IT Scripting and Automation

Network Management on Linux Systems

Lecturer: Art Ó Coileáin

Network Management

- Network management is the discipline of keeping a network healthy.
- It generally includes the following tasks:
  - Fault detection for network, gateways, and critical servers
  - Schemes for notifying and administrator of problems
  - General network monitoring, to balance load and plan expansion
  - Documentation and visualisation of the network
  - Administration of network devices from central site
- As networks grows, management procedures should be more automated.

Network Troubleshooting

- On networks consisting of several different subnets joined with switches or routers, it is recommended to start automating management tasks with shell scripts and simple programs.

Consider these Principles:

1. Make one **change** at a time. Test each change to make sure that it had the effect you intended. Back out any changes that have an undesired effect.
2. **Document** the situation as it was before you got involved, and document every change you make along the way.
3. Problems may be transient, so begin by capturing relevant **information** with tools like sar and **nmon**. This information may come in handy as you are unravelling the problem.

Network Troubleshooting

4. **Methodical**: Start at one end of a system or network and work through the system's critical components until your each the problem.

   For example, you might start by looking at the network configuration on a client, work your way up to the physical connections, investigate the network hardware, and finally, check the server's physical connections and software configuration.

5. **Communicate** regularly. Most network problems affect lots of different people: users, ISPs, system administrators, telco-engineers, network administrators, etc.

   Clear, consistent communication prevents you from hindering one another's efforts to solve the problem.

Network Troubleshooting

7. **Work as a team**.

Years of experience show that people make fewer stupid mistakes if they have a peer helping out.

If the problem has any visibility, management will also want to be involved. Take advantage of managers' interest to get technical people from other groups on board and to cut through red tape where necessary.

8. Use the **layers of the network** to negotiate the problem.

Start at the "top" or "bottom" and work your way through the protocol stack.

Network Troubleshooting

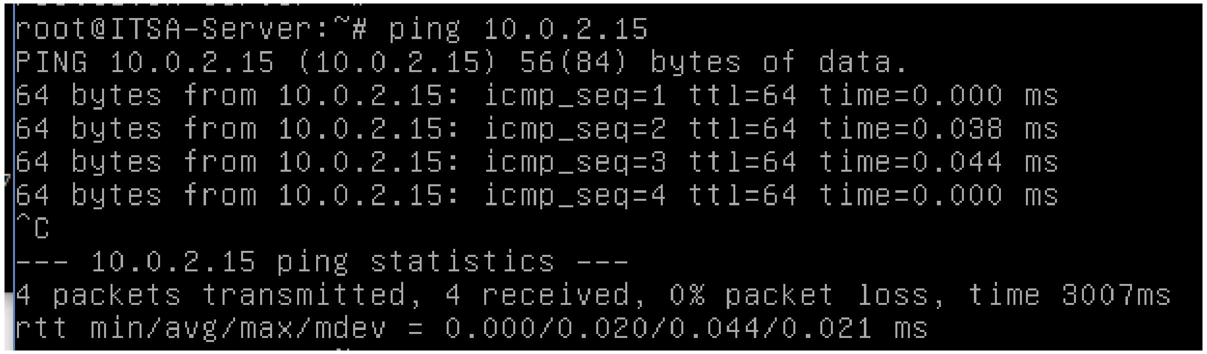
8. Ask yourself questions like these as you work up or down the stack:
    - Do you have physical connectivity and a link light?
    - Is your interface configured properly?
    - Do your ARP tables show other hosts?
    - Is there a firewall on your local machine?
    - Is there a firewall anywhere between you and the destination?
    - If firewalls are involved, do they pass ICMP ping packets and responses?
    - Can you ping the localhost address (127.0.0.1)?
    - Can you ping other local hosts by IP address?
    - Is DNS working properly?
    - Can you ping other local hosts by hostname?
    - Can you ping hosts on another network?
    - Do high-level services such as web and SSH servers work?
    - Did you really check the firewalls?

PING: Check if a host is alive

- The **ping** command sends an ICMP ECHO_REQUEST packet to a target host and waits to see if the host answers back.
- Ping is used to check status of individual hosts and to test segments of networks.
- Be aware of networks that block ICMP echo request with a firewall. You might consider disabling a meddlesome firewall for a short period of time to facilitate debugging.

Usage:

$ ping *ipaddress/hostname*



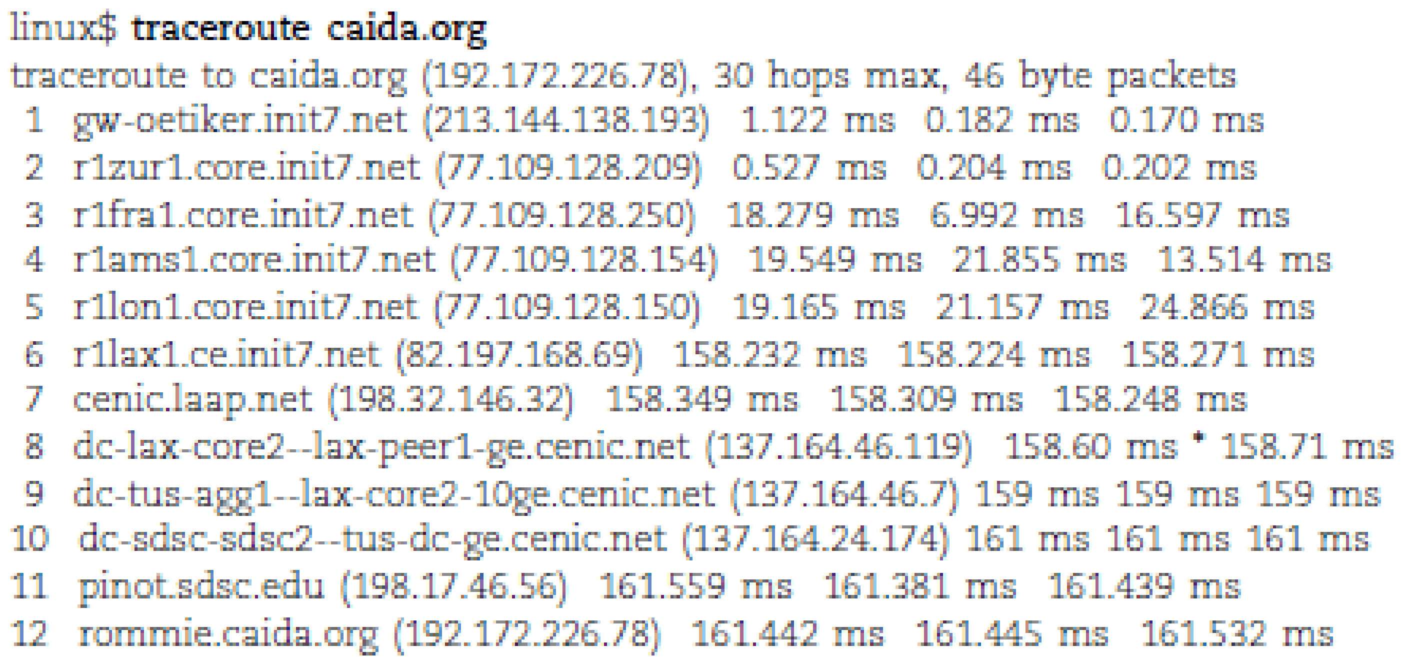
```
root@ITSA-Server:~# ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.000 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.044 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.000 ms
^C
--- 10.0.2.15 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.000/0.020/0.044/0.021 ms
```

Traceroute: Trace IP Packets

- traceroute command uncover the sequence of gateways though which an IP packet travels to reach its destination.

Usage:

- **traceroute *hostname/ipaddress***



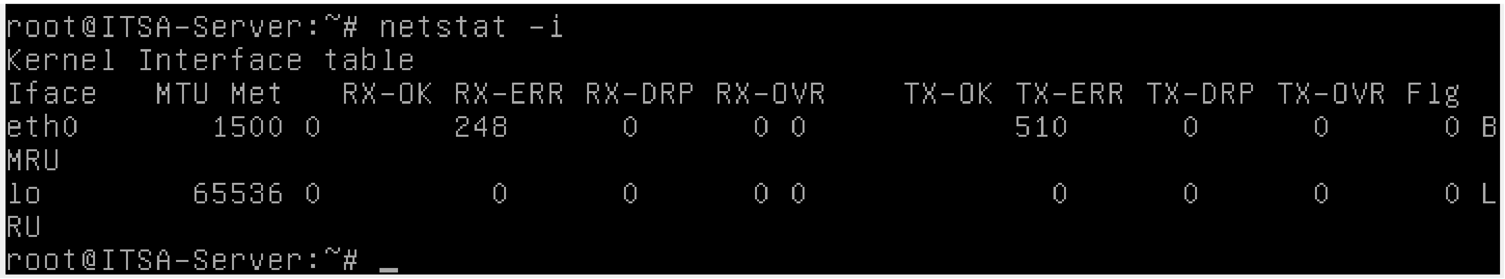
```
linux$ traceroute caida.org
traceroute to caida.org (192.172.226.78), 30 hops max, 46 byte packets
 1  gw-oetiker.init7.net (213.144.138.193)  1.122 ms  0.182 ms  0.170 ms
 2  r1zur1.core.init7.net (77.109.128.209)  0.527 ms  0.204 ms  0.202 ms
 3  r1fra1.core.init7.net (77.109.128.250)  18.279 ms  6.992 ms  16.597 ms
 4  r1ams1.core.init7.net (77.109.128.154)  19.549 ms  21.855 ms  13.514 ms
 5  r1lon1.core.init7.net (77.109.128.150)  19.165 ms  21.157 ms  24.866 ms
 6  r1lax1.ce.init7.net (82.197.168.69)  158.232 ms  158.224 ms  158.271 ms
 7  cenic.laap.net (198.32.146.32)  158.349 ms  158.309 ms  158.248 ms
 8  dc-lax-core2--lax-peer1-ge.cenic.net (137.164.46.119)  158.60 ms * 158.71 ms
 9  dc-tus-agg1--lax-core2-10ge.cenic.net (137.164.46.7) 159 ms 159 ms 159 ms
10  dc-sdsc-sdsc2--tus-dc-ge.cenic.net (137.164.24.174) 161 ms 161 ms 161 ms
11  pinot.sdsc.edu (198.17.46.56)  161.559 ms  161.381 ms  161.439 ms
12  rommie.caida.org (192.172.226.78)  161.442 ms  161.445 ms  161.532 ms
```

NetStat: Get network stats

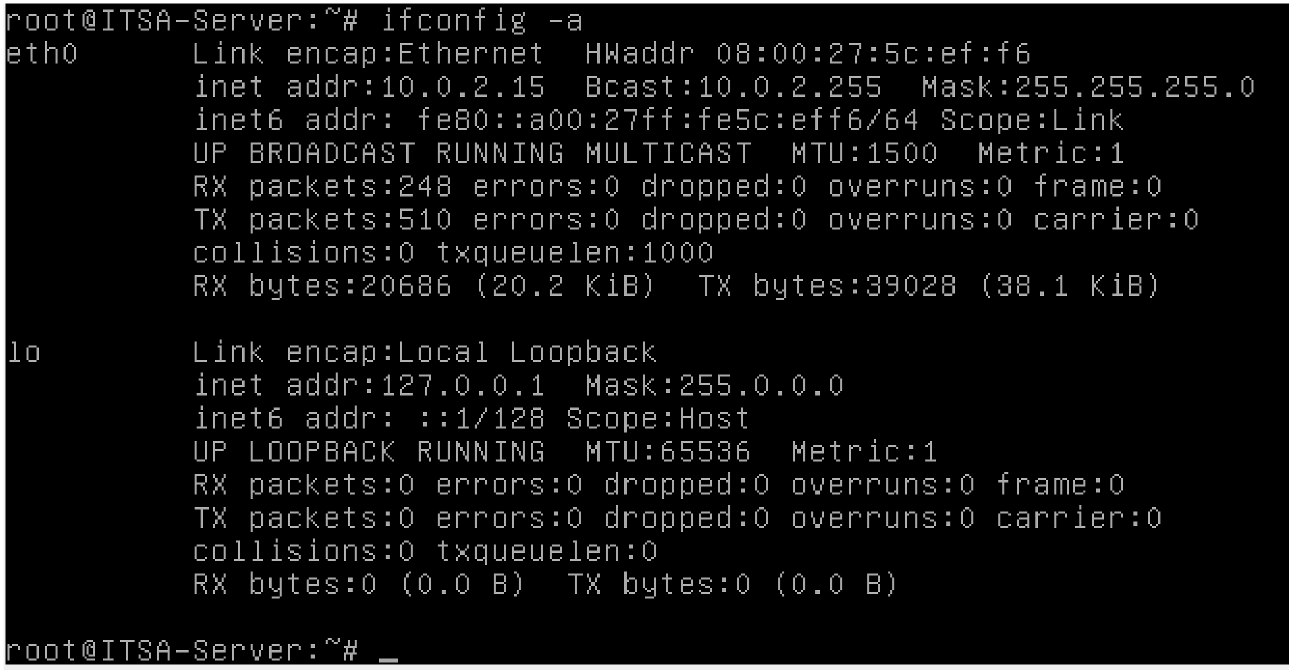
- **netstat** command collects a wealth of information about the state of your computer’s networking software, including interface statistics, routing information, and connection tables.
- It exposes a variety of network information that doesn’t fit anywhere else. The five most common uses are:
  - Inspecting interface configuration information
  - Monitoring the status of network connections
  - Identifying listening network services
  - Examining the routing table
  - Viewing operational statistics for various network protocols
- The command: **$ netstat -i**
- It shows the configuration and state of each of the host’s network interfaces along with the associates traffic counters

NetStat: Get network stats

- On Linux, **ifconfig -a** can be used instead of **netstat –i** to show the network interfaces.



```
root@ITSA-Server:~# netstat -i
Kernel Interface table
Iface    MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0       248      0      0 0           510      0      0      0 B
MRU
lo        65536 0         0      0      0 0             0      0      0      0 L
RU
root@ITSA-Server:~# _
```



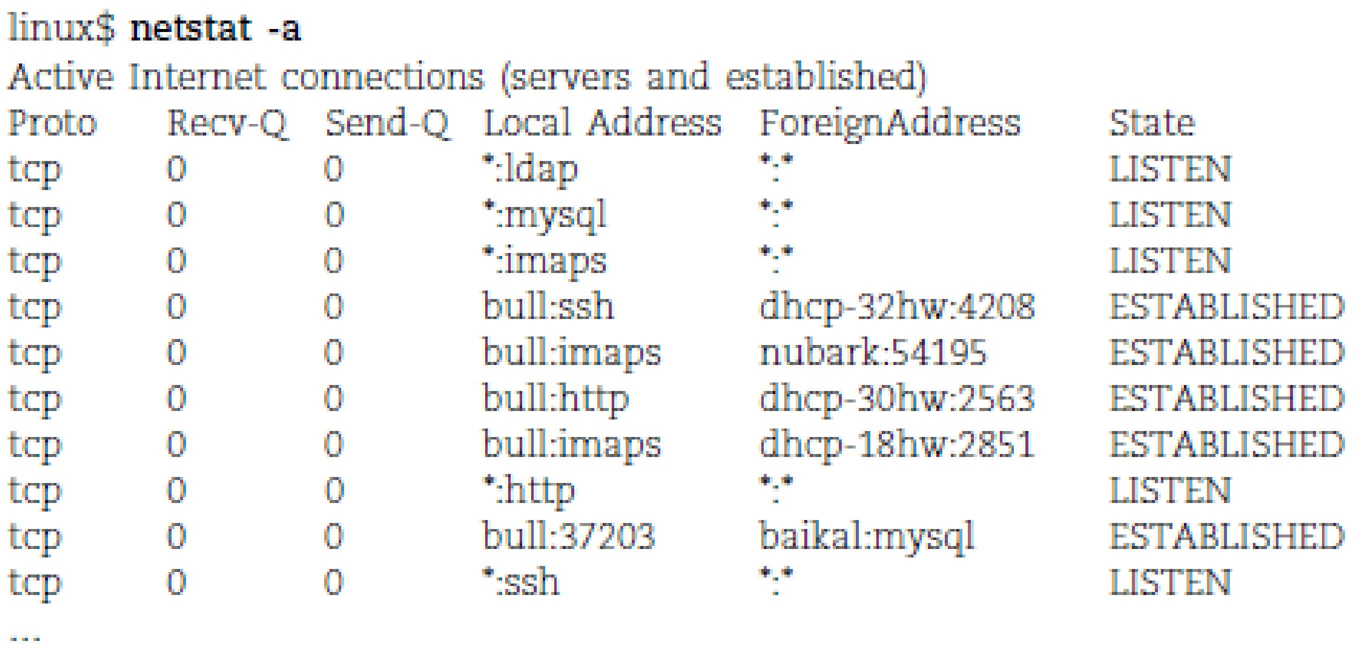
```
root@ITSA-Server:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 08:00:27:5c:ef:f6
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5c:eff6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:248 errors:0 dropped:0 overruns:0 frame:0
          TX packets:510 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20686 (20.2 KiB)  TX bytes:39028 (38.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ITSA-Server:~# _
```

NetStat: Get network stats

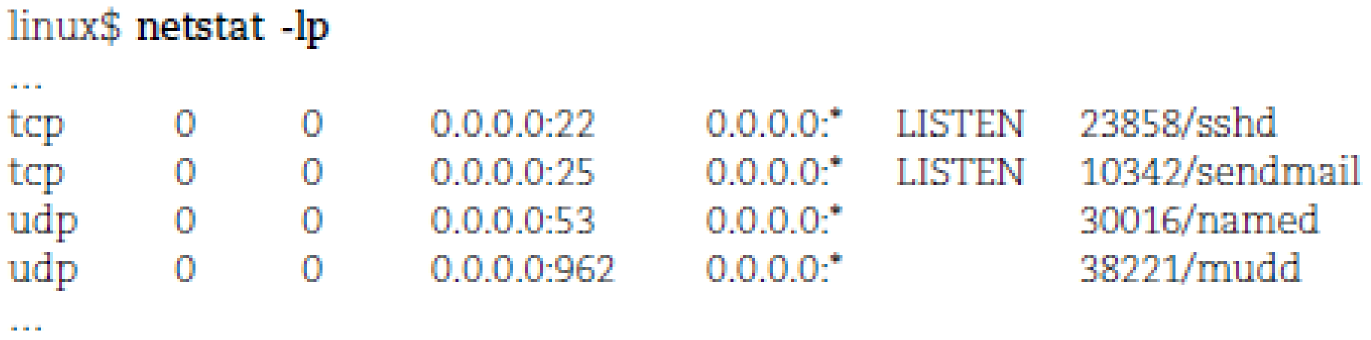
- **netstat -a** displays the status of active TCP and UDP ports.
- In active ("listening") servers that are waiting for connection are normally hidden but this command will show them.
- For well-known services, **netstat** shows the port symbolically, using the mapping defined in **/etc/services**.



```
linux$ netstat -a
Active Internet connections (servers and established)
Proto   Recv-Q   Send-Q   Local Address   ForeignAddress    State
tcp     0        0        *:ldap          *:*               LISTEN
tcp     0        0        *:mysql         *:*               LISTEN
tcp     0        0        *:imaps         *:*               LISTEN
tcp     0        0        bull:ssh        dhcp-32hw:4208    ESTABLISHED
tcp     0        0        bull:imaps      nubark:54195      ESTABLISHED
tcp     0        0        bull:http       dhcp-30hw:2563    ESTABLISHED
tcp     0        0        bull:imaps      dhcp-18hw:2851    ESTABLISHED
tcp     0        0        *:http          *:*               LISTEN
tcp     0        0        bull:37203      baikal:mysql      ESTABLISHED
tcp     0        0        *:ssh           *:*               LISTEN
...
```

NetStat: Get network stats

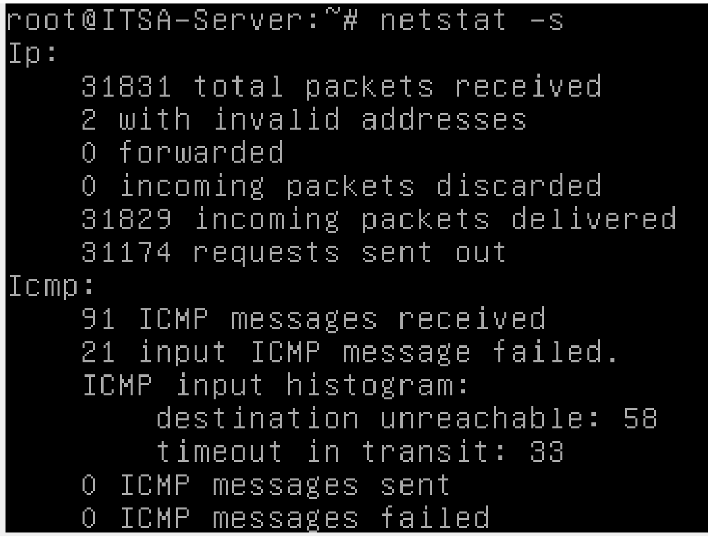
- One common question: What processes on this machine are listening on the network for incoming connections?
- **netstat -a** will show all the ports that are actively listening.
- On Linux, use **netstat -l** to see only the listening ports, add the **-p** flag to identify the specific process associated with each listening port.



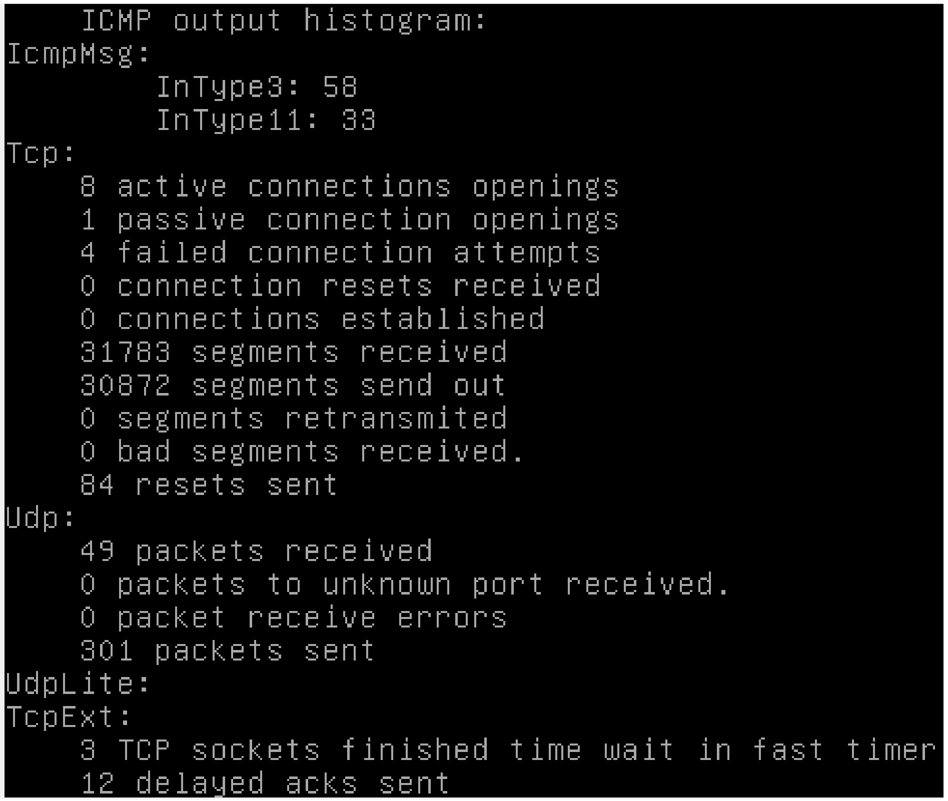
```
linux$ netstat -lp
...
tcp     0    0    0.0.0.0:22     0.0.0.0:*   LISTEN   23858/sshd
tcp     0    0    0.0.0.0:25     0.0.0.0:*   LISTEN   10342/sendmail
udp     0    0    0.0.0.0:53     0.0.0.0:*            30016/named
udp     0    0    0.0.0.0:962    0.0.0.0:*            38221/mudd
...
```

NetStat: Get network stats

- **netstat -r** displays the kernel's routing table.
- **netstat -s** dumps the contents of counter that are scattered throughout the network code. The output has separate section for IP, ICMP, TCP, and UDP.



```
root@ITSA-Server:~# netstat -s
Ip:
    31831 total packets received
    2 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    31829 incoming packets delivered
    31174 requests sent out
Icmp:
    91 ICMP messages received
    21 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 58
        timeout in transit: 33
    0 ICMP messages sent
    0 ICMP messages failed
```



```
    ICMP output histogram:
IcmpMsg:
        InType3: 58
        InType11: 33
Tcp:
    8 active connections openings
    1 passive connection openings
    4 failed connection attempts
    0 connection resets received
    0 connections established
    31783 segments received
    30872 segments send out
    0 segments retransmited
    0 bad segments received.
    84 resets sent
Udp:
    49 packets received
    0 packets to unknown port received.
    0 packet receive errors
    301 packets sent
UdpLite:
TcpExt:
    3 TCP sockets finished time wait in fast timer
    12 delayed acks sent
```

Sar command

- On Linux, the **sar** (System Activity Report) command can Collect, report, or save system activity information.

Example:

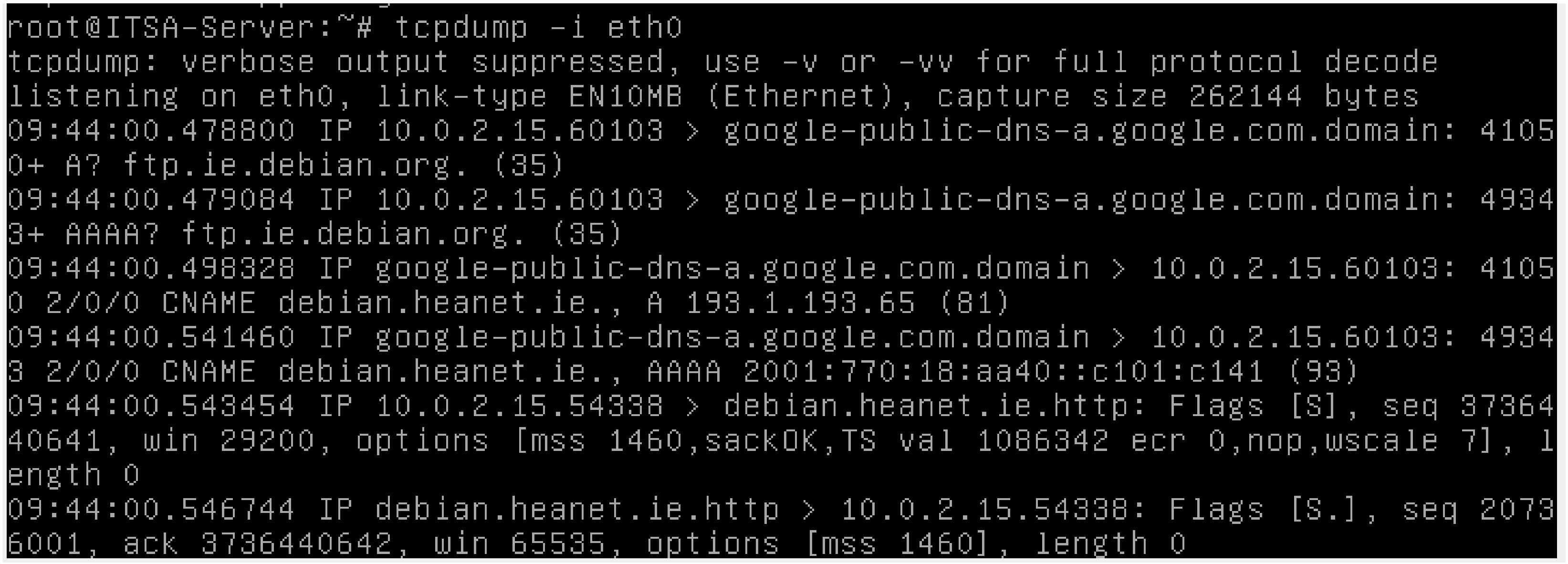
- It can create a report every two seconds for a period of one minute (i.e, 30 reports)

```
$ sar -n DEV 2 30
```

- The DEV argument is a literal keyword.
- The output includes instantaneous and average readings of interface utilisation in units of both bytes and packets.
- The rxbyt/s and txbyt/s columns are probably the most useful since they show the actual bandwidth in use. The final three columns give statistics on compressed (rxcmp/s, txcmp/s) and multicast (rxmcst/s) packets.

Packet Sniffers

- **tcpdump** and Wireshark application (**tshark** console version) belong to a class of tools known as packet sniffers.
- They listen to network traffic and record or print packets that meet criteria of your choice.
- Packet sniffers are useful both for solving problems that you know about and for discovering entirely new problems.



```
root@ITSA-Server:~# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:44:00.478800 IP 10.0.2.15.60103 > google-public-dns-a.google.com.domain: 4105
0+ A? ftp.ie.debian.org. (35)
09:44:00.479084 IP 10.0.2.15.60103 > google-public-dns-a.google.com.domain: 4934
3+ AAAA? ftp.ie.debian.org. (35)
09:44:00.498328 IP google-public-dns-a.google.com.domain > 10.0.2.15.60103: 4105
0 2/0/0 CNAME debian.heanet.ie., A 193.1.193.65 (81)
09:44:00.541460 IP google-public-dns-a.google.com.domain > 10.0.2.15.60103: 4934
3 2/0/0 CNAME debian.heanet.ie., AAAA 2001:770:18:aa40::c101:c141 (93)
09:44:00.543454 IP 10.0.2.15.54338 > debian.heanet.ie.http: Flags [S], seq 37364
40641, win 29200, options [mss 1460,sackOK,TS val 1086342 ecr 0,nop,wscale 7], l
ength 0
09:44:00.546744 IP debian.heanet.ie.http > 10.0.2.15.54338: Flags [S.], seq 2073
6001, ack 3736440642, win 65535, options [mss 1460], length 0
```