Post 11

Thursday, January 24, 2019 11:26 PM

TROUBLESHOOTING NETWORKS

Doc includes: Config files: Network config files/end-sys config files

- Phys/logical topology diagrams
- Baseline performance lvl

Info kept single location: Hard copy/protected server: Backups maintained separate loc

Network Config Files: Accurate, up-to-date records of HW/SW used

- A table should exist for each device used w/relevant info about device

Info captured in device table	○ Type of device, model designation ○ IOS img name ○ Device network hostname ○ Loc of device [building/floor/room/rack/panel]
Mobile	Modular: Include all module types/which module slot loc ○ DLL addresses ○ Network layer addresses

End-sys Config Files: Focus on HW/SW used in end-sys devices [servers/mgmt cons/workstations]

- Incorrectly config end sys: Bad impact on performance of network
- Baseline HW/SW on devices/recorded in end-sys doc: Useful when troubleshooting

Following info could be doc w/in end-sys config table:

- Device name (purpose) | OS/ver | IPv4/6 addr | Subnet mask/prefix length
- Default GW/DNS server/WINS server addresses
- Any high-BW network apps end sys runs

Network Topology Diagrams: Keep track of loc/function/status of devices: 2 types: Physical/logical

Physical Topology	Shows phys layout of devices connected to network Info typically includes: • Device type • Model/manufacturer • OS ver • Cable type/identifier • Cable specification • Connector type • Cabling endpoints
Logical Topology	How devices logically connect to a network: How they xfer data: May not represent phys loc Info recorded may include: • Device identifiers • IP/prefix lengths • Int identifiers • Connection type • DLCI for virtual circuits • Site-to-site VPNs • Routing protocols • Static routes • Data-link protocols • WAN technologies used

Baseline Performance Level: Purpose of monitoring: Performance comparison to predetermined baseline

Baseline: Used to establish normal network/sys performance

- Requires collecting performance data from ports/devices essential to operation
- Measuring initial performance/avail critical/links allows admins to determine the differences
- Insight into current design/if can meet business reqs

- W/out baseline: No standard exists to measure optimum nature of traffic/congestion levels

Analysis after baseline: Tends to reveal hidden problems

- Collected data shows nature of congestion/potential congestion

Plan 1st baseline:

- **Determine what types of data to collect**
    - Select vars that represent defined policies
    - If too many data points selected: Amt of data can be overwhelming
    - Int/CPU utilization good start
- **ID devices/ports of interest**
    - Use topology to ID devices/ports where performance data should be measured
    - Devices/ports of interest include:
        - Device ports that connect to other devices
        - Servers
        - Key usrs
        - Anything considered critical to ops
- **Determine baseline duration**
    - Length of time/baseline info being gathered must be sufficient
    - Daily trends of traffic should be monitored
    - Monitor for trends that occur over longer period of time: Weekly/monthly

Measuring Data

Display up/down status/IP of ints:

R1# sh ip int br | sh ipv6 int br

Display r-table: Learn directly connected neighbors/devices/r-protocols

R1# sh ip route | R1# sh ipv6 route

Obtain info about directly connected Cisco neighbor devices

R1# sh cdp neighbor detail

Manual data collection using **sh** cmds on individual devices is extremely time consuming/not scalable

- Manual collection of data should be reserved for smaller networks/mission-critical devices.

SuperAgent: Module enables admins to autoy create/review reports using Intelligent Baselines feature

- Compares current performance lvls w/historical observation
- Can auto ID performance problems/apps that don't provide expected lvls of service

3 stages to troubleshooting process:

Stage 1	**Gather symptoms**	Gathering/doc symptoms from network/end systems/usrs • Determine which components affected/how func changed compared to baseline • Symptoms may appear in many diff forms • Impt to ask questions/investigate issue to localize problem Example: Is problem restricted to single/group of devices/entire subnet?
Stage 2	**Isolate problem**	Isolate process of eliminating vars until single/set of problems ID'd as cause • Examine chars of problems at logical layers of network • May gather/doc more symptoms, depending on chars identified
Stage 3	**Corrective action**	Work to correct it: Implementing/testing/doc possible solutions • After finding problem/determining solution: • May decide if solution can be implemented immediately or postponed • Depends on impact of changes on usrs/network • Severity of problem should be weighed against impact of solution

Gathering Symptoms

Cmd	**Description**
ping ip/host	Send echo req packet to addr/wait for reply
traceroute dest	ID path packet takes through networks: Dest var is IP of target sys
telnet IP	Connect to IP

sh ip int br | sh ipv6 int br	Summary of status of ints
sh ip route | sh ipv6 route	Display current IPv4/6 routing tables: Routes to all known dest
show running-config	Display contents of run config file
[no] debug ?	Options for enabling/disabling debugging events
show protocols	Display config protocols/show global/int-specific status of any L3 protocol

5 steps to gathering info:

1. Gather info from trouble ticket/usrs/end sys affected by problem to form definition of problem
2. Determine ownership: If problem w/in org: Move to next stage
    - If outside: Contact admin for external sys before gathering addl symptoms
3. Narrow scope: Determine if problem at core/distribution/access
4. Gather symptoms from suspect devices
5. Doc symptoms: Sometimes problem can be solved

IOS cmds/tools: ping/traceroute/telnet/show/debug/packet captures/device logs

Questioning End Users

Guidelines	**Example End-usr Questions**
Ask questions pertinent to problem	What doesn't work?
Questions as means to eliminate/discover	Are things that work/things that don't related?
Speak at lvl usr can understand	Has the thing that doesn't work ever worked?
Ask usr when problem 1st noticed	When problem 1st noticed?
Did anything unusual happen since last it worked?	What changed since last time it worked?
Ask usr to recreate problem	Can you reproduce the problem?
Determine seq of events that took place before	When exactly does problem occur?

Using Layered Models for Troubleshooting

OSI Reference: Common lang for admins/commonly used in troubleshooting networks

- Describes how info from SW app in 1 machine moves through medium to SW app on another

L5-7: Upper layers	Deal w/app issues/generally implemented in SW • App layer closest to end usr • Both usrs/app layer processes interact w/SW apps contain a comm component
L1-4: Lower layers	**Data-transport issues:** • L3/4: Generally implemented only in SW • L1/2: Phys/DLL: Implemented in HW/SW • Phys closest to phys medium [cabling]: Responsible for actually placing info on medium

TCP/IP Model

Application	Combines func of 3OSI layers: Session/Presentation/App • App provides comm bet apps: FTP/HTTP/SMTP on separate hosts
Transport	Directly correspond in function: Responsible for exchanging segments bet devices on network
Internet	Relates to OSI network layer: Internet layer responsible for placing msgs in fixed fmt: • Allows devices to handle them
Network access	Corresponds to Phys/DLL: • Comms directly w/network media/provides an int bet arc of network/Internet layer

Troubleshooting Methods

Using layered models: 3 primary methods for troubleshooting:

- **Bottom-up**
- Top-down
- Divide-and-conquer

Bottom-Up	Start w/phys components of network/move up through layers of OSI until cause ID • Good approach to use when problem suspected to be physical

	• Most networking problems reside at lower lvls: Implementing bottom-up approach is often effective **Disadvantage:** Req checking every device/int on network until possible cause of problem found
Top-Down	Starts w/end-usr apps/moves down through layers of OSI model until cause ID'd • End-user apps of end sys tested before tackling more specific networking pieces • Use for simpler problems **Disadvantage:** Req checking every network app until the possible cause found • Each conclusion/possibility must be doc • Challenge to determine which app to start examining 1st
Divide/Conquer	Start by collecting usr experiences: Doc symptoms: Using info: Make informed guess which layer to start w/ • When layer verified to be functioning: Assumed layers below function • Admin can work up layers: If layer not func properly: Admin work down layers

Compare working/non-working situation: Spotting sign diff: Configs/SW ver/HW/device properties

Substitution: Another quick troubleshooting methodology: Swapping problematic device w/known, working one

- If problem fixed: Admin knows problem is w/device

Guidelines for Selecting Troubleshooting Method

SW Troubleshooting Tools

NMS: Network Mgmt Sys Tools: Device-lvl monitoring/config/fault-mgmt tools

- "WhatsUp Gold" NMS software: Tools can be used to investigate/correct network problems

Network monitoring SW: Graphically displays view of devices/monitor w/out physically checking them

- Provides dynamic status/stats/config info for switched products

Examples: CiscoView/HPBTO SW/SolarWinds

Knowledge Bases

Baselining Tools: SolarWinds LANsurveyor/CyberGauge

- Help w/common doc tasks: Can draw diagrams/keep network SW/HW doc up-to-date/help cost

Host-Based Protocol Analyzers

IOS Embedded Packet Capture: Troubleshooting/tracing tool: Capture IPv4/6 packets

HW Troubleshooting

NAM	**Network Analysis Module:** Can be installed in Cisco Catalyst 6500 switches/7600 series rtrs • Provides graphic representation of traffic from local/remote switches/rtrs • NAM: Embedded browser-based int that generates reports on traffic that consumes resources • Can capture/decode packets/track response times to pinpoint app problem to network/server
DMM	**Digital Multi Meter:** Fluke 179: Test instruments used to measure voltage/current/resistance • Most multimedia tests involve checking PS voltage lvls/verifying devices receiving power
Cable Testers	Specialized, handheld devices for testing various types of data comm cabling • Detect broken wires/crossed-over wiring/shorted connections/improperly paired connections **Devices can be:** • Continuity tester $ • Data cabling tester $$ • **TDR: Time-Domain Reflectometers** $$$ **TDRs: Used to pinpoint distance to a break in cable** • Send sigs along cable/wait for them to be reflected • Time bet sending/receiving is converted into distance measurement • Used to test fiber cables known as **OTDRs: Optical Time-Domain Reflectometers**
Cable Analyzers	Multifunctional handheld devices used to test/certify copper/fiber cables for diff services/standards • Typically include PC-based SW: After field data collected: Device can UL data for up-to-date reports
Portable Analyzer	Troubleshooting switched networks/VLANs

Syslog Troubleshooting

Con	Logging on by default: Msgs log to con/can be viewed when mod/testing rtr/switch using term
Term lines	Can be config to receive log msgs on any term lines
Buffered logging	Log msgs stored in mem for time: Cleared when device rebooted
SNMP traps	Certain thresholds can be preconfig on rtrs/devices
Syslog	Rtrs/switches can be config to fwd log msgs to external syslog service

```
R1(config)# logging host 209.165.200.225
R1(config)# logging trap notifications
R1(config)# logging on
```

Problems Phys layer:

Perf < Baseline	Slow/poor perf include: Overloaded/underpowered servers | Unsuitable switch/rtr configs • Traffic congestion on low-capacity link/chronic frame loss
Loss connectivity	Cable/device fails: Loss of connectivity bet devices that comm over link/w/failed device/int • Indicated by ping: Loose/oxidized connection
Bottlenecks/congestion	Rtr/int/cable fails: R-protocols may redirect traffic to other routes not designed to carry capacity
High CPU rates	Device op @exceeded limits: If not fixed: Overloading can cause shut down/fail
Con error msgs	Indicate phys layer problem

Network problems @Phy phys layer:

Power-related	Check op of fans/ensure chassis intake/exhaust vents clear • If nearby units also down: Suspect power failure at main PS
HW faults	Faulty NICs: Cause of transmission errors: Late collisions/short frames/jabber **Jabber:** Network device continually transmits random data onto network • Faulty/corrupt NIC drivers/bad cabling/grounding problems
Cabling faults	Look for damaged/improper types/poorly crimped RJ-45s
Attenuation	Cable length exceeds limit for media: Poor connection/dirty/oxidized contacts • If severe: Receiving device can't distinguish component bits of stream from each other
Noise	**EMI: AKA Noise:** Many sources: FM stations/police radio/avionics • **Crosstalk:** Noise induced by other cables in same pathway/adjacent cables ○ Nearby electric cables/devices w/large motors/anything that includes transmitter
Int config errors	
Exceed design limits	Component may be op sub-optimally at phys layer b/c being utilized at higher avg rate
CPU overload	Processes w/high CPU util %'s: Input queue drops/slow/rtr services [Telnet/ping]: High traffic

DLL Troubleshooting

No func @L2+	Some L2 problems can stop exchange of frames across link: Others only cause performance to degrade
Network op below baseline	**2 Distinct types:** 1. Frames take suboptimal path to dest but arrive: High-BW usage on links 2. Some frames drop: ID error counter stats/con error msgs on switch/rtr: eth: Continuous ping
Excessive broadcasts	Poorly coded/config apps: Large L2 broadcast domains: STP loops/route flapping
Con msgs	Rtr detects problem w/interpreting inc frames (encapsulation/ framing problems) • Keepalives expected don't arrive: Line protocol down msg

Issues at DLL Connectivity/Performance problems:

Encapsulation errors	Bits placed in particular field by sender not what receiver expects • Encapsulation at 1 end of WAN link config diff from encapsulation used at other
Addr mapping errors	Topologies: Point-to-multipoint/Frame Relay/Broadcast Ethernet: Appropriate L2 dest be given to frame • Device must match dest L3 addr w/correct L2 addr using static/dynamic maps • Dynamic: Mapping L2/L3 info can fail b/c devices may have been config not to respond to ARP/I-ARP • L2/L3 info cached may have phys changed • Invalid ARP replies received b/c misconfig/sec attack
Framing errors	Usually work in groups of 8-bit bytes: Error when frame doesn't end on 8-bit byte boundary • Receiver may have problems determining where 1 frame ends/another starts • Too many invalid frames: Prevent valid keepalives from being exchanged • Framing errors caused by noisy serial line/improperly designed cable • Incorrectly config CSU: Chan Service Unit line clock
STP failures/loops	STP purpose: Resolve redundant phys topology into tree-like topology by blocking redundant ports • Most STP problems: Related to fwding loops that occur when no ports in redundant topology blocked • Traffic fwded in circles indefinitely: Flooding • Mismatch bet real/doc topology/config error/overloaded switch CPU/SW defect

Network Layer Troubleshooting
Common symptoms:

Network failure	When network nearly/non-functional
Subop perf	Optimization problems usually involve subset of usrs/apps/dest/particular type of traffic • Multiple layers/host computer itself: Can take time

Troubleshooting:

Gen issues	Often change in topology: Down link: Install new routes: Static/dynamic/Removal of routes
Connectivity	Check for equip/connectivity problems: Power problems: Outages/env problems: L1 problems
Neighbors	R-protocol establishes adj w/neighbor: Check if problems w/rtrs forming neighbor adj
Topology DB	R-protocol uses topology table/db: Check table for unexpected
R-Table	Missing/unexpected routes: debug cmds to view r-updates/table maintenance

Transport Troubleshooting – ACLs
Areas where misconfigs commonly occur:

Selection of traffic flow	Most common: Applying ACL to incorrect traffic: Defined by both rtr int/direction traffic traveling • ACL must be applied to correct int/correct traffic direction to function properly
Order of access control	Entries in ACL should be specific to general: • ACL may have entry to specifically permit particular traffic flow: • Packets never match entry if being denied by another entry earlier in list • Inbound traffic processed by inbound ACL before outside-to-inside NAT • Outbound traffic processed by outbound ACL after being processed by inside-to-outside NAT
Implicit deny all	Can be cause of ACL misconfig
Addr/IPv4 wildcards	Common sources of misconfigs
Selection of transport protocol	Impt that only correct transport layer protocols specified: • Specifying both TCP/UDP opens hole through FW • Introduces extra element into ACL: Longer to process: More latency
Src/dest ports	Properly controlling traffic bet 2 hosts req symmetric access control elements for inbound/outbound ACLs • Addr/port info for traffic generated by replying host
Established keyword	Increases sec for ACL: If applied incorrectly: Bad

Uncommon protocols	Misconfig ACLs often cause problems for protocols other than TCP/UDP

Log keyword useful for viewing ACL op/entries

- Instructs rtr to place entry in sys log whenever condition matched
- Logged event includes details of packet that match ACL element
- Useful for troubleshooting/provides info on intrusion attempts being blocked by ACL

Transport Troubleshooting: NAT IPv4

BOOTP/DHCP	Both protocols manage auto assignment of IPv4 addr • 1st packet new client sends is DHCP-Req broadcast IPv4 • DHCP-Req packet has src IPv4 addr 0.0.0.0 • NAT req both valid dest/src IPv4 • BOOTP/DHCP can have diff op over rtr running either static/dynamic NAT • Config IPv4 helper feature can solve
DNS/WINS	Rtr running dynamic NAT change relationship bet inside/outside addr as entries expire/recreated • DNS/WINS server outside NAT rtr doesn’t have accurate representation of network inside rtr • Config IPv4 helper feature can solve
SNMP	Similar to DNS packets: NAT unable to alter addr info stored in data payload of packet • SNMP mgmt station on 1 side of NAT rtr may not be able to contact SNMP agents on other side of NAT rtr • Config IPv4 helper feature can solve
Tunneling/encryption protocols	Often req traffic be src from specific UDP/TCP port/protocol at transport can’t be processed by NAT • Example: IPsec tunneling protocols/GRE used by VPN can’t be processed by NAT

App Layer Troubleshooting
Known TCP/IP app protocols:

SSH/Telnet	Establish term session connections
HTTP	Exchanging of txt/imgs/sound/video/files on web
FTP	File xfers
TFTP	File xfers bet hosts/networking devices
SMTP	Basic msg delivery services
POP	Mail servers/DL email
SNMP	**Simple Network Mgmt Protocol:** Collects mgmt info from network devices
DNS	Maps IP to names assigned
NFS	**Network File Sys:** Enables computers to mnt drives on remote hosts/op as if local drives • Dev: Sun: Combines w/2 other app layer protocols • XDR: External Data Representation | RPC: Remote-Procedure Call • Allow transparent access to remote network resources.

Troubleshooting End-to-End Connectivity

1. Check phys connectivity at point where network comm stops: Cables/HW
2. Check for duplex mismatches
3. Check DLL/network layer addr: IPv4 ARP tables/IPv6 neighbor tables/MAC addr tables/VLAN assignments
4. Verify default GW correct
5. Ensure devices determining correct path from src to dest: Manip r-info if necessary
6. Verify transport func properly: Telnet can be used to test connections
7. Verify no ACLs blocking traffic
8. Ensure DNS settings correct: DNS server accessible

Verify problem w/end-to-end: ping | traceroute

Ping	Sends reqs for responses from specified host addr: Uses L3 protocol part of TCP/IP: ICMP: Echo req/reply packets

Traceroute	Path IPv4 packets take to reach dest: List of hops/rtr IP/final dest IP successfully reached along path • If data reaches dest: Trace lists int on every rtr in path • If data fails at hop along way: Addr of last rtr known

Output of sh int cmd lists # impt stats:

Input queue drops	At some point more traffic delivered to rtr than it could process: Not necessarily problem • Could indicate CPU can't process packets in time if consistently high
Output queue drops	Packets dropped due to congestion on int: Peaks: Packets dropped if delivered to int faster than sent • Leads to packet drops/queuing delays • VoIP, might suffer: Indicator advanced queuing mech needed for QoS
Input errors	Indicate errors exp during reception of frame: CRC errors • High CRC errors could indicate cabling/int HW/eth-based network problems/duplex mismatches
Output errors	Errors like collisions during transmission of frame: Full-duplex norm/half exception • Full-duplex: Op collisions can't occur

Check for Duplex Mismatches

- P-t-P eth links should always be run in full-duplex
- Half-duplex not common
- Autonegotiation of speed/duplex recommended
- If autonegotiation doesn't work: Manually set speed/duplex on both ends
- Half-duplex on both ends performs better than duplex mismatch

Verify L2/L3 Addr on Local Network

IPv4 ARP Table: arp cmd displays/mod entries in ARP cache used to store IPv4 addr/resolved MAC'

- Cache can be cleared by **arp -d** to repopulate cache w/updated inf

IPv6 Neighbor Table: netsh int ipv6 show neighbor Lists all devices in neighbor table

- Info displayed for each device includes IPv6 addr/MAC/type of addr
- Linux/MAC OS X: **ip neigh show**

Switch MAC Addr Table: Switch fwds frame only to port where destination connected

- To do this: Consults MAC addr table
- **sh mac address-table** Display MAC addr table

VLAN Assignment: In switched network, each port belongs to VLAN

- Each VLAN considered separate logical network
- Packets destined for stations that don't belong to VLAN must be fwded through device that supports routing
- If host in 1 VLAN sends broadcast Eth frame [arp req]: All hosts in same VLAN receive frame
- Hosts in other VLANs don't
- **sh vlan** Validate vlan assignments

Troubleshooting Network Layer

sh ip route Examine IPv4 r-table

IPv4/6 r-tables populated by:

- Directly connected networks
- Local host/routes
- Static/Dynamic routes
- Default routes