive config info from off-link server • DHCPv6 server doesn't supply default GW addr to host: Host derives info from RA • If RA flag set to off: M flag set to on: Informs host to obtain addr from DHCPv6 server **Shared features w/stateless autoconfig in IPv6:** • All autoconfig addr leased/use same "dual lifetimes" paradigm for name lease renewal • Can be set up to dynamically update DNS records

IPv6 Autoconfig Addr Options

	RA-ICMP		RA-Prefix Info				
Autoconfig Method	**M Flag**	**O Flag**	**A Flag**	**L Flag**	**Prefix Derived**	**IID Derived**	**Other config Options**
SLAAC	0	0	1	1	RA	EUI-64/Privacy	Manual
Stateful: DHCPv6	1	n/r	0	1	DHCPv6	DHCPv6	DHCPv6
SLAAC/DHCPv6	0	1	1	1	RA	EUI-64/Privacy	DHCPv6
Stateless/DHCOv6	1	1	1	1	RA/DHCPv6	EUI-64 or Privacy and DHCPv6	DHCPv6

Functional States of IPv6 Autoconfig Addr

Tentative addr	Happens as node initializes int on network segment/link in order to config its own link-local addr • To verify: Node sends NS w/addr as dest: If another node responds must stop autoconfigure • DAD: If no duplicates found: Addr is valid: Node assigns int as preferred
Valid addr	Usable based on Valid Lifetime field in Prefix Info option of an RA/Valid Lifetime field in DHCPv6 IA • IA: ID Association Addr option: Either preferred/deprecated state of op • Valid lifetime value must be => than preferred lifetime value • When valid lifetime expires: Addr invalid
Preferred addr	Usable for all comm based on Preferred Lifetime field in Prefix Info of RA • Or Preferred Lifetime field in DHCPv6 IA Address option • When expires but valid lifetime still valid: Addr moves to deprecated
Deprecated addr	Allow nodes to continue to function while they renew leases: • May be used normally but shouldn't be used for anything other than completion of

	sessions • While in this state: If another node initiates new session: ○ Host will continue to receive/send traffic ○ When valid lifetime expires: Addr becomes invalid
Invalid addr	Can't be used as either source/dest addr when valid lifetime expires

Node Interface Identifiers: Node IID's

- Used to ensure IPv6 addr unique: Generally 64 bits long: Can be construed from diff sources

3 most common:

1. Modified EUI-64 fmt
2. RNG to create 64-bit #
3. CGA: Crypto Generated Addr process

After IID computed: Process for creating complete addr via various autoconfig options will continue
Modified EUI-64: Based on IEEE-defined 64-bit extended unique identifier (EUI-64)

- MAC padded w/0xFF/0xFE bet leftmost/rightmost 3 bytes
- 7th byte: AKA "u"/universal/local bit: Inverted: Allows addr to indicate universal scope

CGA: SEND: SEcure ND protocol must be running on network:

- Addl fields for ND to exchange keys
- Difficult to deploy/processor intensive: Can be attacked easily w/NS flood toward SEND-enabled node
  - Causes slowdown of sys as it attempts to process all public key ops

DHCPv6: **Uses diff UDP ports than DHCP**: Clients listen on UDP 546 and in DHCPv6 it's 547
2 specific multicast addresses:

1. ff02::1:2
2. ff05::1:3

ff02::1:2	Link scope multicast used by clients to comm w/on-link servers/relay agents
ff05::1:3	Site-scope multicast addr used by relay agent to comm w/server/on-site servers

Addr not bound to MAC: Bound to DUID: DHCP Unique Identifier

- DUIDs: Must all be globally unique: Each client/server must have 1
- Shouldn't change after initial assignment, even if HW does

B/C addr bound to a DUID: Binding table/log file may not provide much assistance for troubleshooting

- DUID may not have any info w/in it that uniquely ID's specific HW int/device

DUIDs: Defined in 1 of 3 methods:

DUID-LLT	LL addr plus time
DUID-EN	Vendor-assigned UID based on Enterprise #
DUID-LL	Link-layer addr

IA: Identity Association: Mech for servers/clients to ID/manage group of IPv6 addr

- Composed of IAID: ID Association Identifier/Config info
- Each host must have a unique IAID for each int
- IAID for specific IA must be maintained after restarts of host

When host sends Solicit req to DHCPv6 server: Client provides IAID assigned to int in req

- Server captures/stores IAID in lease table

DHCPv6 Msgs

Field	Description
Msg-type	1-byte field: Defines msgs sent bet nodes/servers/relay agents
Transaction-id	3-byte field: Transaction ID for specific msg exchange
Options	Var-sized field sent by node req IPv6 addr/other info like DNS addr

IPv6 DHCPv6 Msg Types

Message	Value	From/To	Description
SOLICIT 1	Node	Server	Sent by nodes to locate DHCPv6 servers
ADVERTISE	2	Server/Node	Sent by servers replying to node Solicit msg
REQUEST	3	Node/Server	Sent by node req IPv6 addr/possible other info such as DNS addr

CONFIRM	4	Node/Server	Sent by node to any server: Verifies addr is still valid on link
RENEW	5	Node/Server	Sent by node to server where received original info to extend timers
REBIND	6	Node/Server	Sent by node to any server if it didn't receive reply to Renew req
REPLY	7	Server/Node	By server: Reply to Solicit/Renew/Rebind/Release/Decline/Info-Req
RELEASE	8	Node/Server	Sent by node informing server it no longer using assigned addr
DECLINE	9	Node/Server	Sent by node to inform server addr it was assigned already in use
RECONFIGURE	10	Server/Node	By server to node so can exe Renew/Info-Req to receive new info
INFO- REQUEST	11	Node/Server	Sent by node to req only info: Occurs when O flag on/M flag off in RA
RELAY-FORW	12	Relay Agent	Server: By relay agent on behalf of node req when server not on-link
RELAY-REPL	13	Server/RAgent	Sent by server in reply to msg sent by relay agent
LEASEQUERY	14	Node/Server	Sent by node to any avail server to get info on leases
LEASEQUERY-REPLY	15	Server/Node	Sent by server to inform client of lease info
LEASEQUERY-DONE	16	Server/Node	Sent by server indicating end of group LEASEQUERY replies
LEASEQUERY-DATA	17	Server/Node	Sent by server if more than 1 client's data to be sent for LEASEQUERY

DHCPv6 Options Fields

Field	**Description**
Option-code	2-byte field: Contains specific option
Option-len	2-byte field: Contains option-data fields length
Option-data	Var-sized field containing data for option

DHCPv6 Options Fields

Option	**Value**	**Description**
OPTION_CLIENTID	1	Client to provide server its DUID
OPTION_SERVERID	2	Server to provide client its DUID
OPTION_IA_NA	3	Carries IA for non-temp addr/params
OPTION_IA_TA	4	Carries IA for temp addr/params
OPTION_IAADDR	5	Specifies IPv6 addr/options for IA_NA/IA_TA
OPTION_ORO	6	Client specifies list of options being req from server
OPTION_PREFERENCE	7	Server influences selection of server for client
OPTION_ELAPSED_TIME	8	Client indicates how long client has been trying to complete DHCPv6 transaction
OPTION_RELAY_MSG	9	Contains DHCPv6 msg in relay-fwd/relay-reply msg
OPTION_AUTH	11	Contains info to auth contents/ID of DHCPv6 msg
OPTIN_UNICAST	12	Server informs client it can comm to server via unicast addr
OPTION_STATUS_CODE	13	Returns status code relating to DHCPv6 msg

OPTION_RAPID_COMMIT	14	Client informs server it can support 2-msg exchange for addr assignment
OPTION_USER_CLASS	15	Client informs server of type of usr/app it is, so server can supply config info
OPTION_VENDOR_CLASS	16	Client ID's vendor of HW client is operating
OPTION_VENDOR_OPTS	17	Servers/clients exchange vendor-specific info
OPTION_INTERFACE_ID	18	Relay agent sends int info on where client msg received
OPTION_RECONFIGURE_MSG	19	Server sends reconfig msg to client: Informs to reply w/a Renew/info-request msg
OPTION_RECONF_ACCEPT	20	Client informs server to accept reconfig msg; server to inform client same
OPTION_SIP_SERVER_D	21	SIP server domain list; SIP outbound proxy server for clients to use
OPTION_SIP_SERVER_A	22	SIP server IPv6 addr for clients to use
OPTIN_DNS_SERVER	23	IPv6 addr of DNS recursive name server
OPTION_DOMAIN_LIST	24	Domain list
OPTION_IA_PD	25	Carries prefix delegation ID and params/prefixes associated with
OPTION_IAPREFIX	26	Specifies IPv6 addr prefixes for IA_PDs
OPTION_NIS_SERVERS	27	NIS server list for nodes
OPTION_NISP_SERVERS	28	NIS+ server list for nodes
OPTION_NIS_DOMAIN_NAME	29	NIS servers inform client of NIS Domain Name
OPTION_NISP_DOMAIN_NAME	30	NIS+ servers inform client of NIS+ Domain Name
OPTION_SNTP_SERVERS	31	SNTP server avail via IPv6 for nodes
OPTION_INFORMATION_REFRESH_TIME	32	Server informs client to refresh other config info: No timers
OPTION_BCMCS_SERVER_D	33	BCMCS Control Server Domain Name List
OPTION_BCMCS_SERVER_	34	BCMCS Control Server IPv6 addr
OPTION_GEOCONF_CIVIC	36	Defines civic loc of client/DHCPv6 server
OPTION_REMOTE_ID	37	Relay agents terminate perm/switched circuits to ID remote client end of the circuit
OPTION_SUBSCRIBER_ID	38	Providers separately ID sub systems
	39	Allows client to inform DHCPv6 of FQDN
OPTION_PANA_AGENT	40	Contains IPv6 addr for PANA Auth Agents avail to PANA client
OPTION_NEW_POSIX_TIMEZONE	41	POSIX TZ str: Express time zone info in char-based string
OPTION_NEW_TZDB_TIMEZONE	42	Refs name from time zone entry of TZ DB
OPTION_ERO	43	Relay agent to DHCPv6 server req list of options from server
OPTION_LQ_QUERY	44	ID query in LEASEQUERY msg
OPTION_CLIENT_DATA	45	Client on link to encapsulate data in LEASEQUERY-REPLY msg
OPTION_CLT_TIME	46	ID's how long ago server comm w/client

OPTION_LQ_RELAY_DAT	47	ID's how long ago client comm w/server
OPTION_LQ_CLIENT_LINK	48	Client: LEASEQUERY-REPLY msg to ID links client has 1/more bindings
OPTION_MIP6_HNINF	49	Mobile node exchange home network info w/DHCPv6 server
OPTION_MIP6_RELAY	50	Relay agent sends home network info of mobile node to DHCPv6 server
OPTION_V6_LOST	51	Allows client to obtain LoST server domain name
OPTION_CAPWAP_AC_V6	52	Contains 1/more IPv6 addr of CAPWAP ACs avail to WTP
OPTION_RELAY_ID	53	Contains DUID from relay agent
OPTION-IPv6_Address-MoS	54	MoS IPv6 addr option for DHCPv6 server
OPTION-IPv6_FQDN-MoS	55	MoS Domain List
OPTION_NTP_SERVER	56	Provides 1 addr for either NTP/SNTP server
OPTION_V6 _ACCESS_DOMAIN	57	Provides domain name for an access network
OPTION_SIP_UA_CS_LIST	58	Provides list of domain names for SIP User Agent Config Service Domains
OPT_BOOTFILE_URL	59	DHCPv6 server sends URL for boot file for client to use
OPT_BOOTFILE_PARAM	60	DHCPv6 server to specify params for boot file for client
OPTION_CLIENT_ARCH_TYPE	61	Client informs DHCPv6 server supported arch so server can supply correct boot file
OPTION_NII	62	Client to inform DHCPv6 server it supports Universal Network Device Int (UNDI)
DHCPv6 GeoLoc Option	63	Client to inform DHCPv6 server of coordinate-based geo loc
OPTION_AFTR_NAME	64	Provides Addr Family Transition Rtr's (AFTR's) FQDN to B4 element
OPTION_ERP_LOCAL_DOMAIN_- NAME	65	Client req ERP Local Domain Name from DHCPv6 server
OPTION_RSOO	66	Relay agent sends RSOO in -Fwd msg to server options: Sent by DHCPv6 server passed through relay agent

DHCPv6 Relay-Fwd Msg Fmt Fields

Field	**Description**
Hop-count	1-byte field: Indicates # of relay agents that have relayed msg
Link-addr	16-byte field: Global addr of relay agent on same segment as req node: • For server to supply correct scoped addr in reply msg
Peer-addr	16-byte field: Node's addr msg came from
Options	Var-sized field: Includes relay msg option/other options from relay agent

DHCPv6 Relay-Reply MSG Fmt Fields

Field	**Description**
Hop-count	Duplicated from original Relay-Fwd msg
Link-addr	Duplicated from original Relay-Fwd msg
Peer-addr	Duplicated from original Relay-Fwd msg
Options	Var-sized field: Includes relay msg option/other options from relay agent

DHCPv6 Process:

1. Host sends RS
2. Rtr replies w/RA: A flag set to off: L/M flags set to on

3. Host sends Solicit msg looking for DHCPv6 servers at link-scope multicast ff02::1:2
4. DHCPv6 server replies to host w/advertise msg: Informs host it can provide addr/other config options
5. Host sends Req msg: Req addr/other config options
6. DHCPv6 server sends host reply msg w/addr/other config options/timers info

DHCPv6 Stateless Process:

1. Host sends RS
2. Rtr replies to host w/RA: it's A/L/O flags set to on: M flag set to off
3. Host sends Info-Req msg: Req only other config options
    1. Looks for any DHCPv6 servers at link-scope multicast ff02::1:2
    2. Host also uses supplied prefix for network prefix to prepend to IID it generated
4. DHCPv6 server sends host Reply msg w/other config options it may have avail

DHCPv6 Relay Process:

1. Host sends RS
2. Rtr replies to host w/RA: A flag set to off: L/M flags set to on
3. Host sends Solicit msg looking for any DHCPv6 servers at link-scope multicast ff02::1:2
4. Rtr relay-fwds host Solicit msg to DHCPv6 server
5. DHCPv6 server relay-replies to rtr w/Advertise msg: Informs host it can provide addr/other config
6. Rtr replies w/Advertise msg to host
7. Host sends Req msg: Req addr/other config options
8. Rtr relay-fwds host Req msg to DHCPv6 server
9. DHCPv6 server relay-replies to rtr w/Reply msg w/addr/other config options
10. Rtr replies w/Reply msg to host

All cases of IPv6 addr autoconfig: DAD process must be executed to verify uniqueness of assigned IPv6 addr

Post 9

Thursday, January 24, 2019 11:47 PM

WCNA NOTES: CH 9: DNS (SMALL RECAP ALREADY HAVE STUFF ON THIS)

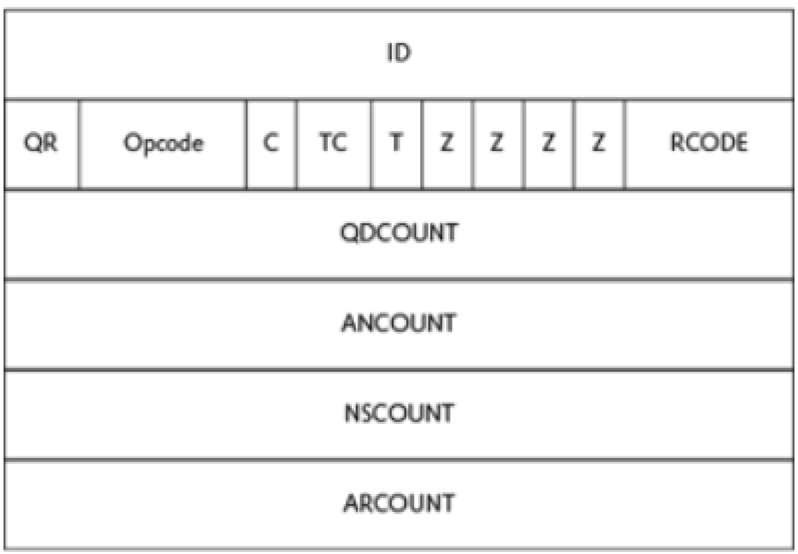
Network Name Resolution Protocols: Procedures that govern rules used in manual/dynamic name resolution

- Provide the definitions/mechanisms involved in client/server apps

<table>
<tr>
<td>LLMNR</td>
<td>Link-Local Multicast Name Resolution: RFC 4795: Based on DNS packet fmt
• Allows IPv4/6 nodes to perform resolution for other devices connected to same local link
• Server12/WIn7+ support
• Limited to single network segment: Not designed to cross rtr boundaries
• Name resolution services in envs where DNS can’t be used: Smaller networks
• Can be sent using TCP
<u>LLMNR exchange:</u>
1. LLMNR sending node transmits query to link-scope multicast
2. Responding node answers query if authoritative for name in query: Responds w/unicast UDP
3. After receiving response: Processes info/completes name-to-addr resolution</td>
</tr>
</table>

DNS servers: Authoritative: Portion of namespace assigned to them
LLMNR hosts: Authoritative: Names specified to them
LLMNR packet fmt: Like DNS packet: RFC 1035: Queries/responses: Sends UDP queries/resp large enough w/out req fragmentation

- **Default packet size:** MTU uncertain: 512 octets
- **Req to accept UDP queries/responses up to size of smallest MTU:** 9,194 octets
- **Calc from size of eth0 jumbo frame:** 9,216 octets, – 22 octets for header



DNS: RFC 1034: System for naming computers/network services in hierarchical structure for org objects into domains
RFC 3596: DNS extensions for IPv6: Apps using current implementation of DNS expect addr queries to return 32-bit addr

- IPv6: 128-bit addr space: Out of scope for DNS client using IPv4

<u>Required extensions for IPv6 are:</u>

- RR type that will map domain/computer name to IPv6 addr
- Domains defined to support lookups based on addr
- Queries under current DNS sys conducting addl processing for IPv4/6 addr

Extensions don’t create new DNS version: Designed to work w/existing apps w/continued support
Paul Mockapetris: Original RFC’s for DNS (882/883): Built first reference implementation named JEEVES

Kevin Dunlap: Another implementation of DNS: **BIND: Berkeley Internet Name Domain**: BSD UNIX ver 4.3

- BIND most popular in use: Avail for most ver of UNIX/Win Server 2012

DNS DB Structure: DNS DB mirrors structure the domain namespace itself: Tree structure: root: Top figure

- All domains meet at root ID's by period [.]
- Beneath root: Top-level/Primary domains

US: Take form of 3 letter codes

.com	**.edu**	**.gov**	**.mil**	**.net**	**.org**

Better expressed as **.com.** Or **.org.**

Entire domain space for Internet fits beneath root: 13 root servers (A.ROOT/B.ROOT-SERVERS.NET etc..)

- Act as top DNS hierarchy worldwide: Provide source for all name lookups that can't be resolved through other means

FQDN: Fully Qualified Domain Name: All elements of a domain name: Followed by a period/final period for root of DNS

DNS DB Records: Data: Domain names/addr records/stored in **RR: Resource Records:** RR's divided into 4 classes

Taxonomy of record types:

A	**Address:** Stores domain name to IP addr translation data
CNAME	**Canonical Name:** Used to create aliases
HINFO	**Host Information:** Stores descriptive info about
MX	**Mail Exchange:** Route SMTP based email on web: ID IP for email server
NS	**Name Server:** ID all DNS servers in a domain
PTR	**Pointer:** Stores IP addr-to-domain name translation data/supports reverse DNS lookup
SOA	**Start of Authority:** ID's NS that is authoritative for specific DNS DB segment: • ID's master DNS server for a specific domain/subdomain
SRV	**Service Location:** Provides info about avail services/used in AD to map name of service to server • AD clients/DC's use SRV records to determine IP addr for other DC's
TXT	**Text:** Add arbitrary txt info to DNS DB, usually for doc
WKS	**Well-Known Services:** Lists IP-based services (Telnet/FTP/HTTP) that an Internet host can supply

Delegating DNS Authority: Translates into assignment of authority for subdomains to diff domain NS's

- Usually at various locations w/in org's overall scope/geo layout

Types of DNS Servers:

Primary	Where primary DNS files for domains/subdomains for which server is authoritative reside • ASCII snapshot of DNS DB server loads into mem while it runs • **Zone: AKA: Zone file:** DB segment **<u>Primary master:</u>** • Distinguished from other NS's by ability to always read data from zone file on disk • Designation impt config item when setting up DNS server • Can only be 1 primary master NS
Secondary	<u>Gets data for zone from master server for that zone:</u> • Can read data from local file • Always checks to see if its on-disk version is as current as version on primary server • Does so by checking specific field in its SOA record/compares it to master server's DB **Zone Transfer:** 2ndary can update its DB from primary domain NS • Always originates from primary server **Incremental Zone Transfer:** • DNS implementations that limit updates only to data changed on primary server • DNS zone should have at least 1 2ndary master NS/primary NS **Secondary/Slaves impt:** • Provide backup copy of domain DB for a specific zone • Can continue to handle name service reqs for zones if primary NS goes out of service • Can help distribute load for DNS lookups • Permits hosts in 1 zone to gain access to DNS data from other zones

Caching	Store recently accessed DNS records from other domains to avoid perf overhead • Primary/2ndary DNS servers can provide caching functions • Possible to setup/config separate caching-only servers w/in specific domain • Speeds access to specific domain names by storing copy of data locally • Size/Internet access volume factors that determine if org implements separate servers

How Domain Name Servers Work:

Resolver	Client sends a name query to a DNS server: Obtains addr for server it queries from TCP/IP config • Servers queried in order they appear in config file from top/down

Sequence of lookups that produces some kind of reply from whichever domain NS is queried:

1. DNS servers retrieve name data from general domain namespace
2. If given DNS server authoritative: Provides data about zones for which it is authoritative
3. When queried: Any DNS server will search cached domain name data/answer queries
   - For non-authoritative servers: Unless query originates from root server
4. When local server doesn't have info avail in DB/name cache: May turn to caching-only or 'neighborhood'
   - **Neighborhood:** Refers to collection of domains for which any given server may be primary/2ndary master NS
5. If no searches give result: NS sends req for name resolution to a root server: Directs query to authoritative server
   - Root server locates authoritative server by contacting it for the domain/following NS pointers

Recursive Query	Client side: Delegate to 1st DNS server they contact task of going out/finding translation • Keeps telling DNS servers to ask closest known NS's to resolve a domain name
Iterative Query	Target specific DNS server/terminate w/w.e response may be forthcoming • Don't cause other queries to be issues

SOA: Start of Authority Record: 1st entry in any DNS file: Both domain.dns/addr.in-addr.arpa.dns

- ID's current NS/another NS in same domain/subdomain as best source of info for data in its zone
- Both 2ndary/primary NSs can designate themselves as authoritative in SOA records
- What allows load-balancing across primary/1/more 2dary DNS servers in domain

Serial	Unsigned 32-bit num: Original value: What 2ndary NSs use to compare w/primary to see if updated records needed • Incremented each time value updated on primary NS
Refresh	Num of sec that can elapse until zone DB needs to be refreshed • Guarantees no 2dary servers will ever be more than 3 hrs out of sync w/primary • Specifies interval 2dary checks w/primary to learn if any changes have been made to zone definitions/info
Retry	Num of sec that should be allowed to relapse before failed refresh attempted again
Expire	Num of sec that should be allowed to elapse before zone DB no longer authoritative • Reflects value of a counter that allows server to calc how long it has been since last update • 2dary DNS servers discard zone data if refresh can't be accomplished w/in interval
Min TTL	How long any RR should be allowed to persist in another nonauthoritative server's cache • How long a cached entry can persist on servers outside zone where the record originates • Adjusting will affect how long it takes servers to update caches