Linux BridgeWall: il firewall che non c'e'

Come proteggere la propria rete senza accorgersene
(....o quasi)

Luca Carbone
INFN – Sezione di Milano Bicocca
(complice il Gruppo Security di CCR)

Programma iniziale (~*verbatim*)

- Analisi preliminare dei rischi;
- *Definizione di strumenti (possibilmente open source) e topologie standard (o quasi) che coprano la maggioranza delle esigenze dei siti INFN;*
- *Formulazione di alcune proposte implementative;*
- *Sperimentazione (prestazioni, affidabilita', ...)*
- Individuazione di strumenti di controllo/logging e prevenzione da integrare negli scenari proposti.

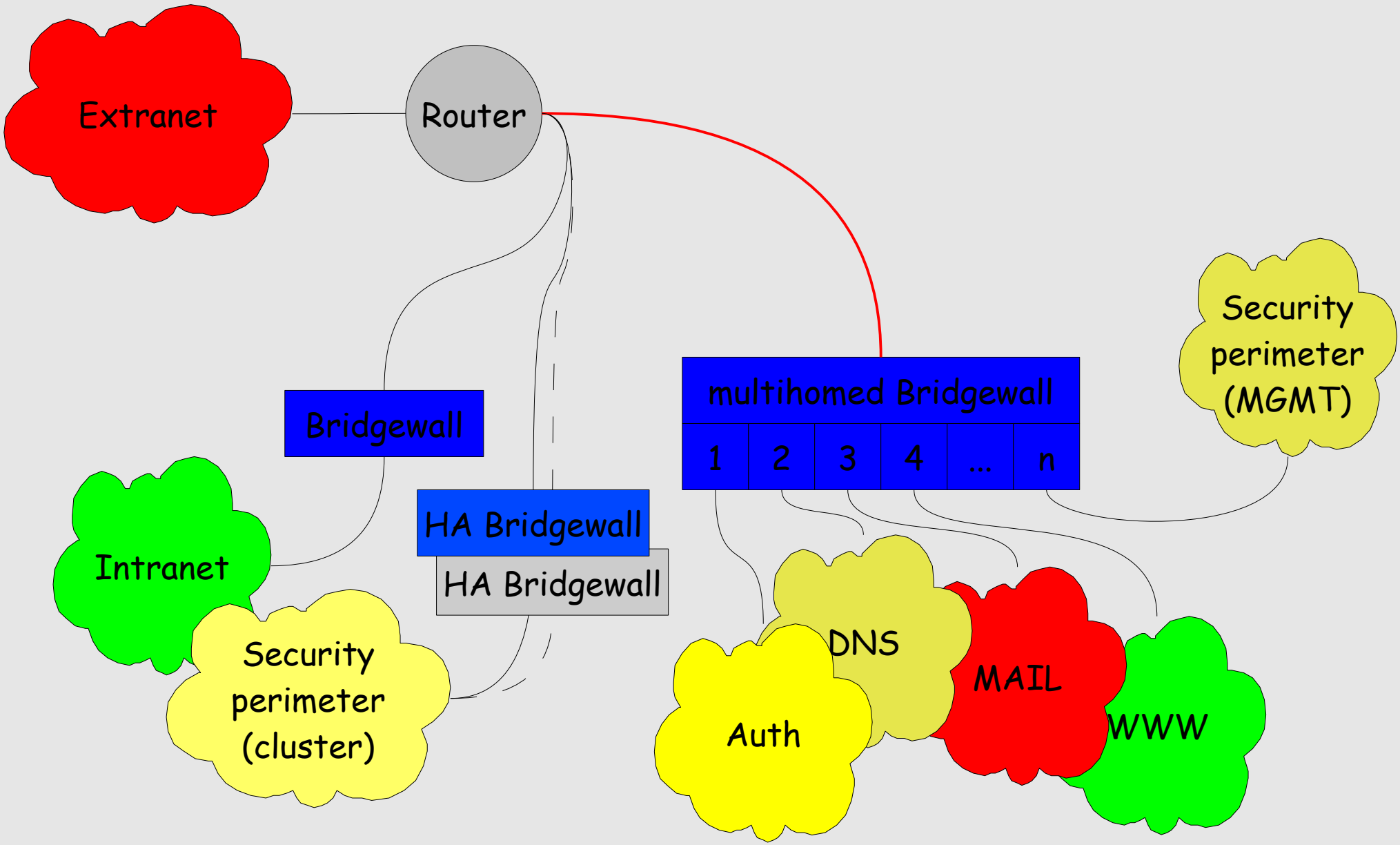
Il problema:

- E' possibile implementare una Security Policy (fisica & logica) tale da:
  - *rendere piu' sicura la rete senza 1) sacrificare (troppo) prestazioni e comodita' d'accesso ai servizi, e 2) affrontare pesanti ristrutturazioni (ad es. routing)?*
  - non basare la sicurezza della rete su *buone pratiche* (spesso/talvolta) al di fuori del nostro controllo?
  - minimizzare la possibilita' di *effetto Domino?*
  - minimizzare il numero di punti di controllo/auditing?

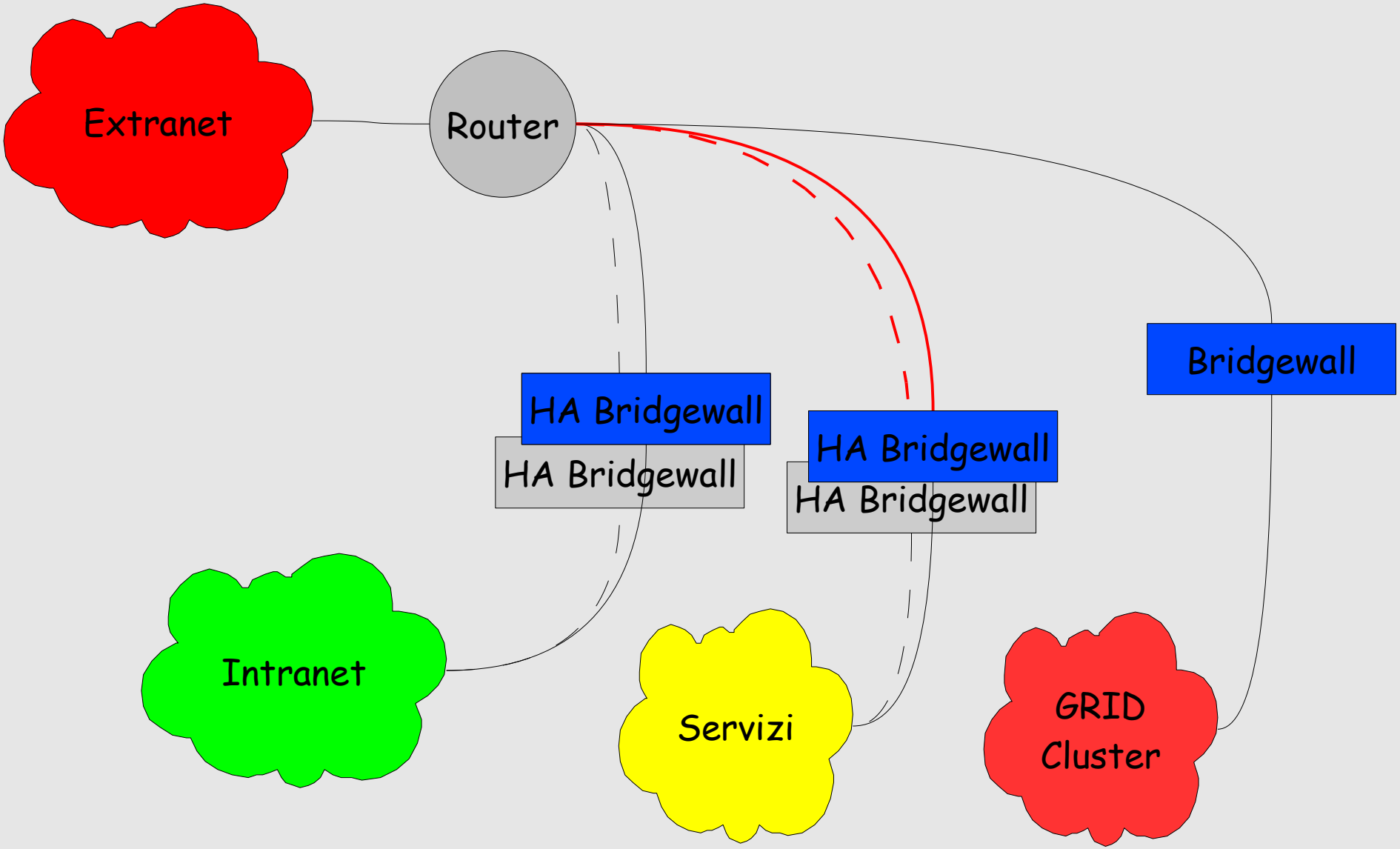
... ed una sua possibile soluzione (l'uovo di Colombo?)

- **BRIDGEWALL = BRIDGE + FIREWALL**
  - **stealth (transparent) firewalling** tramite apparato di livello 2 (**invisibile = inaccessibile**) con funzionalita' di filtering/inspection ai livelli 2,3,4,...,7 della pila OSI
  - implementabile con:
    - soluzioni proprietarie (Cisco, Juniper, SonicWALL)
    - **soluzioni Open Source** (Linux bridge + ebtables/arptables/iptables/ipset/nf-hipac...)

Topologia possibile (1)

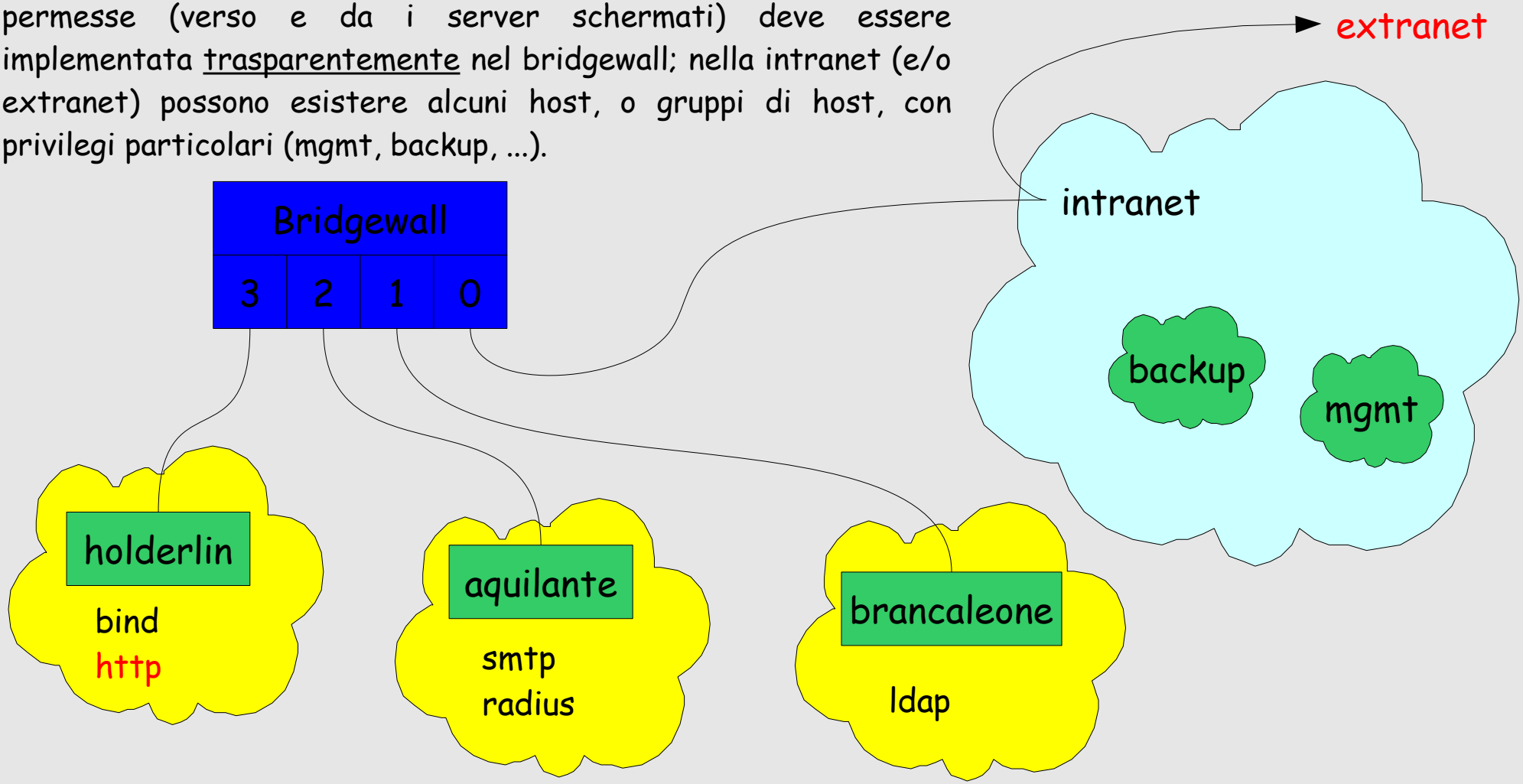


Topologia possibile (2)



LBW testbed – il caso piu' complicato

E' possibile isolare/schermare completamente i servizi sensibili con una semplice linuxbox? Tutta la **matrice delle connessioni** permesse (verso e da i server schermati) deve essere implementata trasparentemente nel bridgewall; nella intranet (e/o extranet) possono esistere alcuni host, o gruppi di host, con privilegi particolari (mgmt, backup, ...).



Matrice delle connessioni

L'idea (perversa e/o paranoica...): ad ogni macchina dietro il BW sono consentite solo le connessioni indispensabili al servizio che svolge sulle porte di pertinenza. Ad esempio: il mail server (MTA) deve essere raggiungibile da Extranet ed Intranet (sulla porta 25,...), ed a sua volta deve poter contattare il server DNS (su 53 tcp/udp), il server Auth (es., LDAP: 636/tcp) e via dicendo. Ogni altra possibile connessione e' – di fatto – inutile (e pericolosa in caso di compromissione), e puo'/deve quindi essere **trasparentemente** proibita direttamente sul BW.

SRC\DST	**Extranet**	**DNS**	**MAIL**	**WWW**	**AUTH**	**MBOX**	**Intranet**
Extranet		allow	allow	allow	limited		NA
DNS	allow						
MAIL	allow	allow			allow	allow	allow
WWW		allow			allow		
AUTH	limited	allow					
MBOX		allow			allow		
Intranet	allow	allow	allow	allow	allow	allow	

Implementazione della matrice
(file di configurazione ad hoc + perl)

```
<in>
  <aquilante>
    25/tcp      any
    465/tcp     any
    587/tcp     any
    80/tcp      local
    443/tcp     local
    22/tcp      mgmt
    5000/tcp    mgmt
    9102/tcp    backupC
    12865/tcp   local
    1812/udp    radiusC
  </aquilante>
</in>
```

```
<out>
  <aquilante>
    25/tcp      any
    443/tcp     any
    80/tcp      any
    53/tcp      bindSrv
    53/udp      bindSrv
    123/udp     ntpSrv
    88/udp      krbSrv
    389/tcp     ldapSrv
    636/tcp     ldapSrv
    9103/tcp    backupSrv
    514/udp     logSrv
    1812/udp    radiusSrv
  </aquilante>
</out>
```

Implementare il tutto con iptables puo' essere un discreto problema gia' con una sola macchina (siamo gia' a =>14 regole, e le iptables non scalano bene, e non permettono di definire strutture di particolare intelligenza). Quindi????

0.0.0.0/0

management boxes
novalis.mib.infn.it
ssire.mib.infn.it

radius clients
local networks
capetto.fi.infn.it

radius servers
capetto.fi.infn.it

IPSET!!!

- Almost verbatim from http://ipset.netfilter.org/:

IP sets are a framework inside the Linux 2.4.x and 2.6.x kernel, which can be administered by the *ipset* utility. Depending on the type, currently an IP set may store IP addresses, (TCP/UDP) port numbers or IP addresses with MAC addresses in a way, which ensures lightning speed when matching an entry against a set.

If you want to:

•**store multiple IP addresses or port numbers and match against the collection by iptables at one swoop**;

•**dynamically update iptables rules against IP addresses or ports without performance penalty**;

•**express complex IP address and ports based rulesets with one single iptables rule and benefit from the speed of IP sets**

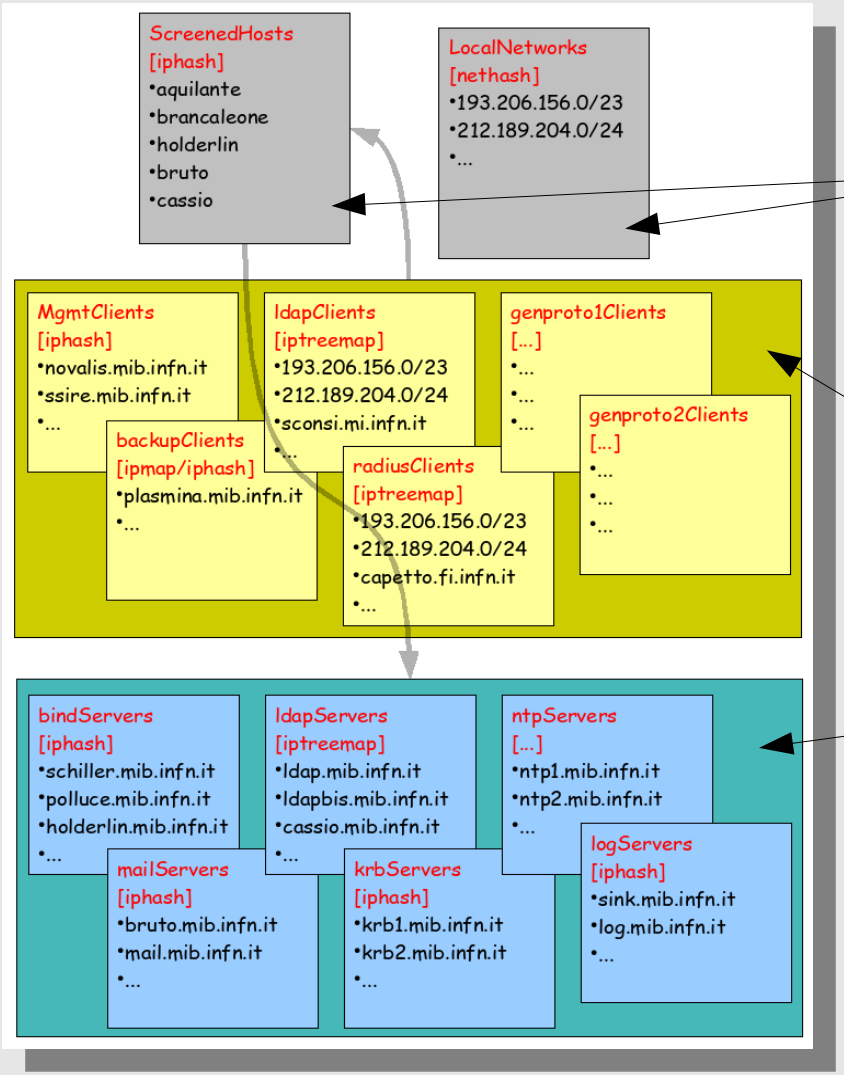
then ipset may be the proper tool for you.

set & bindings – un esempio semplice

```
# targets: ipmap set
# ipmap type: memory range where each bit represents one IP address
ipset -N targets ipmap --network 193.206.156.0/23
ipset -A targets novalis.mib.infn.it
ipset -A targets ssire.mib.infn.it
ipset -A targets promiscuo.mib.infn.it
# ports: portmap set
# portmap type: memory range where each bit represents one port
ipset -N ports portmap --from 1 --to 1024
ipset -A ports 22
ipset -A ports 25
ipset -A ports 80
# bind 'ports' set to novalis, ssire
ipset -B targets ssire -b ports
ipset -B targets novalis -b ports
# iptables rule using the set match & bindings
...
iptables -A FORWARD -m set --set servers dst,dst -j ACCEPT
...
```

Firewall will forward pkts destined to any port on *promiscuo*, while only ports 22,25,80 will be reachable on *ssire* & *novalis* ->1 line, N matches. Ports can be added to *ports* (and hosts to *targets*) on the fly.

Sets, sets, sets, ...



Basic sets:

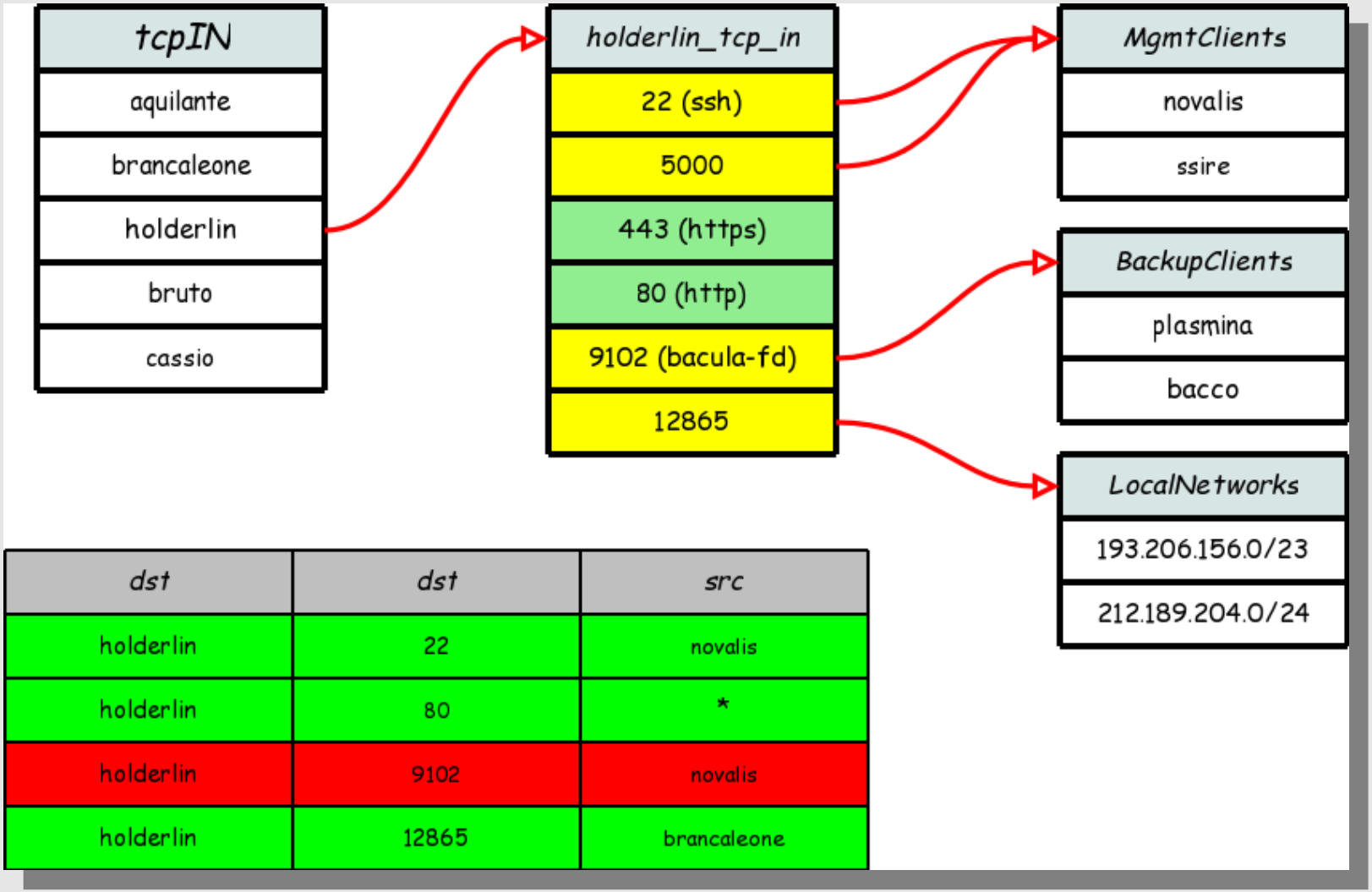
- ScreenedHosts
- LocalNetworks

Clients (in)

Servers (out)

on test BW: 34 sets (max 256, configurable parameter)

... set tcpIN dst,dst,src ...



dst	dst	src
holderlin	22	novalis
holderlin	80	*
holderlin	9102	novalis
holderlin	12865	brancaleone

Bridgewall rules w/ipset

```
Chain PREROUTING (policy ACCEPT) # raw table
target     prot src dst
NOTRACK    all  *   *   PHYSDEV match   physdev-in UPLINK ! Set ScreenedHosts dst
DROP       all  *   *   PHYSDEV match   physdev-in UPLINK ! Set ScreenedHosts dst
Chain FORWARD (policy DROP) # filter table
target     prot src dst
ACCEPT     all  *   * state RELATED,ESTABLISHED
DROP       all  *   * state INVALID
LOG_DENY   all  *   * state NEW set hostDeny src
LOG_DENY   all  *   * state NEW set portDeny dst
TCPSRC     tcp  *   * state NEW PHYSDEV match ! --physdev-in UPLINK
TCPDST     tcp  *   * state NEW PHYSDEV match ! --physdev-out UPLINK ...
UDPSRC     udp  *   * state NEW PHYSDEV match ! --physdev-in UPLINK
UDPDST     udp  *   * state NEW PHYSDEV match ! --physdev-out UPLINK ...
ACCEPT     icmp *   * PHYSDEV match ! --physdev-in UPLINK icmp echo-request
LOG_DROP   all  *   *
Chain TCPDST # filter table
target     prot src dst
ACCEPT     all  *   *   set tcpIN dst,dst,src
LOG_DROP   all  *   *
Chain LOG_DROP # filter table
target     prot src dst
SET        all  *   *   ! set hostDeny src add-set hostDeny src
LOG        all  *   *   ...
DROP       all  *   *   ...
```

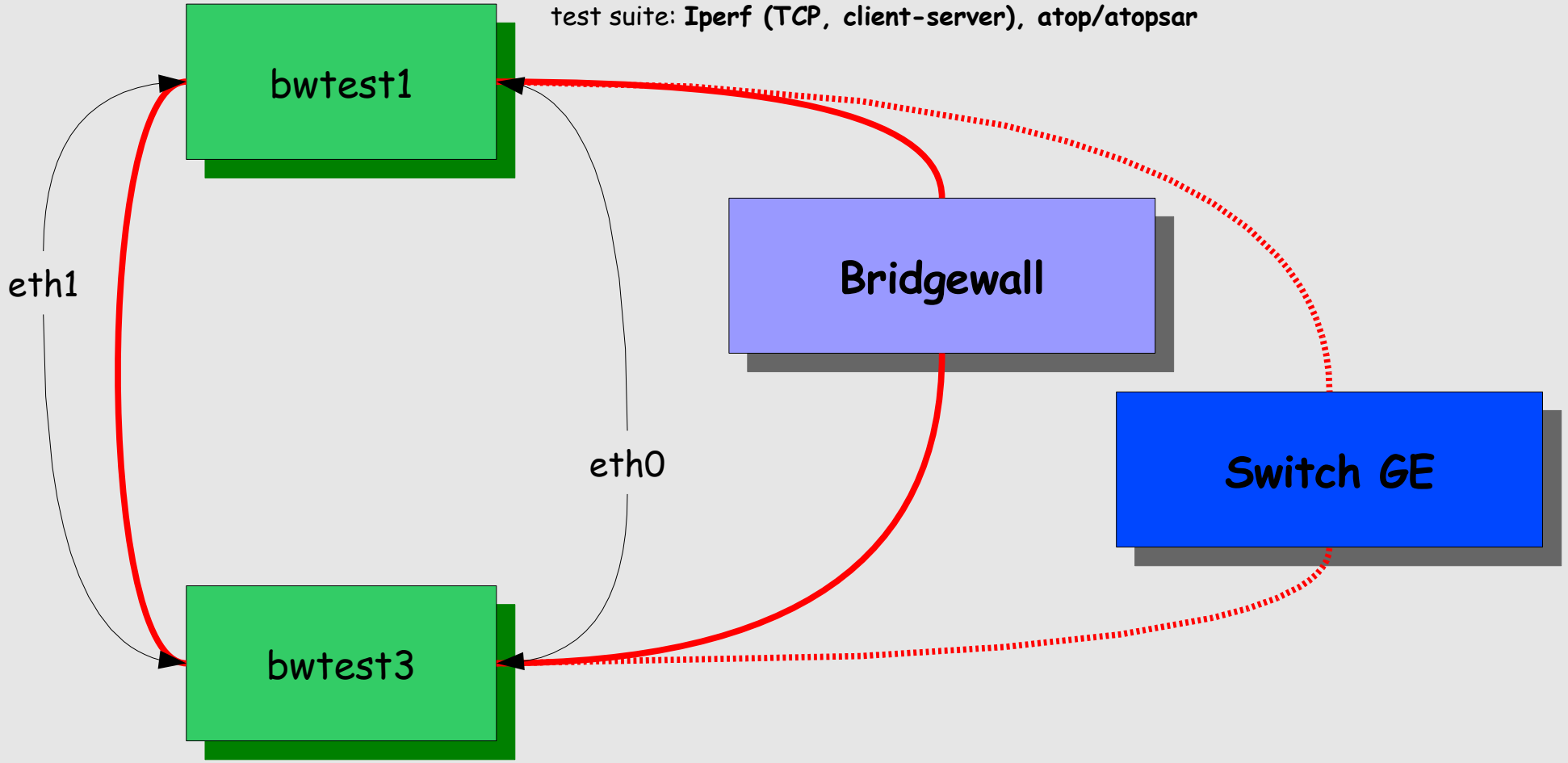
Dynamic update: *bad hosts* added to hostDeny set at runtime

Bridgewall: H/W & S/W, test

- XEON dual core 3060 @ 2.40GHz
- 2GB RAM
- 4xIntel Gbit Eth NIC (1 irq line/NIC, irq -> CPU#0)
  - 2x82573 on board
  - 2x82571 on PCI-E 4x dual port card
- Fedora 7, kernel 2.6.23 (~ FC-out-of-the-box + *ipset*)
- e1000 driver: 7.3.20-k2-NAPI
- iptables 1.3.8, ipset 2.3.0, ebtables 2.0.8
- netfilter tweaking – conntrack table max size:
  - **`nf_conntrack_max = 262144`**
- Test: throughput, latency, connection setup rate ...

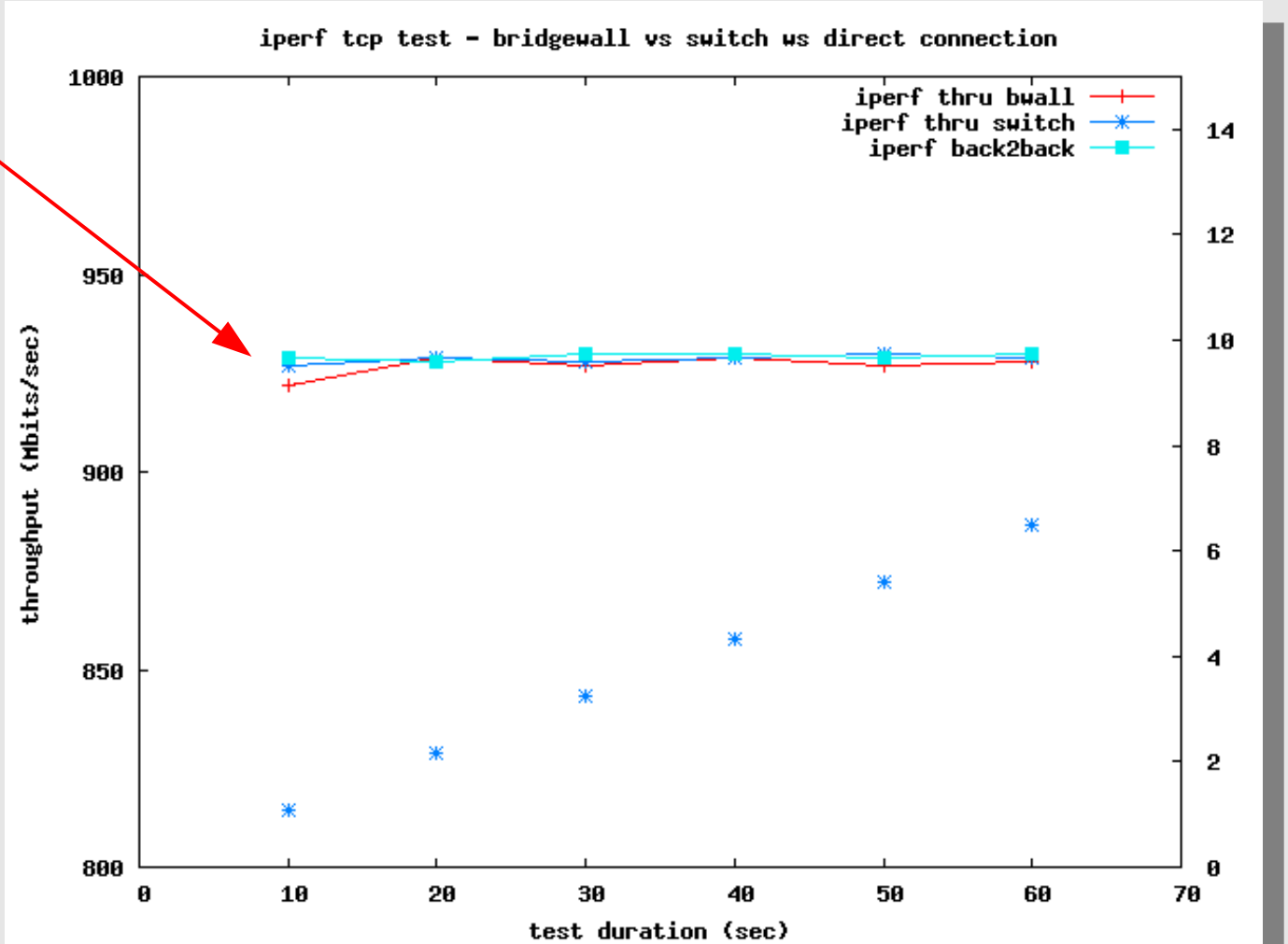
Throughput – test setup

bwtest1, bwtest3, Bridgewall: Xeon 3060 @ 2.40GHz, 2GB RAM, Intel 825XX PCI-E
Switch GE: Nortel 5510-48T – 48 x 10/100/1000
test suite: **Iperf (TCP, client-server), atop/atopsar**

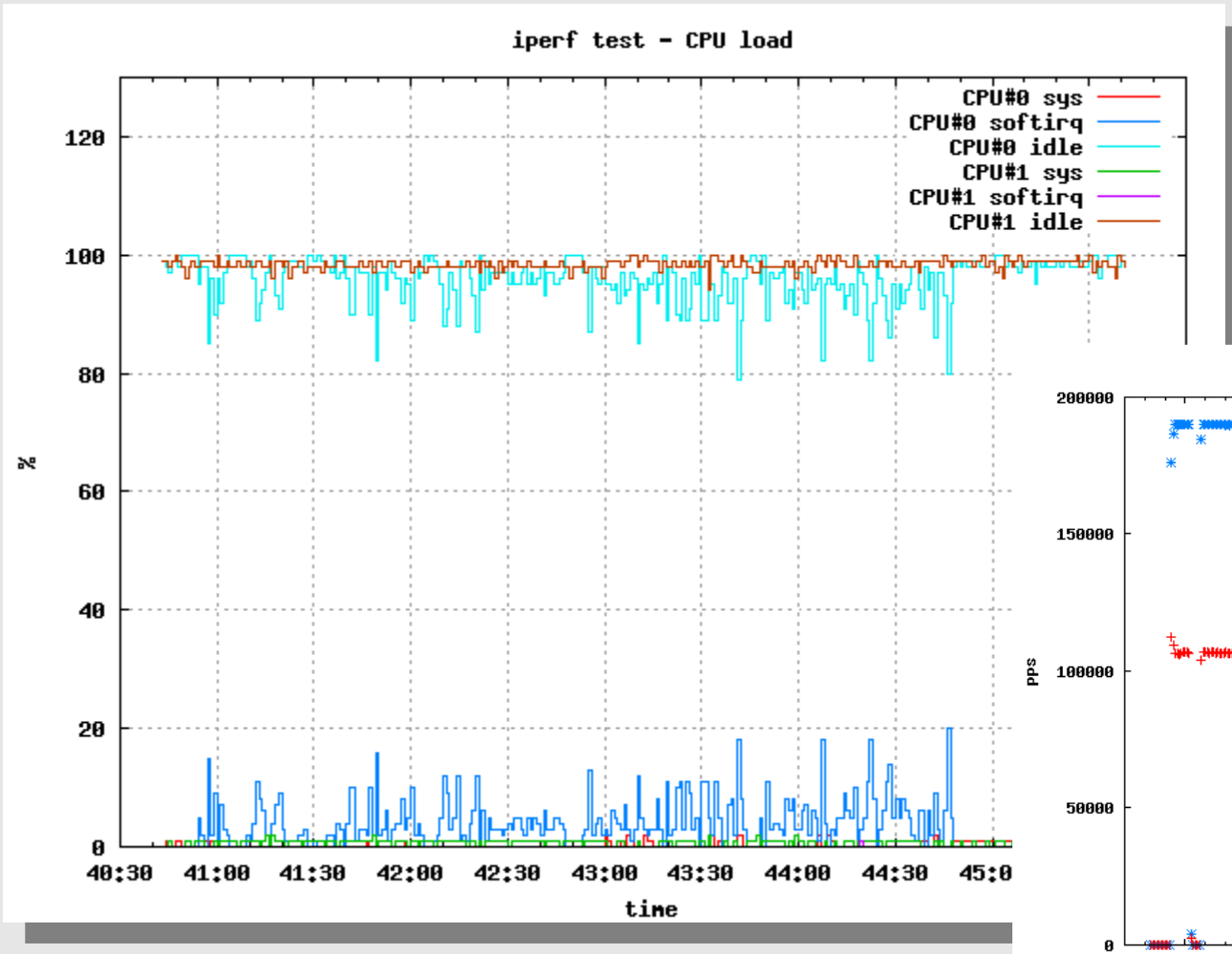


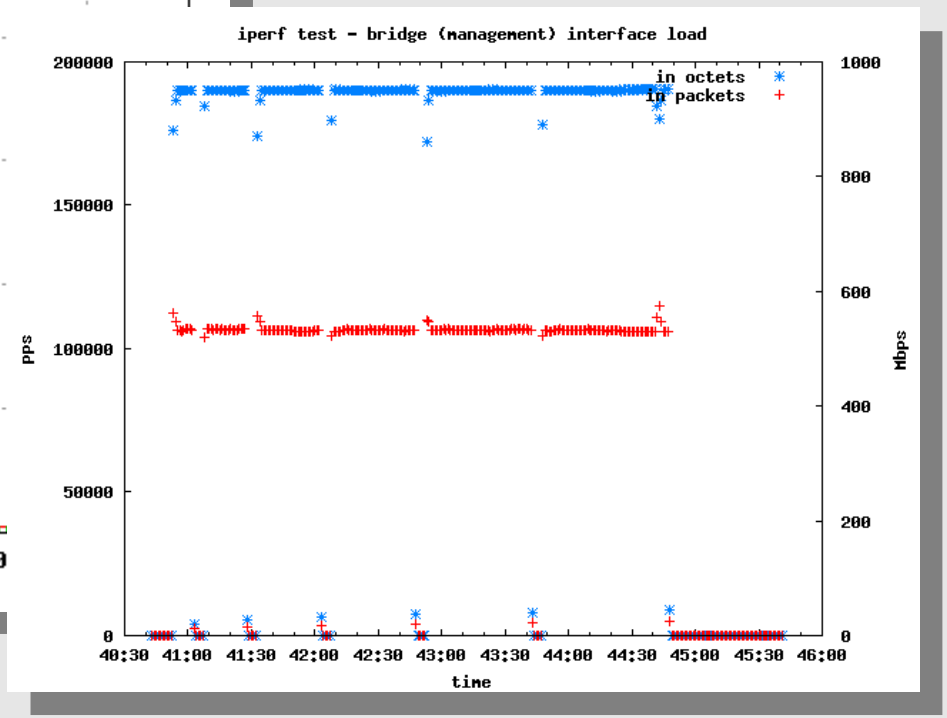
Throughput - BW vs switch vs b2b

~930 Mbps



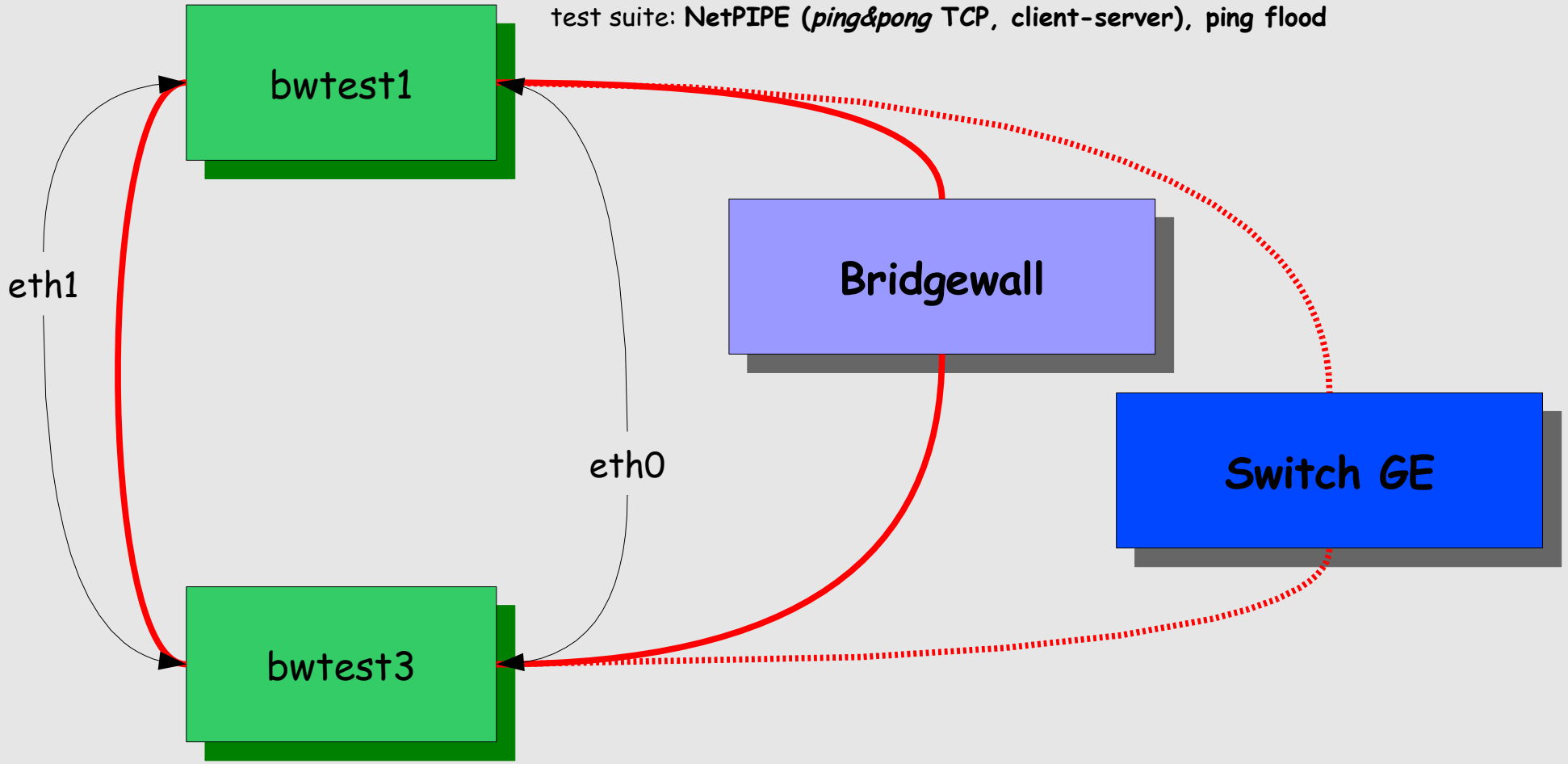
Throughput - BW CPU & net load



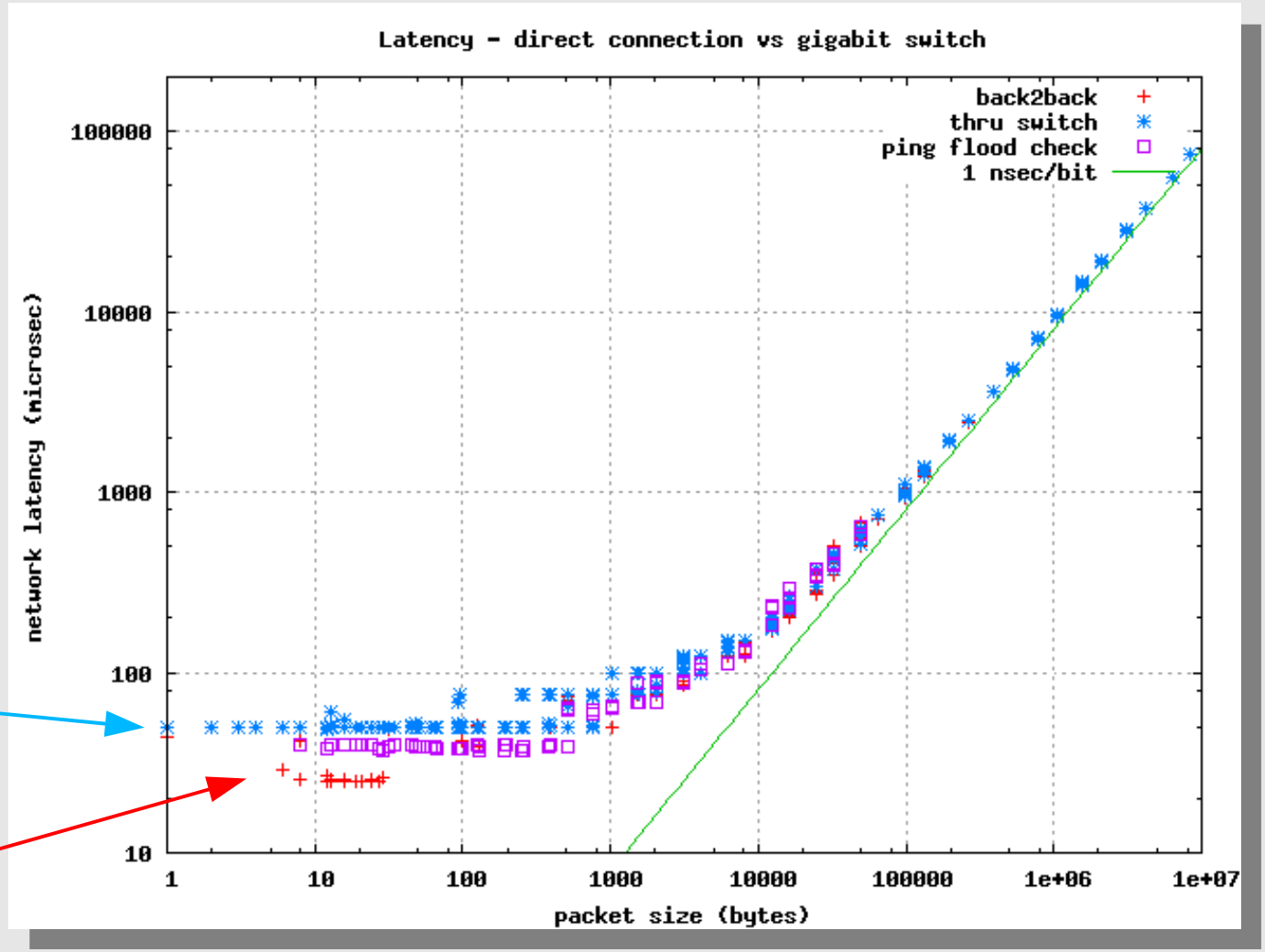


Latency - test setup

bwtest1, bwtest3, Bridgewall: Xeon 3060 @ 2.40GHz, 2GB RAM, Intel 825XX PCI-E
Switch GE: Nortel 5510-48T - 48 x 10/100/1000
test suite: **NetPIPE (*ping&pong* TCP, client-server), ping flood**



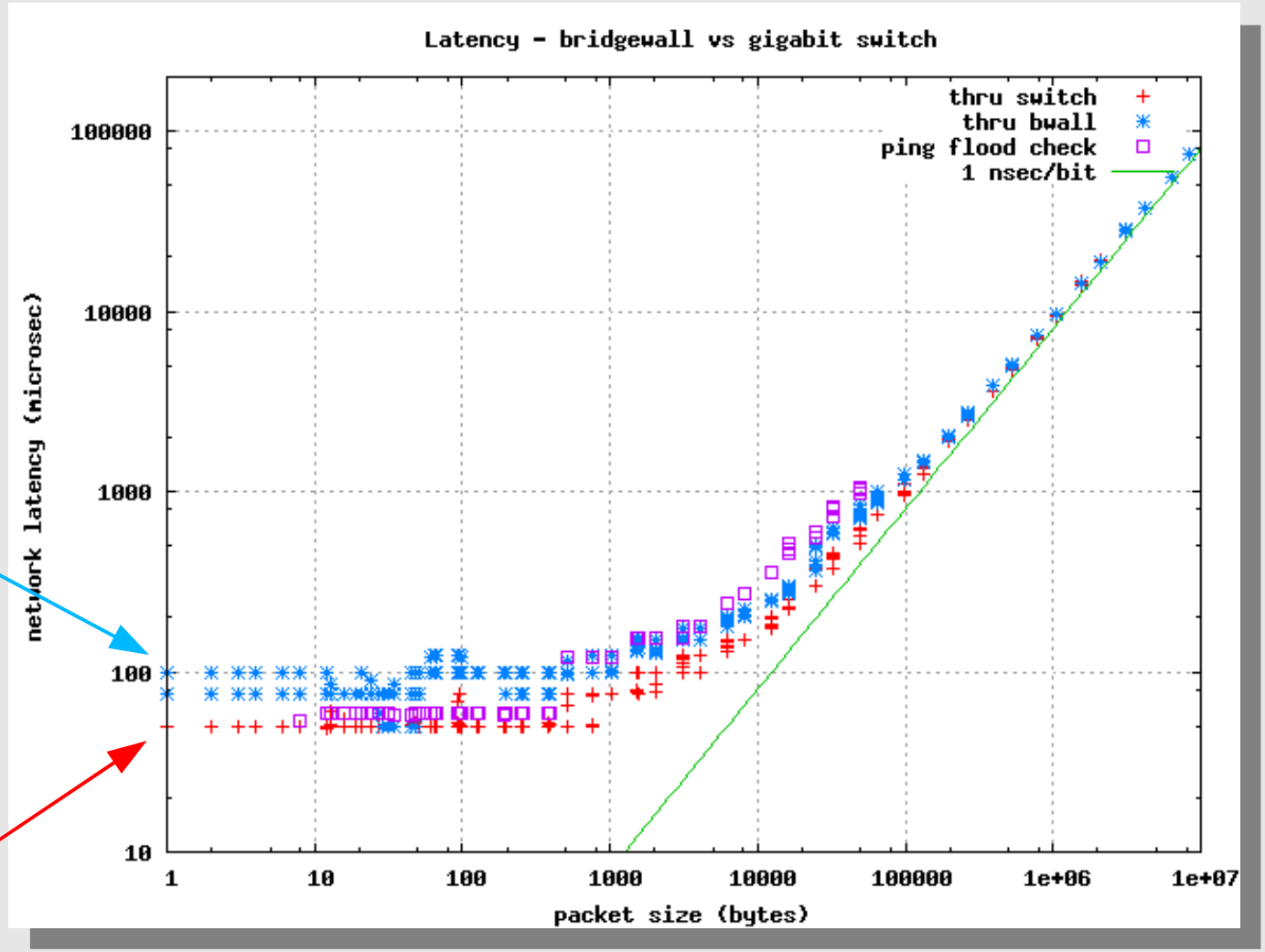
Latency - back2back vs switch



~50 µs

~25-50 µs

Latency - bridgewall vs switch

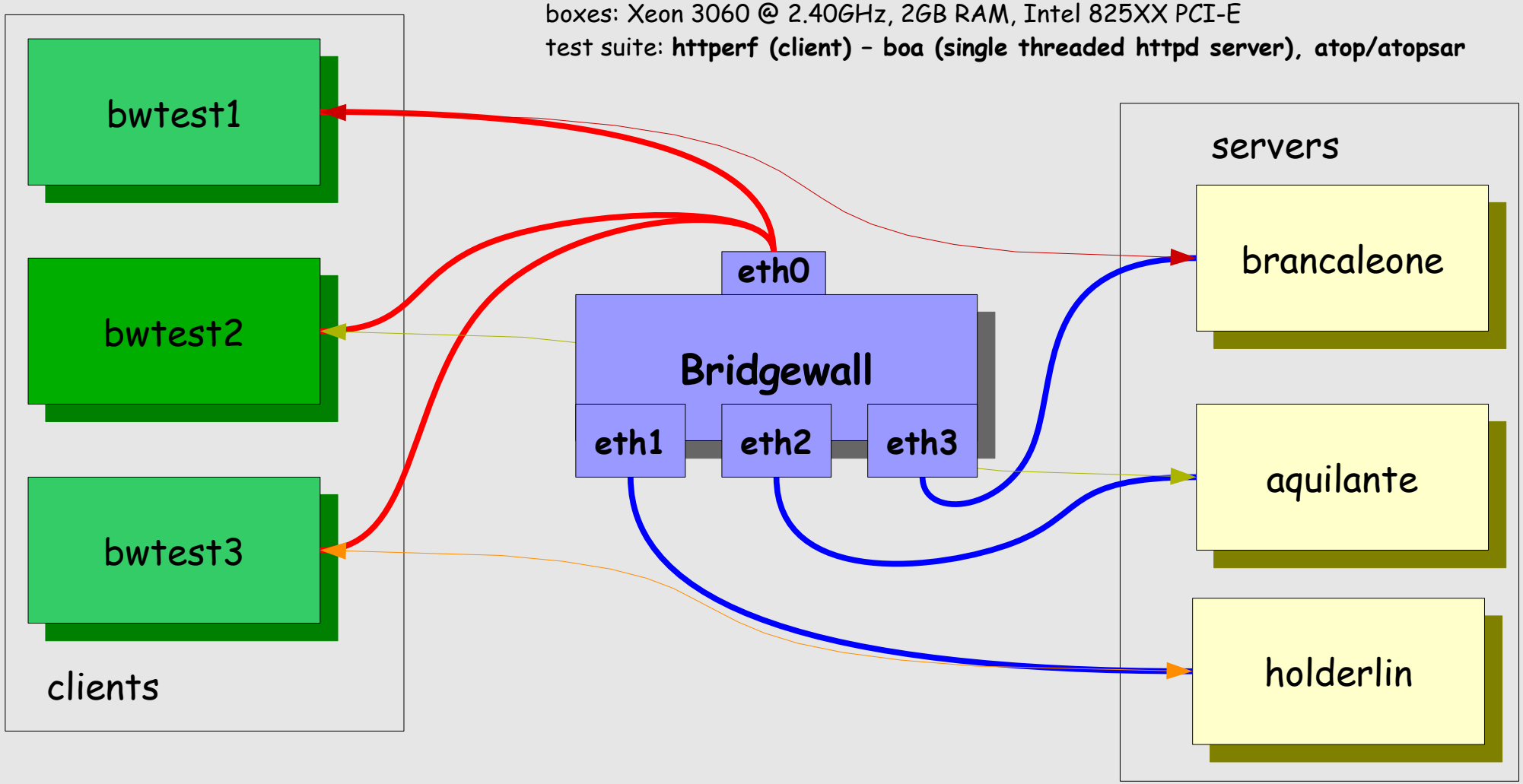


~75-100 µs

~50 µs

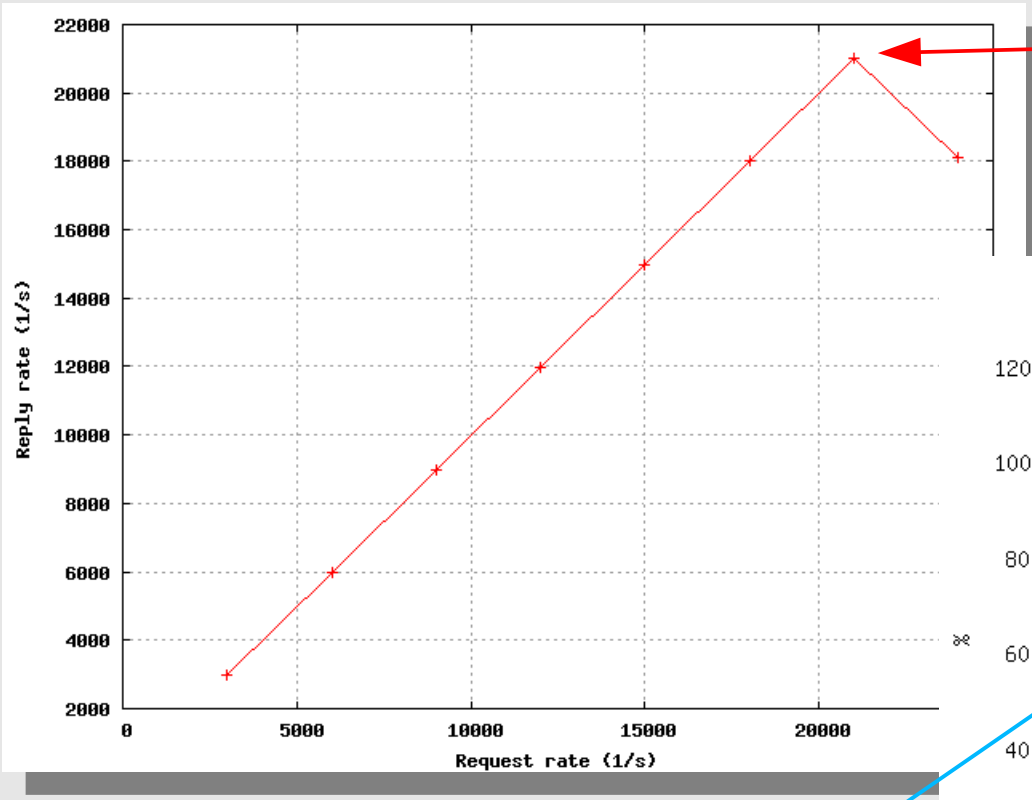
Connection rate - test setup

boxes: Xeon 3060 @ 2.40GHz, 2GB RAM, Intel 825XX PCI-E
test suite: **httperf (client) - boa (single threaded httpd server), atop/atopsar**

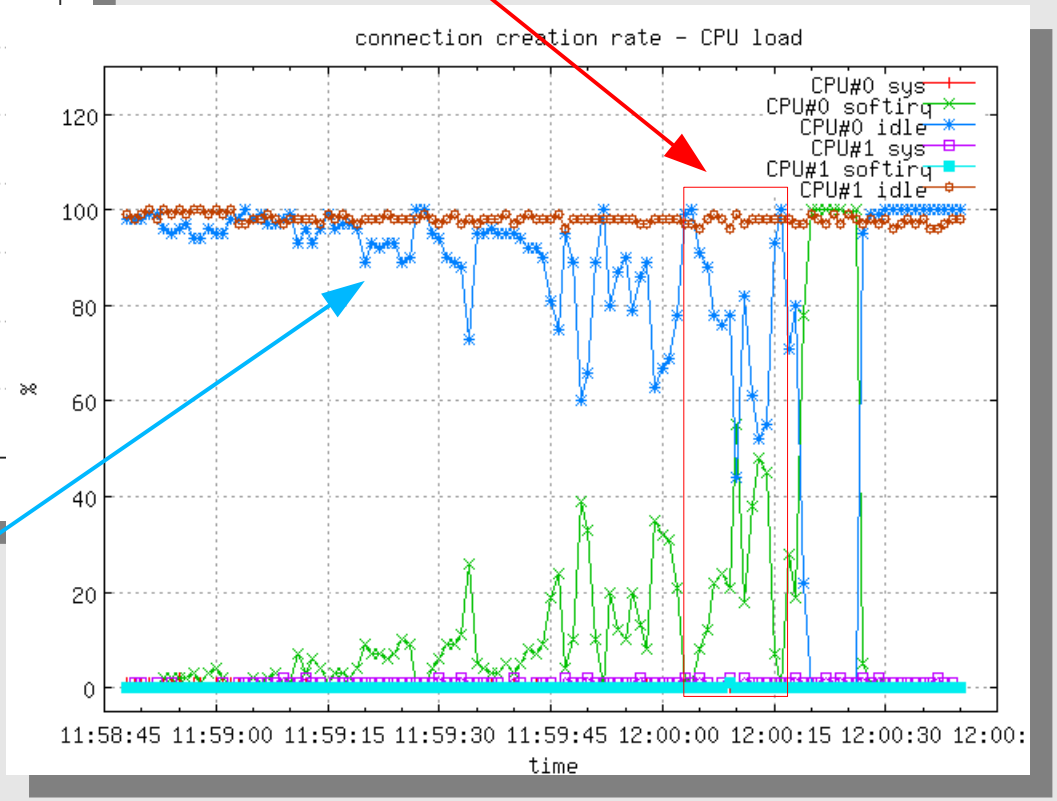


Connection rate - avg performance

[8 runs 1k->8k req/sec/client]



~21000 (new) cps @ CPU ~50%



CPU < ~10% up to ~ 10000 (new) cps

Prestazioni: soluzioni proprietarie vs LBW

	Cisco ASA 5520	Juniper NetScreen 208	SonicWALL PRO 4060	LBW
Max firewall throughput (Mbps)	450	375	300	930
Latency (microSec)	?	?	?	75 – 100
Max connections	280000	128000	524288	>262144
Max connections/second	9000	11500	5000	21000
Integrated ports	4x10/100/1000+1x10/100	8x10/100	6x10/100	*
Expansion slot	yes (4GE, CSC, AIP)	no	no	*
L2 transparent firewalling	yes	yes	yes	yes
Sec. contexts/zones – VFW	2 – 20 (licensed feature)	8 – 18 (upgrade)	yes (PRO1260: 24!!)	~
Stateful HA support	Act/Act & Act/Stb	Act/Act & Act/Stb	Act/Stb	Act/Stb?
Price	~10k?	~10k?	???	~1.5k

CSC: Content Security & Control
AIP: Advanced Inspection & Prevention

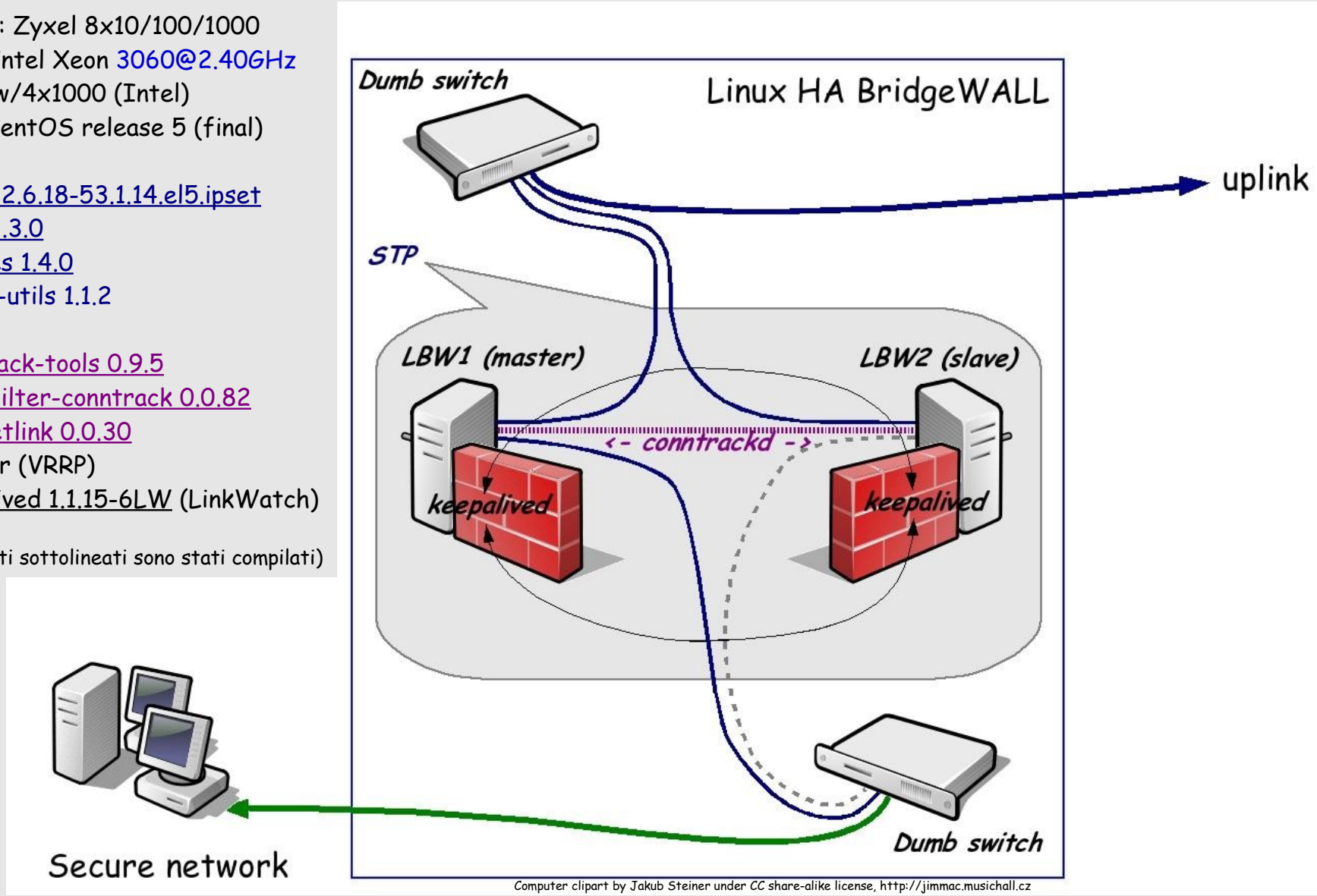
bridge + conntrack + netfilter (ipset)

Il Dinamico Duo: *HA Linux Bridgewall*

Dumb switch: Zyxel 8x10/100/1000
H/W LBW: Intel Xeon 3060@2.40GHz
w/4x1000 (Intel)
S/W LBW: CentOS release 5 (final)

- bridgewall
  - kernel 2.6.18-53.1.14.el5.ipset
  - ipset 2.3.0
  - iptables 1.4.0
  - bridge-utils 1.1.2
- conntrackd
  - conntrack-tools 0.9.5
  - libnetfilter-conntrack 0.0.82
  - libnfnetlink 0.0.30
- HA Manager (VRRP)
  - keepalived 1.1.15-6LW (LinkWatch)

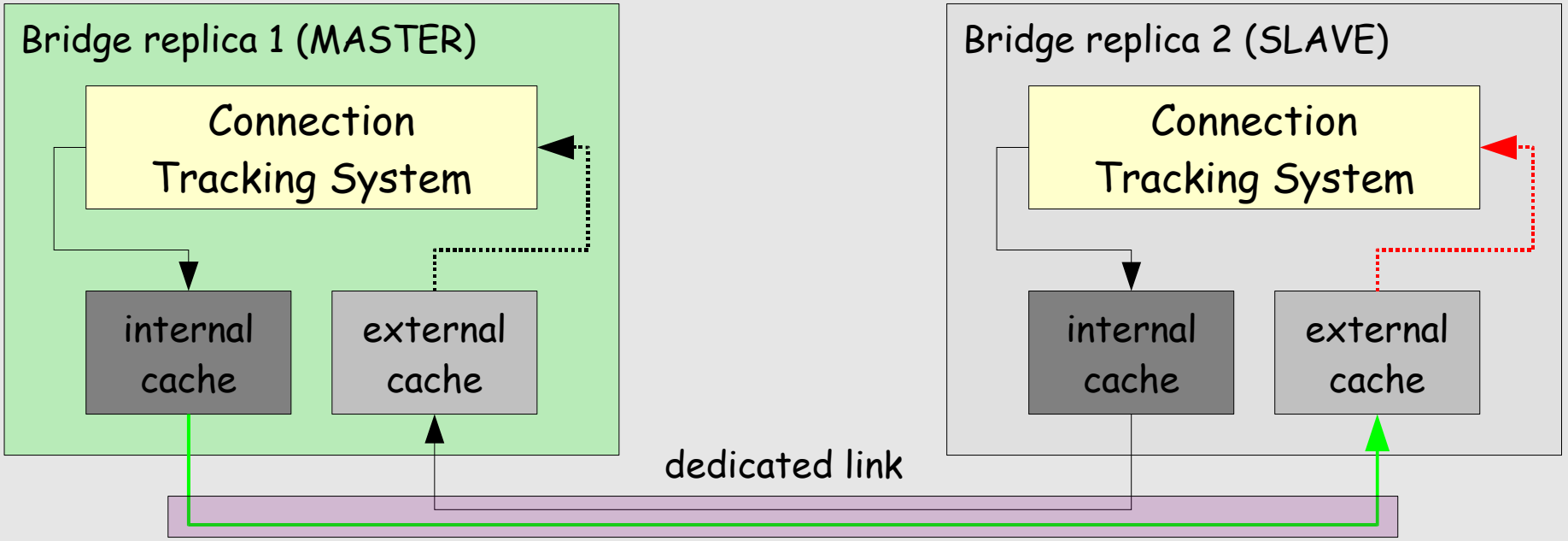
(tutti i pacchetti sottolineati sono stati compilati)



Computer clipart by Jakub Steiner under CC share-alike license, http://jimmac.musichall.cz

In teoria...:

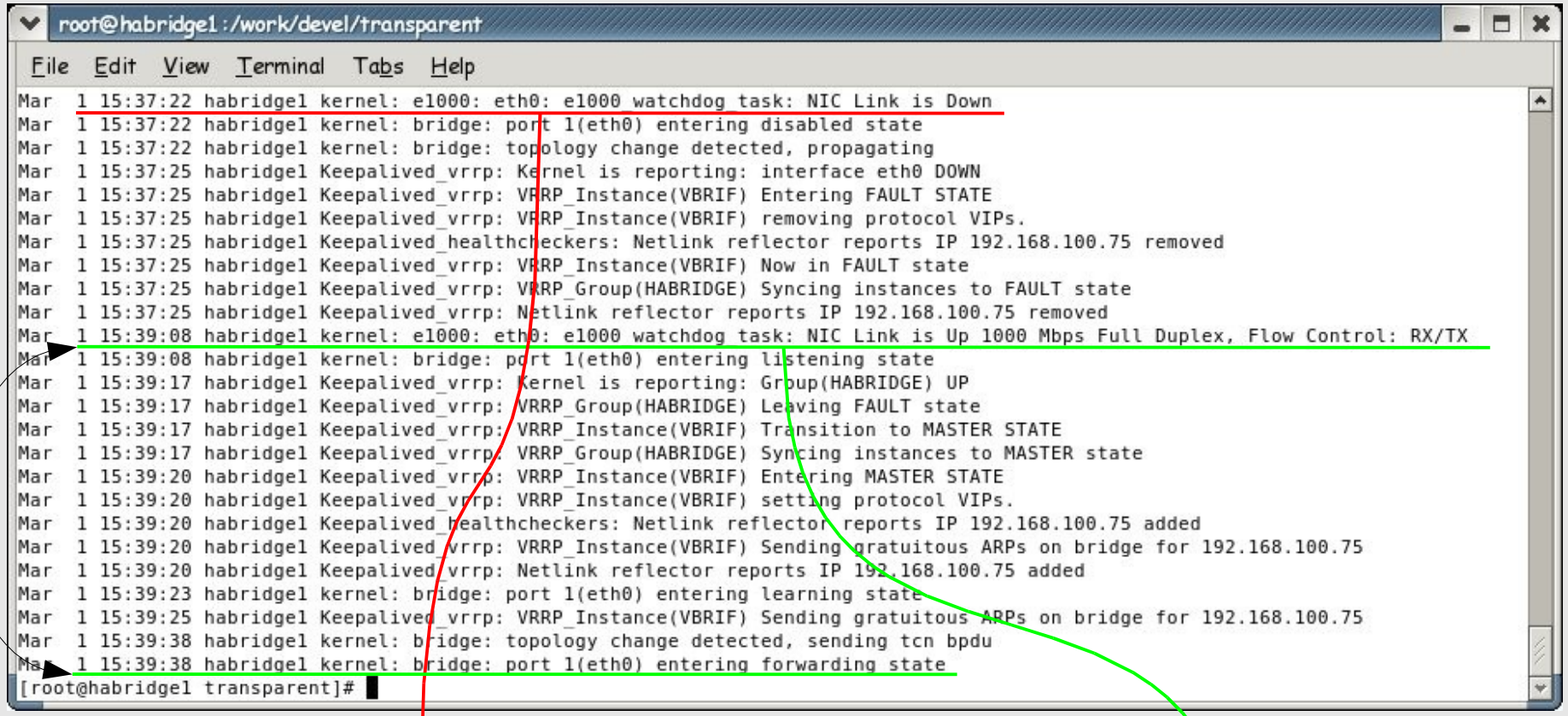
- STP garantisce il failover tra i due bridge *out of the box;*
- *conntrackd* sincronizza lo stato della *conntrack table* dei due *stateful firewall* (delle *external cache*, in realta') tramite il link dedicato (usa un protocollo di comunicazione simile a *netlink* su UDP);
- keepalived gestisce le transizioni SLAVE -> MASTER salvando la *external cache* nella *conntrack table*, trasferendo quindi la tabella delle connessioni attive al bridge di backup (lo slave).



... in pratica:

- In caso di fallimento del *master* (per esempio a causa della perdita del link su un'interfaccia, rilevata in modalita' asincrona da *keepalived* grazie alla feature di *LinkWatch*) lo *slave* ne prende il posto, ereditandone anche la tabella delle connessioni attive, che non vengono quindi rigettate (come *'NEW not syn'*). ***La transizione e' quindi trasparente ai clienti connessi*** *(da entrambi i lati del F/W ed al netto, ovviamente, del tempo di transizione).*
- Il *tempo di transizione* (vale a dire l'intervallo di failover) e' regolato essenzialmente dai parametri ***MAX AGE*** e ***FORWARD DELAY*** dello Spanning Tree; con i valori di default (20sec, 15 sec) e' di ***30-60 secondi***
  - a spanne: *2*ForwDly ≤ transition time ≤ 2*ForwDly+MaxAge*
- Limitazioni (solo se `'...--state INVALID -j DROP'`): *TCP window tracking* disabilitato (patch in arrivo):
  - **`# sysctl -w ip_conntrack_tcp_be_liberal=1`**

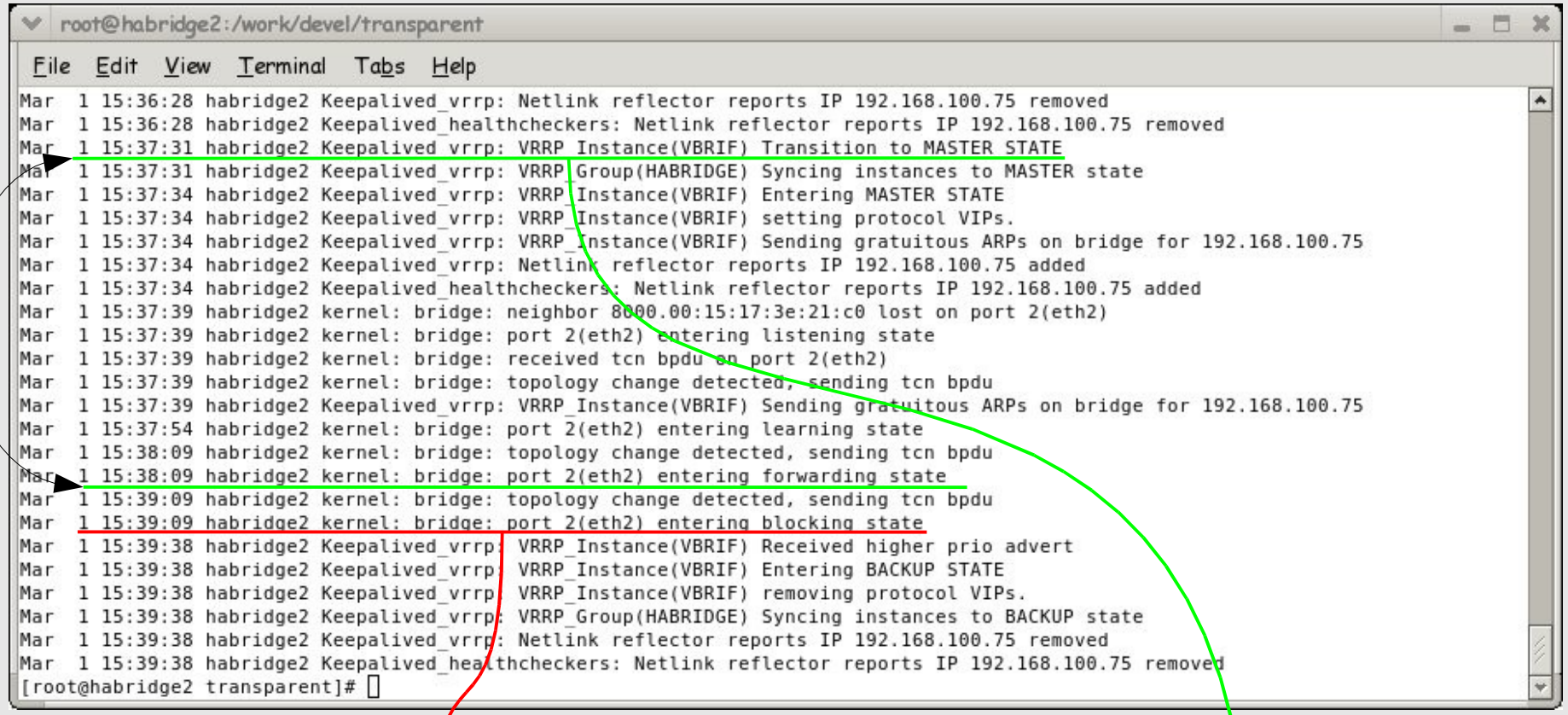
Un'istantanea...



MASTER (active) -> FAULT (standby)

FAULT (standby) -> MASTER (active)

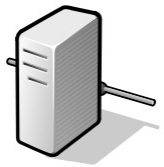
… e la stessa da un altro punto di vista:



MASTER (active) -> BACKUP (standby)

BACKUP (standby) -> MASTER (active)

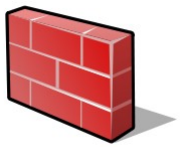
Le prestazioni: *max (new) connection rate*



Client

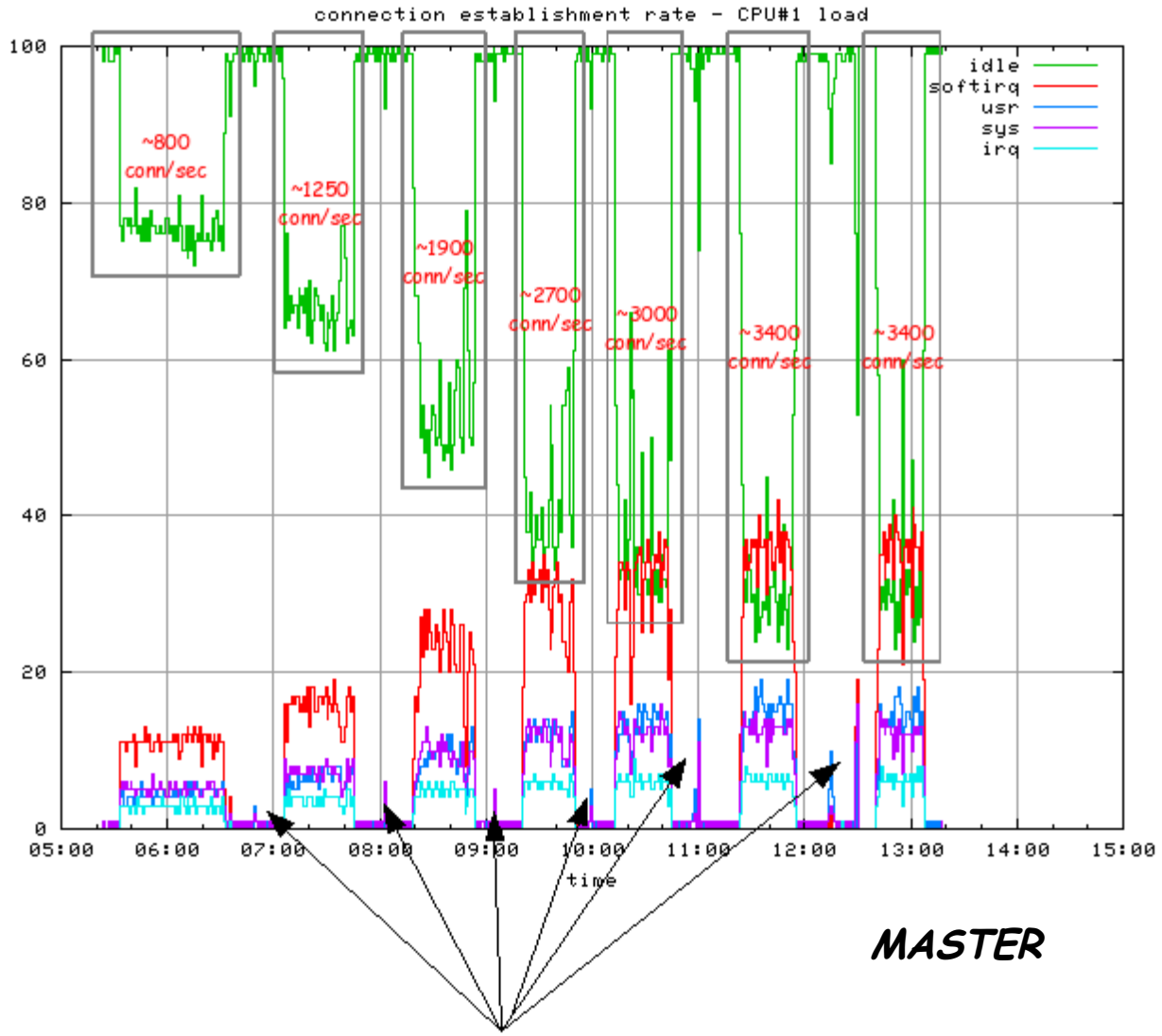
openload http://.../ N

N=1,2,..8 - number of simultaneous client to simulate



Linux HA BridgeWall w/filtering ON *(atop/atopsar)*

Server (apache)



MASTER

conntrack -F => flush connection tracking table

HA-LBW in sintesi

- conntrackd, come prevedibile, impone un carico non trascurabile sulla CPU del bridgewall: lo stesso H/W in configurazione non ridondata (senza conntrackd ma con le medesime regole di filtering) arriva a **~24000 conn/sec**; **~3500 conn/sec** e' in ogni caso un risultato rispettabile, ma i test effettuati non sono esaustivi (poche macchine a disposizione, ed una singola coppia client/server pare saturare *prima* del bridgewall); poiche' in questa applicazione la potenza di calcolo *sembra* il fattore determinante (*invece* - o *prima* - dell'efficienza di gestione degli IRQ) e' probabile che il limite rilevato sia una stima piuttosto conservativa (alcuni parametri di conntrackd sono poi ulteriormente ottimizzabili).
- throughput su singolo stream e latenza sono comparabili con quelli del BW non ridondato: in effetti non c'e' ragione perche' siano sensibilmente diversi, visto in questi casi conntrackd non entra in gioco (o quasi).
- il codice di bridging di Linux e' oramai *rock-stable*, e l'implementazione di STP compatibile con la (quasi?) totalita' dell'H/W di rete dedicato normalmente in uso (Cisco, Nortel, 3Com, Alcatel, HP, ...) .

Spunti, things to do...

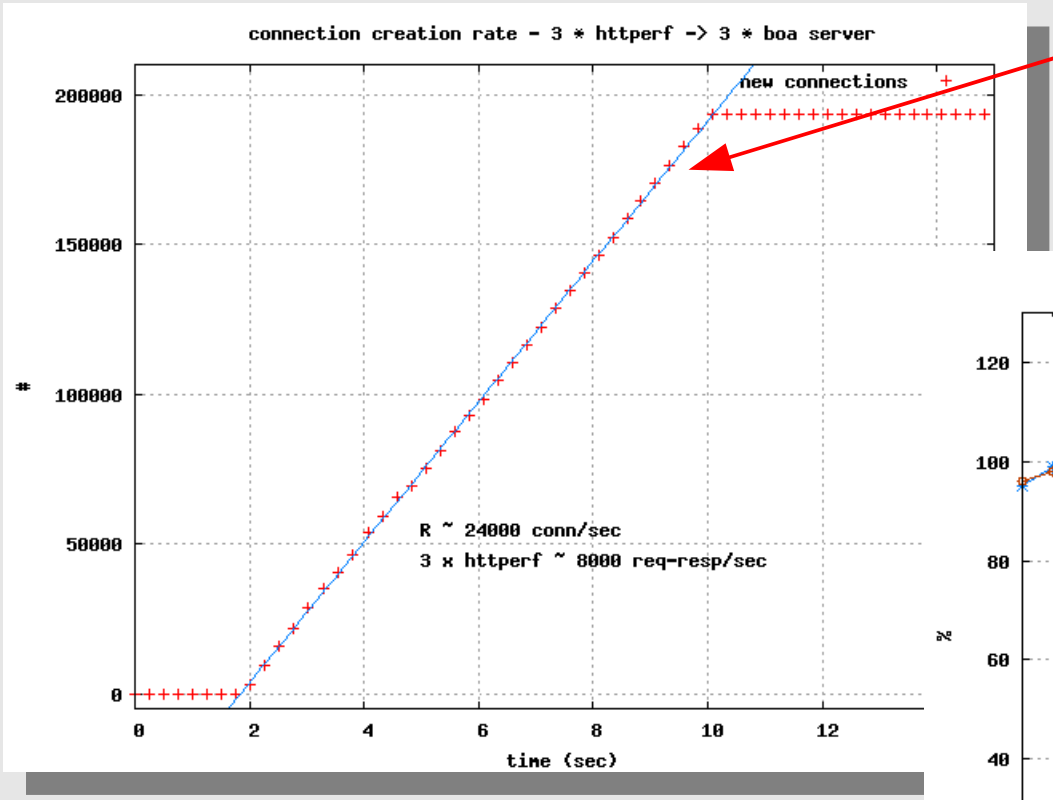
- Test in produzione 1 – FATTO! DNS/MAIL/WWW primari di sezione (server backup centrale) dietro HALBW: nessuno se n'e' accorto...
- A breve: test in produzione 2 - sostituzione proditoria Cisco+ACL con un LBW (se non funziona posso sempre dare la colpa al GARR...);
- ottimizzazione kernel/SO (diskless?)/regole...
- ... NIC 10G?
- Bridgewall virtuali?
- Dual core -> Quad core (IRQ distribuiti sui core)
- Opteron (per-cpu memory controller)
- Intel quad-port adapter (4+4+... ports bridgewall?)
- Policy dinamiche: integrazione IDS – BW w/dynamic ipsets; rilevamento/blocco *anche* attacchi dall'interno
- Traffic shaping (l7 etc.);
- Nf-HIPAC (drop-in replacement di iptables con algoritmo di classification/search/match estremamente veloce ed efficiente);

Conclusioni (in corso d'opera)

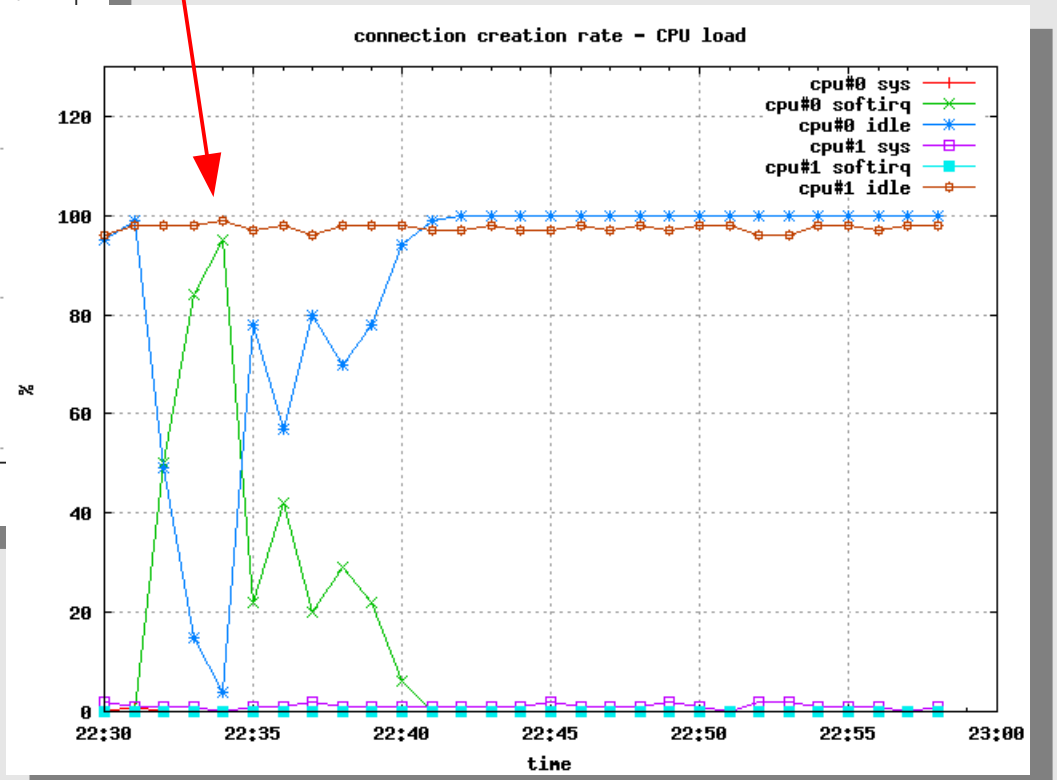
- FUNZIONA! Un LBW/HA-LBW sembra potersi integrare facilmente e trasparentemente (senza richiedere alcuna operazione di riconfigurazione topologica o degli schemi di naming e routing) in strutture di qualsiasi complessita' per isolare/proteggere con la granularita' desiderata servizi fondamentali o gruppi di macchine, o definire in una LAN perimetri di sicurezza multipli ed indipendenti (cluster etc.);
- Queste soluzioni open-source garantiscono buone prestazioni (anche su H/W non particolarmente esotico), stabilita', flessibilita' ed alta affidabilita' (ridondanza) a costi molto contenuti (a prezzo, ovviamente, di un po' di lavoro);
- Il confronto * sulla carta * con soluzioni proprietarie sembra (moderatamente) impietoso: qualche sponsor munifico desidera lavare l'onta?

Connection rate - peak performance

[single shot - 8k req/sec/client]



peak: ~24000 (new) cps @ CPU ~ 100%



"Netfilter Performance Testing"

Kadlecsik & Pasztor

- H/W: Dual Opteron @ 2.2GHz, 4GB RAM, Intel 82546EB Gbit Ethernet (dual), kernel 2.6.11 with NAPI
- Testbed: http request-reply test, 160 clients (httperf) / 160 servers (boa) [=> 3500000 max concurrent connections.]
- iptables /ipset/nf-hipac (with an ACCEPT rule after N=16,32,...,16384 non matching rules) comparison: iptables doesn't scale (the more the rules, the worse the performance); *ipset & nf-hipac perfom almost indifferently with regard of the number of rules (ipset is a tiny bit better than nf-hipac).*
- *netfilter (iptables) performances (ACCEPT rule after 10 non matching rules):*

	ROUTING	CONNTRACK	CONNTRACK/NF
cps	55000	25000	25000
pps	700000	330000-340000	300000

iptables vs ipset/nf-hipac

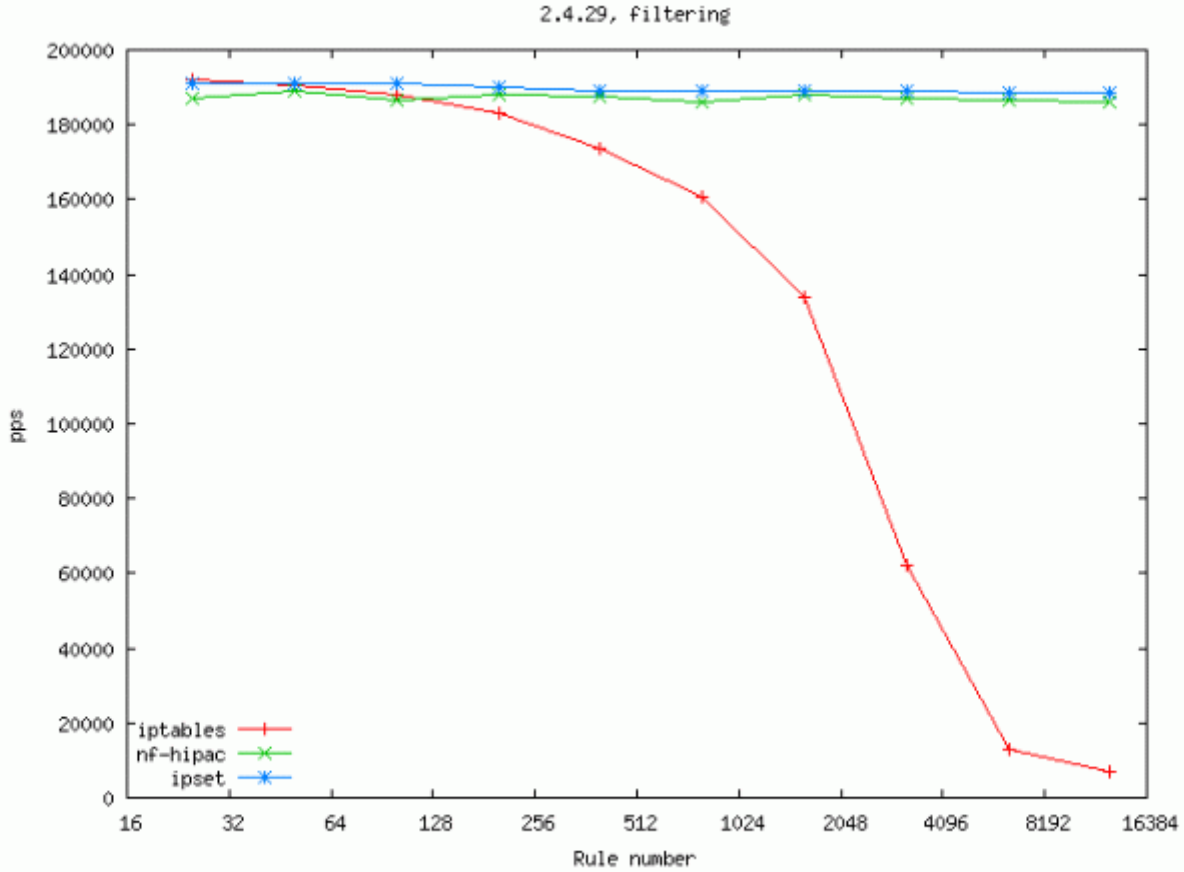
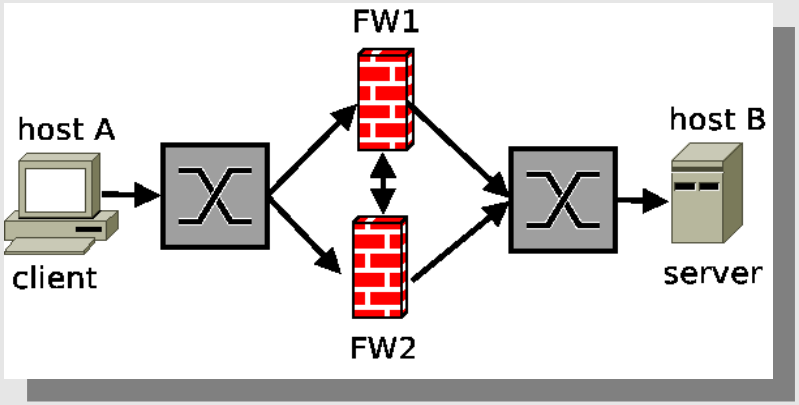


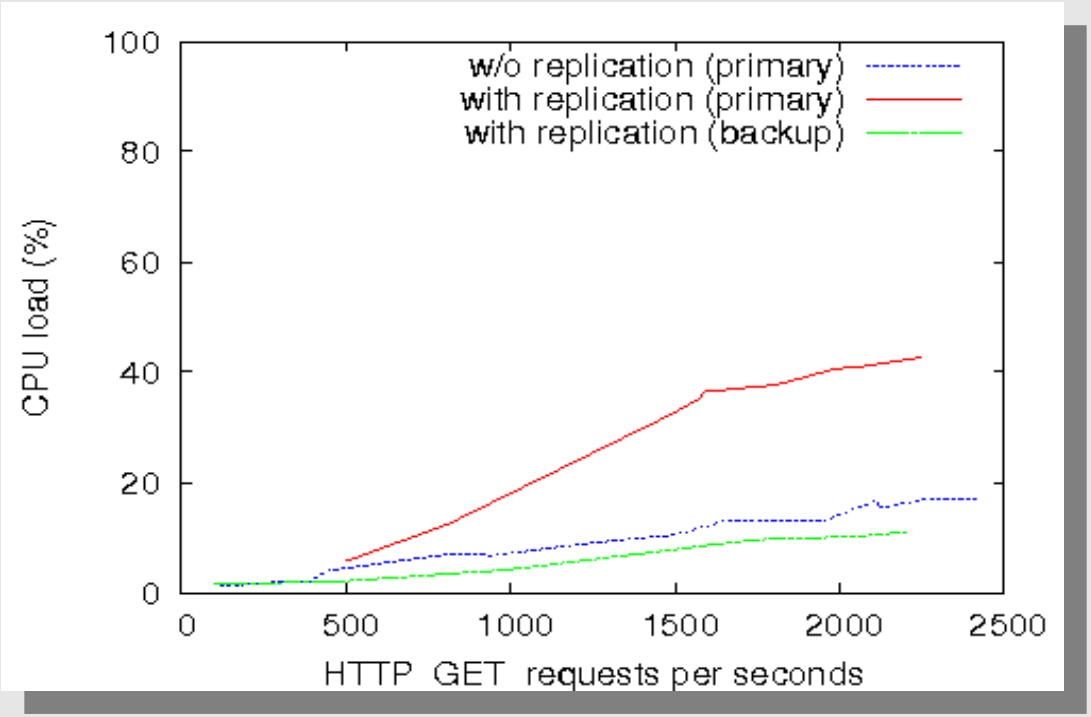
Figure 16 iptables, nf-hipac, ipset

HA Bridgewall: conntrackd

- From *"Conntrack-tools: HA for stateful Linux firewalls"*, by Pablo Neira



HP Proliant 145G2, 2.2GHz



Auth. failover <u>without established connections loss</u>

- Primary/Backup, *Multiprimary*
- protocols: alarm-based, reliable UDP

conntrackd: serve veramente?

Una *feature* assai poco nota di iptables e' quella che prevede di considerare in stato **NEW** anche i pacchetti TCP con il bit di **SYN unset**. Se questo da un lato permette di implementare configurazioni ridondate senza alcun meccanismo di sincronizzazione delle connection table, dall'altro espone al rischio di accettare anche connessioni sospette o maliziose o, nella stragrande maggioranza dei casi, manifestamente pericolose (quasi tutte quelle che non iniziano con il 3-way handshake lo sono: scan, probe, attacchi, etc.). Se un F/W implementa (come auspicabile...) regole di protezione da attacchi di questo tipo:

```
... -p TCP -m state --state RELATED,ESTABLISHED -j ACCEPT
... -p TCP ! --syn -m state --state NEW -j DROP
...
```

allora conntrackd e keepalived (o un qualsiasi altro HA manager) non sono una complicazione inutile (come potrebbero sembrare alla luce della citata *feature* di iptables...) ma ingredienti necessari per la realizzazione di uno stateful F/W ridondato.