

Análisis de Desempeño del Servidor



WALC 2009: Gestión de Redes

Septiembre 21-25, 2009
Bogotá, Colombia

hervey@nsrc.org

Análisis Local

- **Antes de echar la culpa a la red, verificar si los problemas son locales.**
- **Qué puede ir mal localmente?:**
 - Problemas de hardware
 - Carga excesiva (CPU, memoria, I/O)
- **Qué está considerado 'normal'?**
 - Uso frecuente de herramientas de análisis
 - Familiarizarse con los valores y estado de la máquina
 - **Es esencial mantener historia**
 - Agentes de SNMP y bases de datos

Análisis de desempeño en Unix

- Tres categorías principales:
 - Procesamiento
 - Procesos en ejecución (running)
 - Procesos en espera (sleeping)
 - Esperando turno
 - bloqueados
 - Memoria
 - Real
 - Virtual
 - I/O (Input/Output)
 - Almacenamiento
 - Red

Medidas Clave

- **Insuficiente capacidad de procesamiento**
 - Número de procesos en espera de ejecución es constantemente alto
 - Porcentaje de utilización del CPU alto
- **Memoria insuficiente**
 - Poca memoria libre
 - Mucha actividad swap (swap in, swap out)
- **Entrada/Salida (Input/Output) lento**
 - Muchos procesos en estado bloqueado
 - Número alto de bloques (unidades fijas de datos) transferidos

Análisis local

- Afortunadamente, en Unix existen docenas de herramientas útiles (y gratis) que ofrecen mucha información sobre la máquina.
- Algunas de las más conocidas:
 - vmstat
 - top
 - lsof
 - netstat
 - tcpdump
 - wireshark (ethereal)
 - iptraf
 - ntop
 - iperf

vmstat

- Muestra periódicamente información (resumida) sobre procesos, memoria, paginación, I/O, CPU, etc.

```
vmstat <-opciones> <periodo> <máximo de veces>
```

```
# vmstat 2
```

procs		-----memory-----				---swap--		-----io-----		--system--		----cpu----			
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa
2	0	209648	25552	571332	2804876	0	0	3	4	3	3	15	11	73	0
2	0	209648	24680	571332	2804900	0	0	0	444	273	79356	16	16	68	0
1	0	209648	25216	571336	2804904	0	0	6	1234	439	46735	16	10	74	0
1	0	209648	25212	571336	2804904	0	0	0	22	159	100282	17	21	62	0
2	0	209648	25196	571348	2804912	0	0	0	500	270	82455	14	18	68	0
1	0	209648	25192	571348	2804912	0	0	0	272	243	77480	16	15	69	0
2	0	209648	25880	571360	2804916	0	0	0	444	255	83619	16	14	69	0
2	0	209648	25872	571360	2804920	0	0	0	178	220	90521	16	18	66	0

top

- Herramienta básica de análisis de desempeño en un entorno Unix/Linux
- Muestra periódicamente una lista de estadísticas acerca del desempeño del sistema:
 - Uso del CPU
 - Uso de la memoria RAM y SWAP
 - Carga promedio (load average)
 - Información por proceso

Carga Promedio (Load Average)

- Promedio de procesos activos en los últimos 1, 5 y 15 minutos
 - Una medida simplista pero útil
 - Dependiendo de la máquina, los rangos considerados 'normales' pueden variar:
 - Máquinas multi-procesador pueden manejar más procesos activos por unidad de tiempo

top

- **Información por proceso (columnas más relevantes):**

PID: ID del proceso

~ **USER:** usuario que ejecuta el proceso

~ **%CPU:** Porcentaje del tiempo total del CPU utilizado por el proceso desde la última muestra

~ **%MEM:** Porcentaje de la memoria física utilizado por el proceso

~ **TIME:** Tiempo total del CPU utilizado por el proceso desde su inicio

Ejemplo top

```
top - 21:00:57 up 32 days, 15:26, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 134 total, 1 running, 133 sleeping, 0 stopped, 0 zombie
Cpu(s): 2.9%us, 0.2%sy, 0.0%ni, 96.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4149020k total, 3985912k used, 163108k free, 74476k buffers
Swap: 7812492k total, 1708k used, 7810784k free, 3401548k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10749	www-data	20	0	65620	26m	6400	S	10	0.7	0:01.98	apache2
4817	bind	24	4	79908	26m	2116	S	0	0.6	3:15.46	named
1	root	20	0	1812	840	612	S	0	0.0	0:12.10	init
2	root	15	-5	0	0	0	S	0	0.0	0:00.00	kthreadd
3	root	RT	-5	0	0	0	S	0	0.0	0:00.57	migration/0
4	root	15	-5	0	0	0	S	0	0.0	0:01.60	ksoftirqd/0
5	root	RT	-5	0	0	0	S	0	0.0	0:00.01	watchdog/0
6	root	RT	-5	0	0	0	S	0	0.0	0:00.86	migration/1
7	root	15	-5	0	0	0	S	0	0.0	0:02.17	ksoftirqd/1
8	root	RT	-5	0	0	0	S	0	0.0	0:00.01	watchdog/1
9	root	RT	-5	0	0	0	S	0	0.0	0:00.55	migration/2
10	root	15	-5	0	0	0	S	0	0.0	0:01.88	ksoftirqd/2
11	root	RT	-5	0	0	0	S	0	0.0	0:00.01	watchdog/2
12	root	RT	-5	0	0	0	S	0	0.0	0:01.05	migration/3
13	root	15	-5	0	0	0	S	0	0.0	0:02.47	ksoftirqd/3
14	root	RT	-5	0	0	0	S	0	0.0	0:00.00	watchdog/3
15	root	15	-5	0	0	0	S	0	0.0	2:39.97	events/0
16	root	15	-5	0	0	0	S	0	0.0	0:04.59	events/1
17	root	15	-5	0	0	0	S	0	0.0	0:03.39	events/2
18	root	15	-5	0	0	0	S	0	0.0	0:03.15	events/3
19	root	15	-5	0	0	0	S	0	0.0	0:00.00	khelper
56	root	15	-5	0	0	0	S	0	0.0	0:00.39	kblockd/0

top

- **Comandos interactivos más relevantes**

- **f** : Agregar o quitar columnas
- **F** : Especificar la columna a ordenar
- **<** , **>** : Mover la columna a ordenar
- **u** : Especificar un usuario en particular
- **k** : Especificar un proceso a matar (kill)
- **d** , **s** : Cambiar el intervalo de ejecución

netstat

- **Muestra información sobre:**
 - Conexiones de red
 - Tablas de encaminamiento
 - Estadísticas de interfaz
 - Membresías de grupos multicast

netstat

Parámetros más relevantes

- n**: Mostrar direcciones, puertos y usuarios en forma numérica
- r**: Tabla de rutas
- s**: Estadísticas por protocolo
- i**: Estado de las interfaces
- l**: Puertos abiertos (listening sockets)
- tcp, --udp**: Especificar el protocolo
- A**: Familia de direcciones [inet | inet6 | unix | etc.]
- p**: Mostrar el nombre del proceso para cada puerto
- c**: Muestra resultados continuamente

netstat

Ejemplos:

```
# netstat -n --tcp -c
```

Active Internet connections (w/o servers)↑

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	272	::ffff:192.188.51.40:22	::ffff:128.223.60.27:60968	ESTABLISHED
tcp	0	0	::ffff:192.188.51.40:22	::ffff:128.223.60.27:53219	ESTABLISHED

```
# netstat -lnp --tcp
```

Active Internet connections (only servers)↑

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:199	0.0.0.0:*	LISTEN	11645/snmpd
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	1997/mysqld

```
# netstat -ic
```

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	2155901	0	0	0	339116	0	0	0	BMRU
lo	16436	0	18200	0	0	0	18200	0	0	0	LRU
eth0	1500	0	2155905	0	0	0	339117	0	0	0	BMRU
lo	16436	0	18200	0	0	0	18200	0	0	0	LRU
eth0	1500	0	2155907	0	0	0	339120	0	0	0	BMRU
lo	16436	0	18200	0	0	0	18200	0	0	0	LRU
eth0	1500	0	2155910	0	0	0	339122	0	0	0	BMRU
lo	16436	0	18200	0	0	0	18200	0	0	0	LRU
eth0	1500	0	2155913	0	0	0	339124	0	0	0	BMRU

netstat

Ejemplos:

```
# netstat -tcp -listening --program
```

```
Active Internet connections (only servers) ^
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	*:5001	*:*	LISTEN	13598/iperf
tcp	0	0	localhost:mysql	*:*	LISTEN	5586/mysqld
tcp	0	0	*:www	*:*	LISTEN	7246/apache2
tcp	0	0	t60-2.local:domain	*:*	LISTEN	5378/named
tcp	0	0	t60-2.local:domain	*:*	LISTEN	5378/named
tcp	0	0	t60-2.local:domain	*:*	LISTEN	5378/named
tcp	0	0	localhost:domain	*:*	LISTEN	5378/named
tcp	0	0	localhost:ipp	*:*	LISTEN	5522/cupsd
tcp	0	0	localhost:smtp	*:*	LISTEN	6772/exim4
tcp	0	0	localhost:953	*:*	LISTEN	5378/named
tcp	0	0	*:https	*:*	LISTEN	7246/apache2
tcp6	0	0	[::]:ftp	[::]:*	LISTEN	7185/proftpd
tcp6	0	0	[::]:domain	[::]:*	LISTEN	5378/named
tcp6	0	0	[::]:ssh	[::]:*	LISTEN	5427/sshd
tcp6	0	0	[::]:3000	[::]:*	LISTEN	17644/ntop
tcp6	0	0	ip6-localhost:953	[::]:*	LISTEN	5378/named
tcp6	0	0	[::]:3005	[::]:*	LISTEN	17644/ntop

Isof (List Open Files)

- Isof es particularmente útil porque en Unix todo es un archivo: *sockets unix, sockets ip, directorios, etc.*
- Permite asociar archivos abiertos por:
 - p**: PID (Process ID) de un proceso
 - i** : Una dirección de Internet (protocolo:puerto)
 - u**: Un usuario

Isof

- **Ejemplo:**

- Usando *netstat -ln -tcp* determino que el puerto 6010 está abierto y esperando una conexión (LISTEN).

```
# netstat -ln --tcp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:6011          0.0.0.0:*               LISTEN
```

Isof

Determinar qué proceso tiene el puerto (6010) abierto y qué otros recursos está utilizando:

```
# lsof -i tcp:6010
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
sshd	10301	root	6u	IPv4	53603		TCP	localhost.localdomain:x11-ssh-offset (LISTEN) ↵
sshd	10301	root	7u	IPv6	53604		TCP	:::1:x11-ssh-offset (LISTEN) ↵

```
# lsof -p 10301
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
sshd	10301	root	cwd	DIR	8,2	4096	2	/
sshd	10301	root	rtd	DIR	8,2	4096	2	/
sshd	10301	root	txt	REG	8,2	379720	1422643	/usr/sbin/sshd
sshd	10301	root	mem	REG	8,2	32724	1437533	/usr/lib/libwrap.so.0.7.6
sshd	10301	root	mem	REG	8,2	15088	3080329	/lib/libutil-2.4.so
sshd	10301	root	mem	REG	8,2	75632	1414093	/usr/lib/libz.so.1.2.3
sshd	10301	root	mem	REG	8,2	96040	3080209	/lib/libnsl-2.4.so
sshd	10301	root	mem	REG	8,2	100208	1414578	/usr/lib/libgssapi_krb5.so.2.2
sshd	10301	root	mem	REG	8,2	11684	1414405	/usr/lib/libkrb5support.so.0.0
sshd	10301	root	mem	REG	8,2	10368	3080358	/lib/libsetrans.so.0
sshd	10301	root	mem	REG	8,2	7972	3080231	/lib/libcom_err.so.2.1
sshd	10301	root	mem	REG	8,2	30140	1420233	/usr/lib/libcrack.so.2.8.0
sshd	10301	root	mem	REG	8,2	11168	3080399	/lib/security/pam_succeed_if.so

...

Isof

Que servicios estoy corriendo?

```
# lsof -i
COMMAND      PID        USER      FD  TYPE  DEVICE  SIZE  NODE  NAME
firefox      4429      hervey    50u  IPv4  1875852      TCP  192.168.179.139:56890->128.223.60.21:www (ESTABLISHED)
named        5378      bind      20u  IPv6   13264      TCP  *:domain (LISTEN)
named        5378      bind      21u  IPv4   13267      TCP  localhost:domain (LISTEN)
sshd         5427      root       3u  IPv6   13302      TCP  *:ssh (LISTEN)
cupsd        5522      root       3u  IPv4  1983466      TCP  localhost:ipp (LISTEN)
mysqld       5586      mysql     10u  IPv4   13548      TCP  localhost:mysql (LISTEN)
snmpd        6477      snmp       8u  IPv4   14633      UDP  localhost:snmp
exim4        6772      Debian-exim 3u  IPv4   14675      TCP  localhost:smtp (LISTEN)
ntpd         6859      ntp        16u  IPv4   14743      UDP  *:ntp
ntpd         6859      ntp        17u  IPv6   14744      UDP  *:ntp
ntpd         6859      ntp        18u  IPv6   14746      UDP  [fe80::250:56ff:fec0:8]:ntp
ntpd         6859      ntp        19u  IPv6   14747      UDP  ip6-localhost:ntp
proftpd      7185      proftpd    1u  IPv6   15718      TCP  *:ftp (LISTEN)
apache2      7246      www-data   3u  IPv4   15915      TCP  *:www (LISTEN)
apache2      7246      www-data   4u  IPv4   15917      TCP  *:https (LISTEN)
...
iperf        13598     root       3u  IPv4  1996053      TCP  *:5001 (LISTEN)
apache2      27088     www-data   3u  IPv4   15915      TCP  *:www (LISTEN)
apache2      27088     www-data   4u  IPv4   15917      TCP  *:https (LISTEN)
```

tcpdump

- Muestra los encabezados de los paquetes recibidos en una interfaz dada. Opcionalmente, filtra basado en expresiones booleanas
- Permite escribir la información en un archivo para su posterior análisis
- Requiere privilegios de administrador (root), dado que ha de configurarse la interfaz de red en modo 'promiscuo'
 - Nota: El modo 'promiscuo' pierde utilidad cuando la conexión es a través de un switch.

tcpdump

Parámetros relevantes:

- **-i** : Especificar la interfaz (ej: -i eth0)
- **-l** : Pasar la salida por un búfer de líneas (vea mientras que estas capturando paquetes)
- **-v**, **-vv**, **-vvv**: Cada vez más información
- **-n** : No traducir direcciones IP a nombres (evita DNS)
- **-nn** : No traducir números de puerto
- **-w** : Escribir los paquetes a un archivo
- **-r** : Leer paquetes de un archive creado con '-w'

tcpdump

- Expresiones booleanas
 - Utilizan los operadores 'AND', 'OR', 'NOT'
 - Consisten de una o más primitivas, las cuales consisten en un cualificador y un ID (nombre o número)
 - Expresión ::= [NOT] <primitiva> [AND | OR | NOT <primitiva> ...]
 - <primitiva> ::= <cualificador> <nombre|número>
 - <cualificador> ::= <tipo> | <dirección> | <protocolo>
 - <tipo> ::= host | net | port | portrange
 - <dirección> ::= src | dst
 - <protocolo> ::= ether | fddi | tr | wlan | ip | ip6 | arp | rarp | decnet | tcp | udp

tcpdump

Ejemplos:

- Mostrar todo el tráfico HTTP originando en 192.168.1.1

```
# tcpdump -lnXvvv port 80 and src host 192.168.1.1
```

- Mostrar todo el tráfico originando en 192.168.1.1 excepto SSH

```
# tcpdump -lnXvvv src host 192.168.1.1 and not port 22
```

wireshark

- Wireshark es un analizador con interfaz gráfica basado en *libpcap*, la misma biblioteca de captura de paquetes utilizada por *tcpdump*
- La interfaz gráfica ofrece ciertas ventajas, por ejemplo:
 - ~ Visualización jerárquica por protocolo (drill-down)
 - ~ Mostrar una 'conversación' TCP (Follow TCP Stream)
 - ~ Colores para distinguir tipos de tráfico
 - ~ Múltiples estadísticas, gráficos, etc.

wireshark

- Wireshark vino después que *Etheral*
- La combinación de *tcpdump* y *wireshark* pueden ser bastante poderoso. Por ejemplo:

```
- # tcpdump -i eth1 -A -s1500 -2 dump.log port 21  
- $ sudo wireshark -r dump.log
```



wireshark

The image shows the Wireshark network protocol analyzer interface. The main display area shows a list of 12 captured packets, all of which are ICMP Echo (ping) requests and replies between the local host (127.0.0.1). The packets are displayed in a table with columns for No., Time, Source, Destination, Protocol, and Info.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	127.0.0.1	127.0.0.1	ICMP	Echo (ping) request
2	0.000026	127.0.0.1	127.0.0.1	ICMP	Echo (ping) reply
3	0.999003	127.0.0.1	127.0.0.1	ICMP	Echo (ping) request
4	0.999029	127.0.0.1	127.0.0.1	ICMP	Echo (ping) reply
5	1.998003	127.0.0.1	127.0.0.1	ICMP	Echo (ping) request
6	1.998028	127.0.0.1	127.0.0.1	ICMP	Echo (ping) reply
7	2.997007	127.0.0.1	127.0.0.1	ICMP	Echo (ping) request
8	2.997032	127.0.0.1	127.0.0.1	ICMP	Echo (ping) reply
9	3.996674	127.0.0.1	127.0.0.1	ICMP	Echo (ping) request
10	3.996698	127.0.0.1	127.0.0.1	ICMP	Echo (ping) reply
11	4.996671	127.0.0.1	127.0.0.1	ICMP	Echo (ping) request
12	4.996695	127.0.0.1	127.0.0.1	ICMP	Echo (ping) reply

Below the packet list, the details pane shows the structure of the selected packet (Frame 1). It indicates that the packet is 98 bytes on wire and 98 bytes captured. The details pane shows the Ethernet II header (Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00) and the Internet Protocol header (Src: 127.0.0.1, Dst: 127.0.0.1). The packet bytes are displayed in hexadecimal and ASCII format.

File: "/tmp/etherXXXXzJGv70" 1392 Bytes 00:00:04 Packets: 12 Displayed: 12 Marked: 0 Dropped: 0 Profile: Def...

iptraf

- **Múltiples estadísticas que se puede medir y muchas funciones**
 - Por protocolo/puerto
 - Por tamaño de paquetes
 - Genera logs
 - Utiliza DNS para traducir direcciones
- **Ventajas**
 - Simplicidad
 - Basado en menús (utiliza “curses”)
 - Configuración flexible

iptraf

- Es posible ejecutar periódicamente en background (-B)
 - Permite, por ejemplo, ejecutar desde un cron job y analizar los logs periódicamente.
 - Generar alarmas
 - Guardar en una base de datos
 - Tiene un nombre genial... “Interactive Colorful IP LAN Monitor”

Example: `iptraf -i eth1`

ntop: Network Top

- **Equivalente a top, pero para información de red**
 - Información por nodo, protocolo de red, protocolo IP, estadísticas, gráficos, etc.
- **Interfaz web (servidor web integrado!)**
 - Con soporte SSL
- **Dispone de varios plugins que extienden sus funcionalidades**
 - Archivos RRD
 - Análisis de NetFlow

The logo for ntop, featuring the word "ntop" in a stylized, orange, pixelated font.

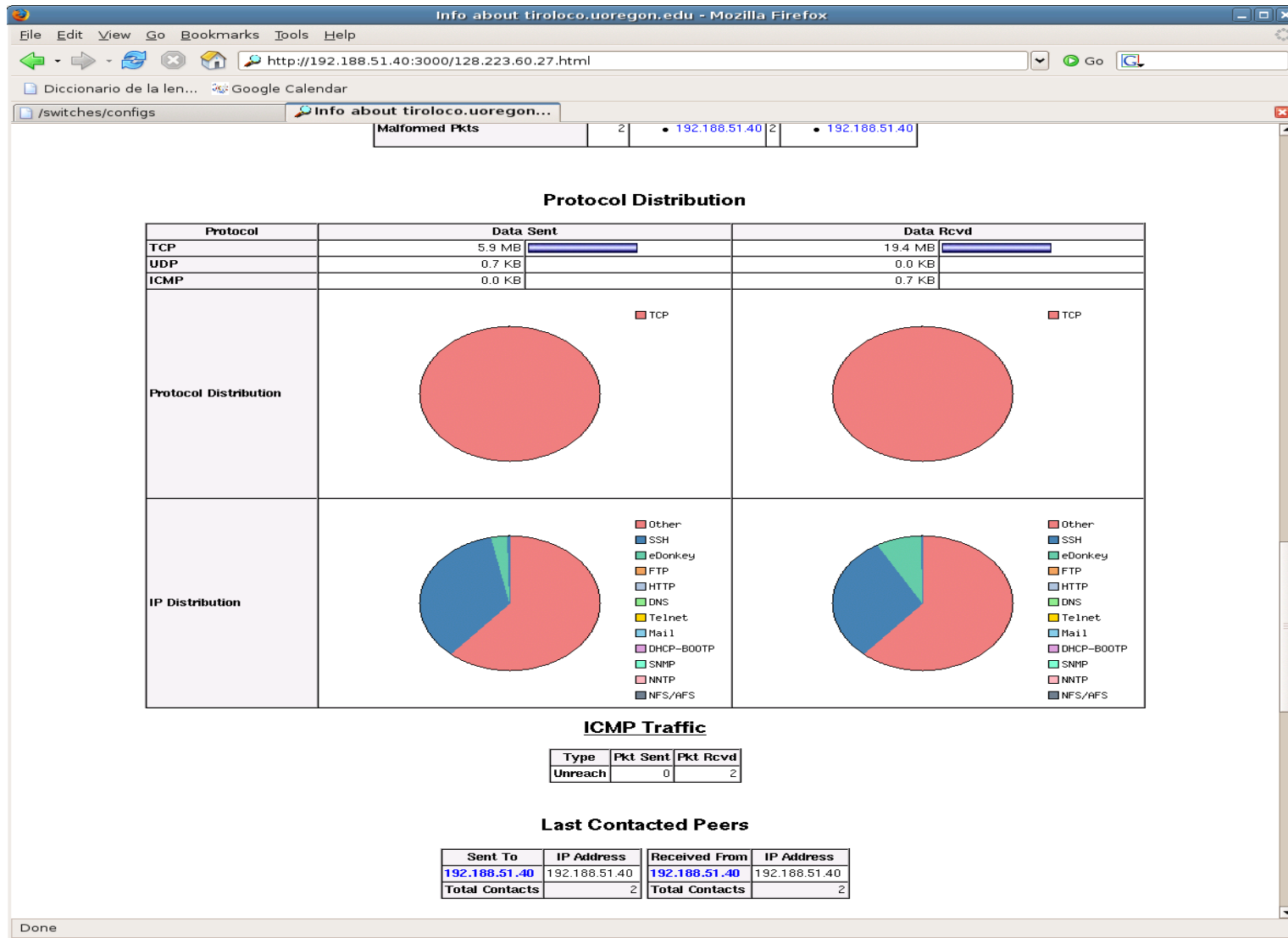
ntop

- Puede ejecutarse en modo daemon con SSL:
 - -d : daemon
 - -W <puerto> : Escuchar en el puerto 3005, modo SSL

```
ntop -d -W 3005
```

- Para ver el interfaz de web ir al:
 - <http://localhost:3000>
 - <https://localhost:3005>

ntop



ntop

Incluye un parámetro que permite crear un archivo con información sobre paquetes “sospechosos”

```
-q | --create-suspicious-packets
```

This parameter tells ntop to create a dump file of suspicious packets. There are many, many, things that cause a packet to be labeled as 'suspicious', including:

Detected ICMP fragment

Detected Land Attack against host

Detected overlapping/tiny packet fragment

Detected traffic on a diagnostic port

Host performed ACK/FIN/NULL scan

Host rejected TCP session

HTTP/FTP/SMTP/SSH detected at wrong port

Malformed TCP/UDP/ICMP packet (packet too short)↑

Packet # %u too long

Received a ICMP protocol Unreachable from host

Sent ICMP Administratively Prohibited packet to host

Smurf packet detected for host

TCP connection with no data exchanged

TCP session reset without completing 3-way handshake

Two MAC addresses found for the same IP address

UDP data to a closed port

Unknown protocol (no HTTP/FTP/SMTP/SSH) detected (on port 80/21/25/22)↑

Unusual ICMP options

ntop

- Luego de hacer una captura con -q, es posible analizar más detalladamente los paquetes sospechosos con ethereal:

```
ethereal -r /usr/local/var/ntop/ntop-suspicious-pkts.deveth0.pcap
```

nmap

Network MAPper

- Bien complejo. La cantidad de parámetros es impresionante.
- Muy útil para ver que esta corriendo en un red o si servidores o servicios están disponibles.
- Ojo con el uso de nmap! Puede ver como un ataque al recipiente de un escaneo de nmap.
- Todo sobre nmap esta disponible aquí:

<http://insecure.org/>

nmap

Ejemplos

- Veamos que servidores están corriendo en un red:

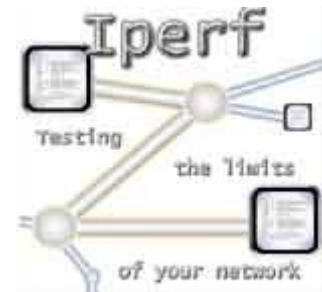
```
# nmap -sP 192.168.5.0/24
```

- Veamos que sistema operativos hay y que servicios están corriendo en un red:

```
# nmap -sT -O 206.212.15.0-50
```

iperf

- Para medir el rendimiento de la red entre dos puntos
- *iperf* tiene dos modos, *servidor* y *cliente*
- Fácil de usar
- Excelente para determinar los parámetros optimales de TCP
 - Tamaño de ventana de TCP por el rendimiento óptimo



iperf

- Usando UDP uno puede generar reportajes de perdida de paquetes y/o *jitter*
- Puede correr varias sesiones usando *threads*
- Apoya IPv6

Parámetros de Iperf

```
Usage: iperf [-s|-c host] [options]
       iperf [-h|--help] [-v|--version]
```

Client/Server:

-f, --format	[kmKM]	format to report: Kbits, Mbits, KBytes, MBytes
-i, --interval	#	seconds between periodic bandwidth reports
-l, --len	#[KM]	length of buffer to read or write (default 8 KB)'
-m, --print_mss		print TCP maximum segment size (MTU - TCP/IP header)'
-p, --port	#	server port to listen on/connect to
-u, --udp		use UDP rather than TCP
-w, --window	#[KM]	TCP window size (socket buffer size)'
-B, --bind	<host>	bind to <host>, an interface or multicast address
-C, --compatibility		for use with older versions does not sent extra msgs
-M, --mss	#	set TCP maximum segment size (MTU - 40 bytes)'
-N, --nodelay		set TCP no delay, disabling Nagle's Algorithm
-V, --IPv6Version		Set the domain to IPv6

Server specific:

-s, --server	run in server mode
-U, --single_udp	run in single threaded UDP mode
-D, --daemon	run the server as a daemon

Client specific:

-b, --bandwidth	#[KM]	for UDP, bandwidth to send at in bits/sec (default 1 Mbit/sec, implies -u)'
-c, --client	<host>	run in client mode, connecting to <host>
-d, --dualtest		Do a bidirectional test simultaneously
-n, --num	#[KM]	number of bytes to transmit (instead of -t)'
-r, --tradeoff		Do a bidirectional test individually
-t, --time	#	time in seconds to transmit for (default 10 secs)'
-F, --fileinput	<name>	input the data to be transmitted from a file
-I, --stdin		input the data to be transmitted from stdin
-L, --listenport	#	port to recieve bidirectional tests back on
-P, --parallel	#	number of parallel client threads to run
-T, --ttl	#	time-to-live, for multicast (default 1)'

Ejemplo “iperf - TCP”

```
$ iperf -s
```

```
-----  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default) ^  
-----
```

```
[  4] local 128.223.157.19 port 5001 connected with 201.249.107.39  
port 39601  
[  4]  0.0-11.9 sec      608 KBytes      419 Kbits/sec  
-----
```

```
# iperf -c nsrc.org
```

```
-----  
Client connecting to nsrc.org, TCP port 5001  
TCP window size: 16.0 KByte (default) ^  
-----
```

```
[  3] local 192.168.1.170 port 39601 connected with 128.223.157.19  
port 5001  
[  3]  0.0-10.3 sec      608 KBytes      485 Kbits/sec  
-----
```

Ejemplo “Iperf - UDP”

```
# iperf -c host1 -u -b100M
```

```
-----  
Client connecting to nsdb, UDP port 5001
```

```
Sending 1470 byte datagrams
```

```
UDP buffer size: 106 KByte (default)↑
```

```
-----  
[ 3] local 128.223.60.27 port 39606 connected with 128.223.250.135 port 5001
```

```
[ 3] 0.0-10.0 sec 114 MBytes 95.7 Mbits/sec
```

```
[ 3] Sent 81377 datagrams
```

```
[ 3] Server Report:
```

```
[ 3] 0.0-10.0 sec 114 MBytes 95.7 Mbits/sec 0.184 ms 1/81378 (0.0012%)↑
```

```
$ iperf -s -u -i 1
```

```
-----  
Server listening on UDP port 5001
```

```
Receiving 1470 byte datagrams
```

```
UDP buffer size: 108 KByte (default)↑
```

```
-----  
[ 3] local 128.223.250.135 port 5001 connected with 128.223.60.27 port 39606
```

```
[ 3] 0.0- 1.0 sec 11.4 MBytes 95.4 Mbits/sec 0.184 ms 0/ 8112 (0%)↑
```

```
[ 3] 1.0- 2.0 sec 11.4 MBytes 95.7 Mbits/sec 0.177 ms 0/ 8141 (0%)↑
```

```
[ 3] 2.0- 3.0 sec 11.4 MBytes 95.6 Mbits/sec 0.182 ms 0/ 8133 (0%)↑
```

```
...↑
```

```
[ 3] 8.0- 9.0 sec 11.4 MBytes 95.7 Mbits/sec 0.177 ms 0/ 8139 (0%)↑
```

```
[ 3] 9.0-10.0 sec 11.4 MBytes 95.7 Mbits/sec 0.180 ms 0/ 8137 (0%)↑
```

```
[ 3] 0.0-10.0 sec 114 MBytes 95.7 Mbits/sec 0.184 ms 1/81378 (0.0012%)↑
```

Bibliografía

Monitoring Virtual Memory with vmstat

<http://www.linuxjournal.com/article/8178>

Ejemplo Básico de tcpdump (Español)

<http://luauf.com/2008/06/21/ejemplo-basico-de-tcpdump/>