Web Load Balancing on a Budget

Pain

- Hosting 60+ websites
- Single web server
  - Redundant subsystems (disk, power)
- SPOF
- Inconvenient maintenance windows
  - Clients
  - MY TEAM!

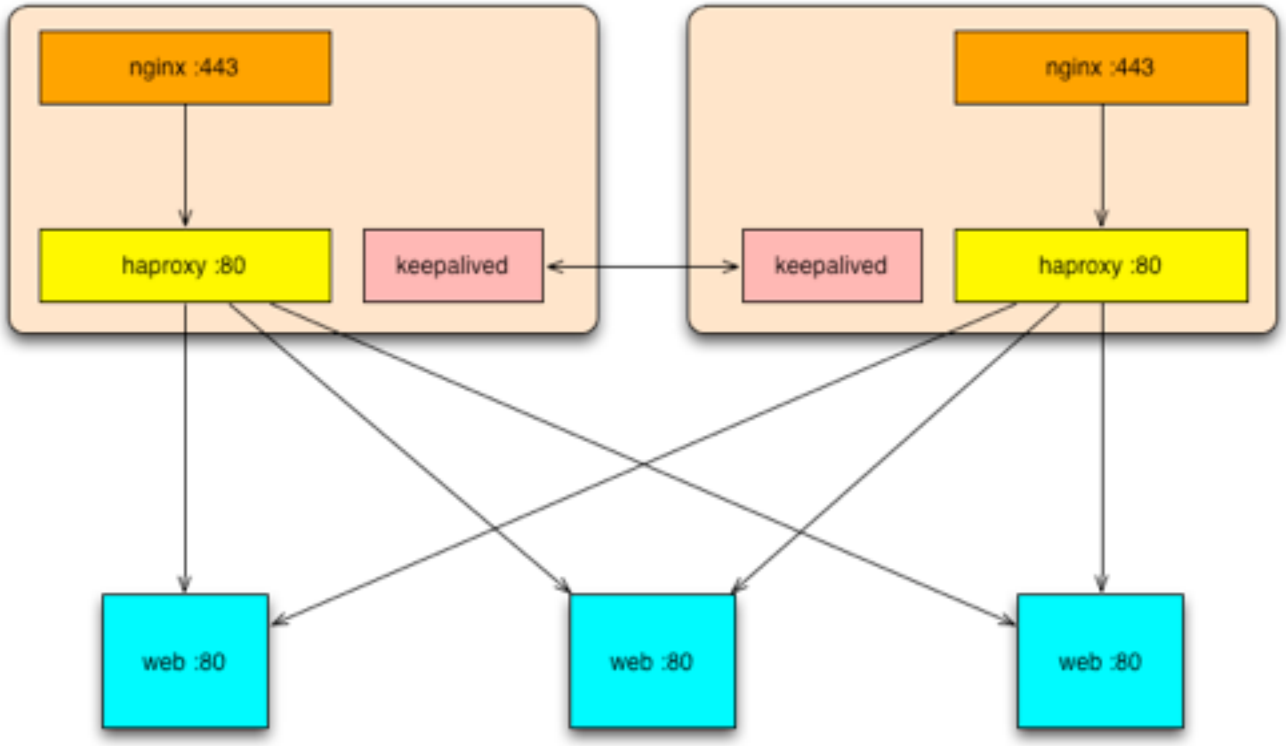
Scope

- Simple. Availability.
  - Minimize/mitigate downtime
    - Outages
    - Planned maintenance
- Session state persistence (for failed web backends) was not required

Stack

- Vmware ESXi (all hosts)
- CentOS (load balancers)
  - Nginx (SSL termination)
  - Haproxy (web load balancing)
  - Keepalived (VRRP)
- Windows Server 2008 (IIS)

Design


nginx :443
nginx :443
haproxy :80
keepalived
keepalived
haproxy :80
web :80
web :80
web :80

```
# Let us bind to addresses and ports that may not be “real” (nginx and haproxy need this)
net.ipv4.ip_nonlocal_bind = 1

# Reuse sockets that are in a TIME_WAIT state so we don’t exhaust resources
net.ipv4.tcp_tw_reuse = 1

# Give us a larger useable port range (default: 32768 61000)
net.ipv4.ip_local_port_range = 1024 65535

# Close TCP FIN connections faster to help lower resources used by the network stack
net.ipv4.tcp_fin_timeout = 30

# Increase the number of slots that iptables has for tracking connections
net.ipv4.netfilter.ip_conntrack_max = 131072

# Decrease the time that iptables waits to close sockets in TIME_WAIT (def: 120)
net.ipv4.netfilter.ip_conntrack_tcp_timeout_time_wait = 30
```

```
http {
    server_tokens off;            # make the hackers work for it

    server_name_in_redirect off;  # use requested Host header

    proxy_read_timeout 1500;      # 25 min; required for long-running reports

    upstream www.example.com {
        server 192.168.1.2:80;
    }

    server {
        listen 192.168.1.2:443;
        access_log /var/log/nginx/access.log;
        ssl on; # go!
        ssl_certificate /usr/local/ssl/wildcard.example.com.cert;
        ssl_certificate_key /usr/local/ssl/wildcard.example.com.key;
        ssl_prefer_server_ciphers on;       # prefer SSLv3 and TLSv1 ciphers
        ssl_ciphers HIGH:+MEDIUM:!ADH:!MD5; # use 128-bit and higher ciphers; exclude ADH, MD5
        ssl_protocols TLSv1 SSLv3;          # guarantee only TLSv1 and SSLv3 protocols

        location / {    # match all HTTP requests
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header secureCookie YES;
            proxy_pass http://www.example.com;
        }
    }

}
```

```
global
    maxconn 32500
    nbproc 1

defaults
    log global           # log all proxy instances
    option httplog       # verbose HTTP logging
    mode http            # layer 7 proxy

    retries 2
    timeout connect 2s   # 2 retries x 2s timeout = 4s until haproxy looks for new backend
    option redispatch    # redispatch requests originally sent to downed backend servers

    timeout client 5s
    timeout server 1500s # 25 minutes, for long-running reports (matches nginx config)

    balance roundrobin   # also leastconn

    option forwardfor    # X-Forward-For header
    option httpclose     # force HTTP connections closed so all connections are logged

listen statsWWW 192.168.1.1:8080
    stats enable         # enable the haproxy stats page (uri-stem = /haproxy?stats)
    stats auth god:sex
    stats refresh 120s

listen www.example.com
    bind 192.168.1.2:80
    acl acl_port_80 dst_port eq 80
    acl acl_secure hdr(secureCookie) YES
    acl acl_sourceLocal src 192.168.1.2
    acl acl_excludeCSS url_dir -i CSS
    redirect location https://www.example.com/sslRedirect.asp code 301 if acl_port_80 !acl_secure !
acl_sourceLocal !acl_excludeCSS
    cookie SERVERID insert indirect nocache
    server www1 192.168.1.3 cookie www1 weight 1 check inter 1s fall 3 rise 2
    server www2 192.168.1.4 cookie www2 weight 1 check inter 1s fall 3 rise 2
    server www3 192.168.1.5 cookie www3 weight 1 check inter 1s fall 3 rise 2
    option httpchk GET /ping.htm HTTP/1.1\r\nHost:www.example.com   # health check
    capture request header X-Forwarded-For len 15
```

```
global_defs {
    lvs_id lb1
}

vrrp_sync_group virtualGroup1 {
    group {
        virtualInstance1
    }
}

vrrp_instance virtualInstance1 {
    interface eth0
    advert_int 1         # advertise interval in seconds
    state MASTER         # either MASTER or BACKUP
    virtual_router_id 1 # MUST be between 1 and 255
    priority 101         # 101 on MASTER, 100 on BACKUP
    authentication {
        auth_type AH
        auth_pass "sexdrugsandrockandroll"
    }
    virtual_ipaddress { # only 20 IPs per block, blah
        192.168.1.1/24 brd 192.168.1.255 dev eth0
        192.168.1.2/24 brd 192.168.1.255 dev eth0
        192.168.1.3/24 brd 192.168.1.255 dev eth0
        192.168.1.4/24 brd 192.168.1.255 dev eth0
        ...
        192.168.1.19/24 brd 192.168.1.255 dev eth0
    }
}
```

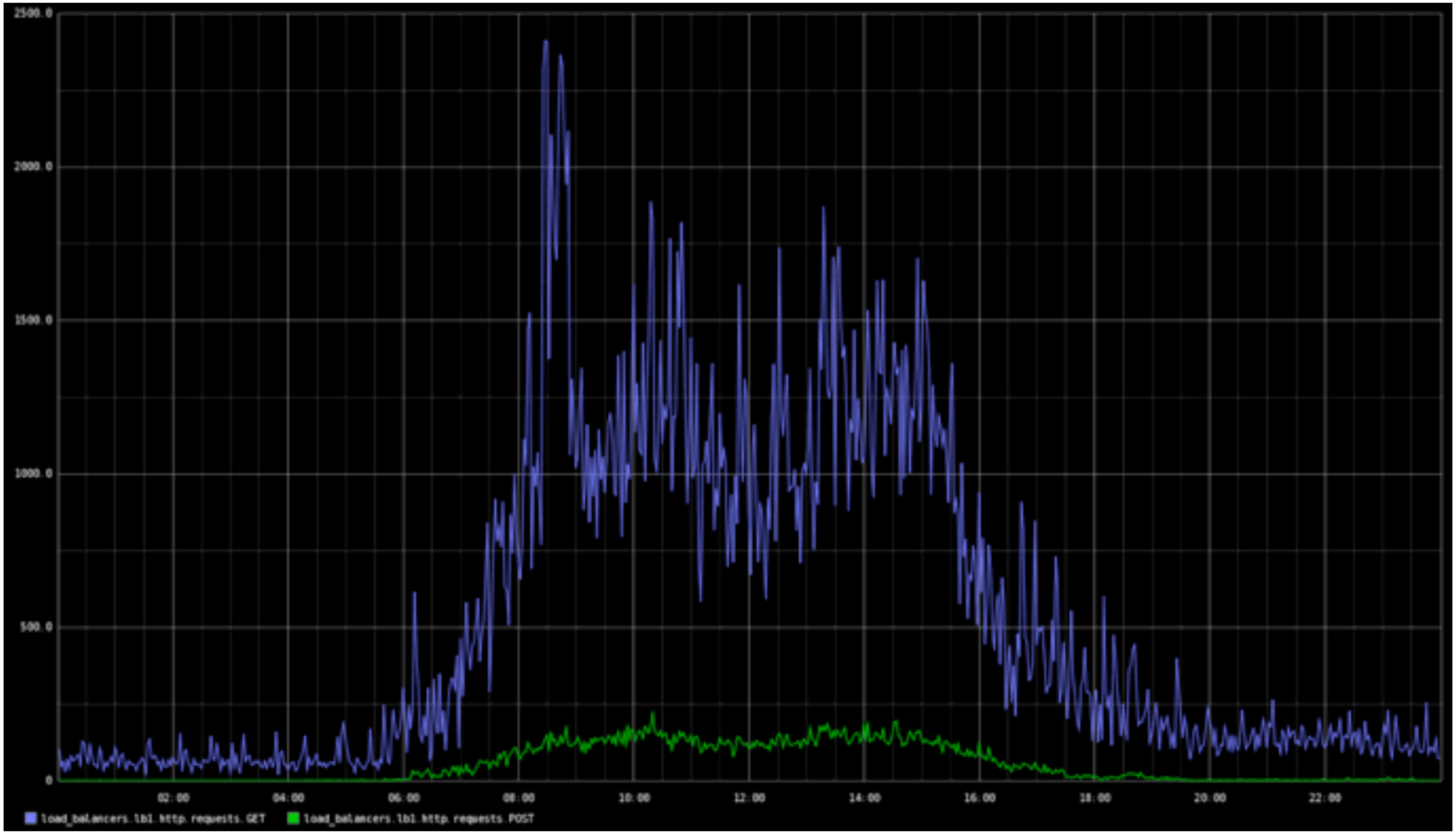
Pitfalls

- VMware and TSO
  - When jumbo frames go bad
  - HTTP 500 errors
  - Web servers “offline” for 20-30 minutes
  - Disable TSO
    - http://www.ryanfrantz.com/2011/02/03/tcp-segmentation-offload/

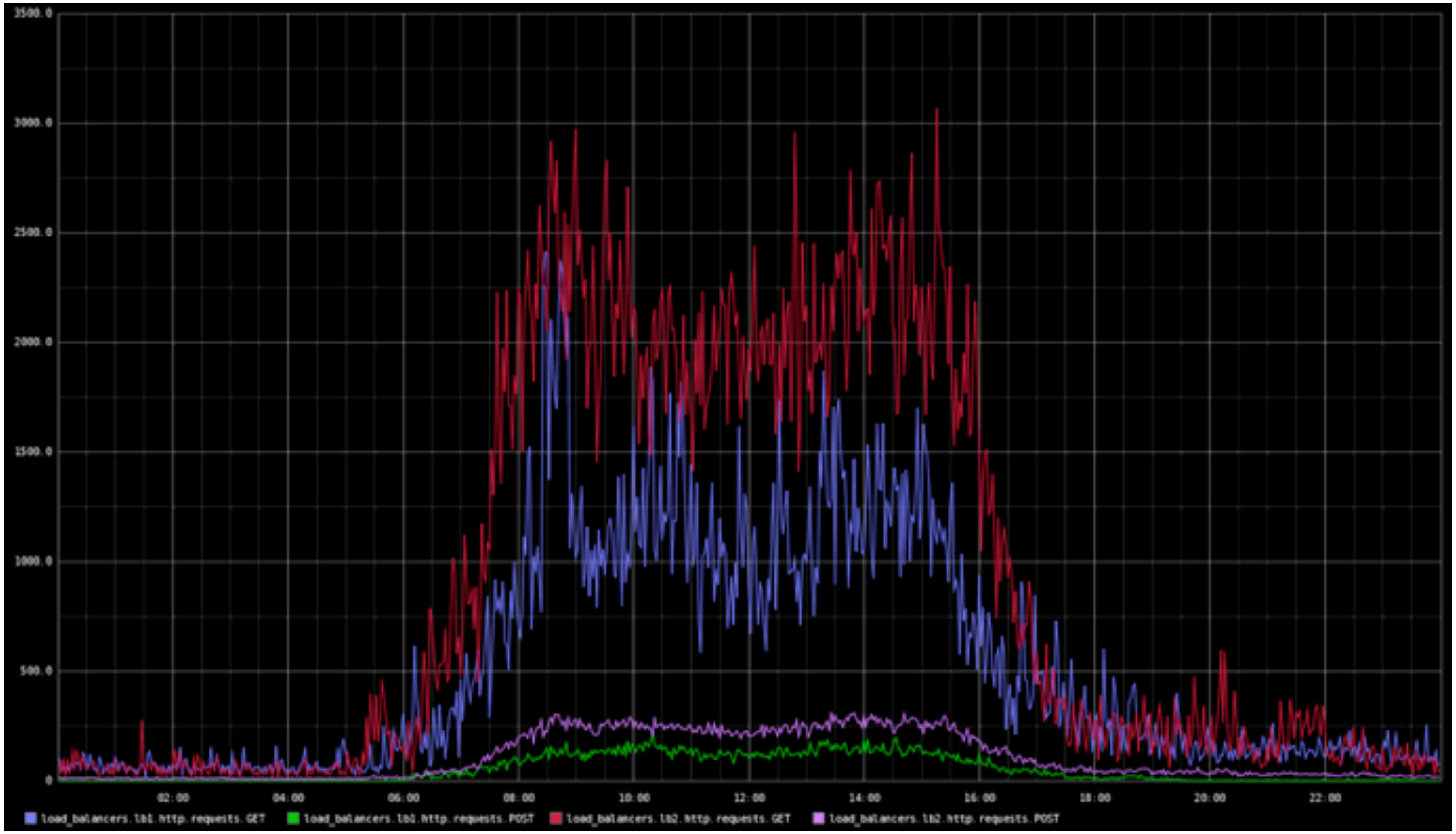
What’s Happening?

- No visibility into performance
  - Request volume (nginx/haproxy/IIS)
  - Seasonality
- Health (beyond up/down)
  - keepalived

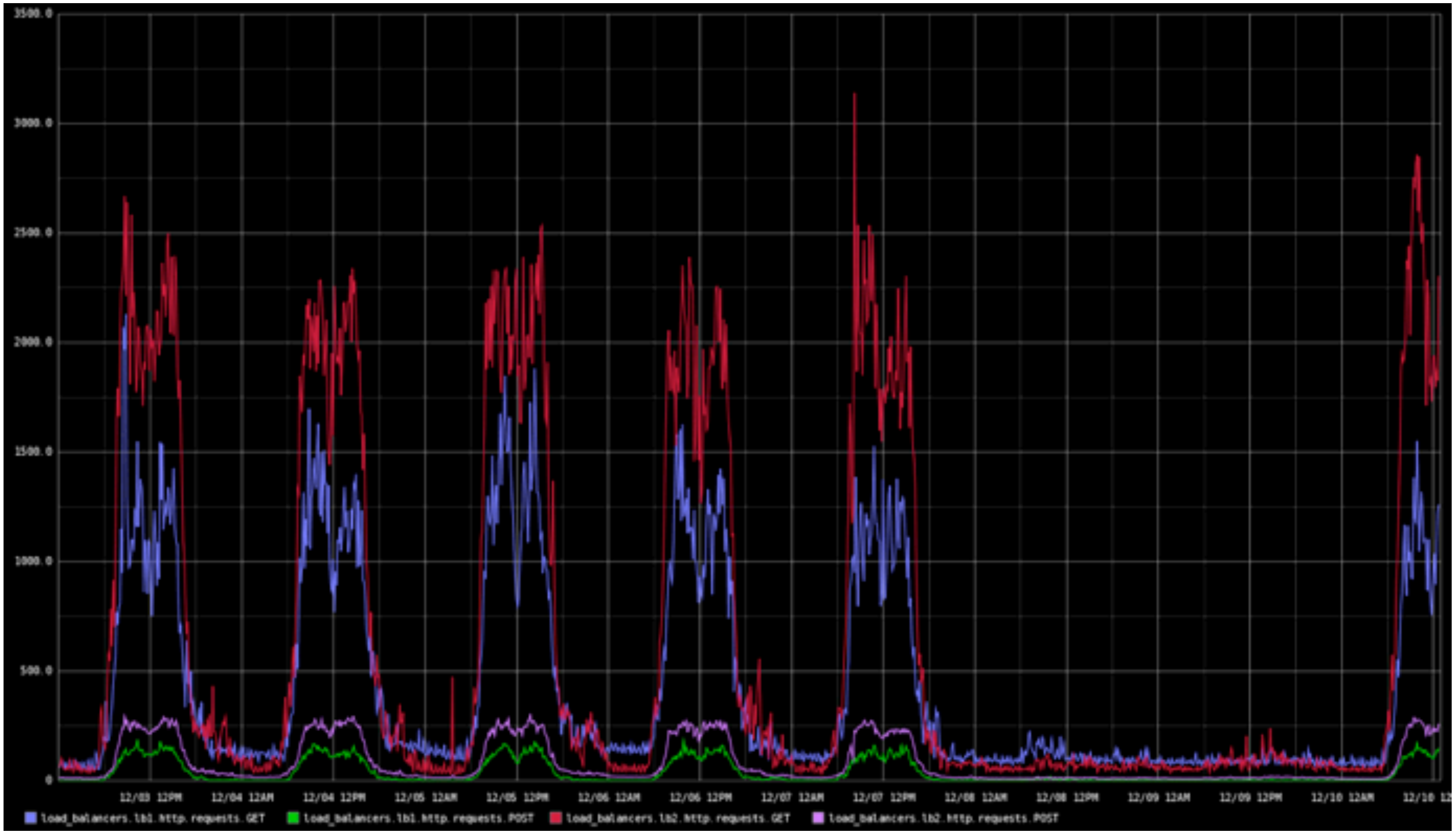
HTTP Traffic: One Load Balancer



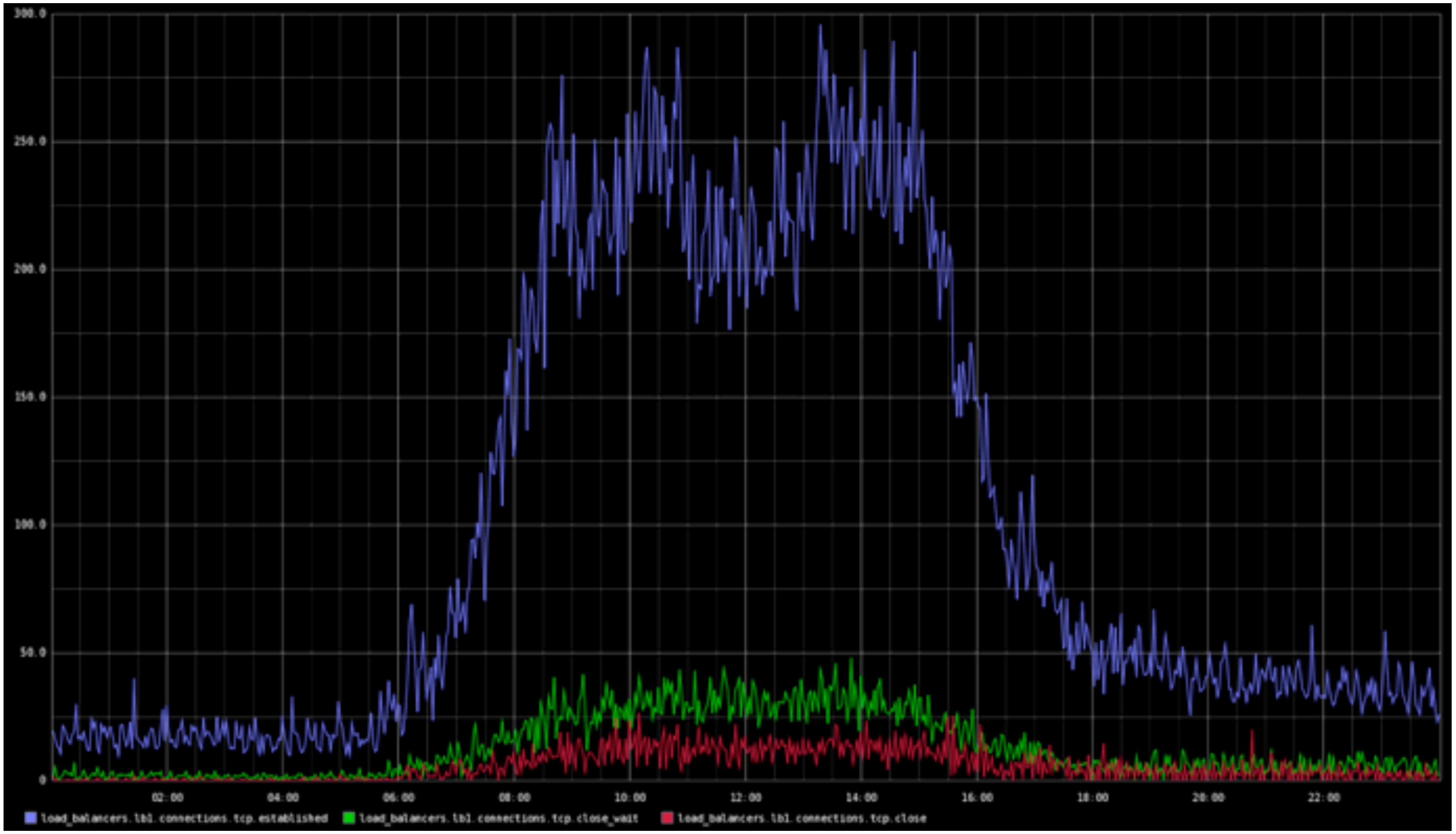
HTTP Traffic: Two Load Balancers



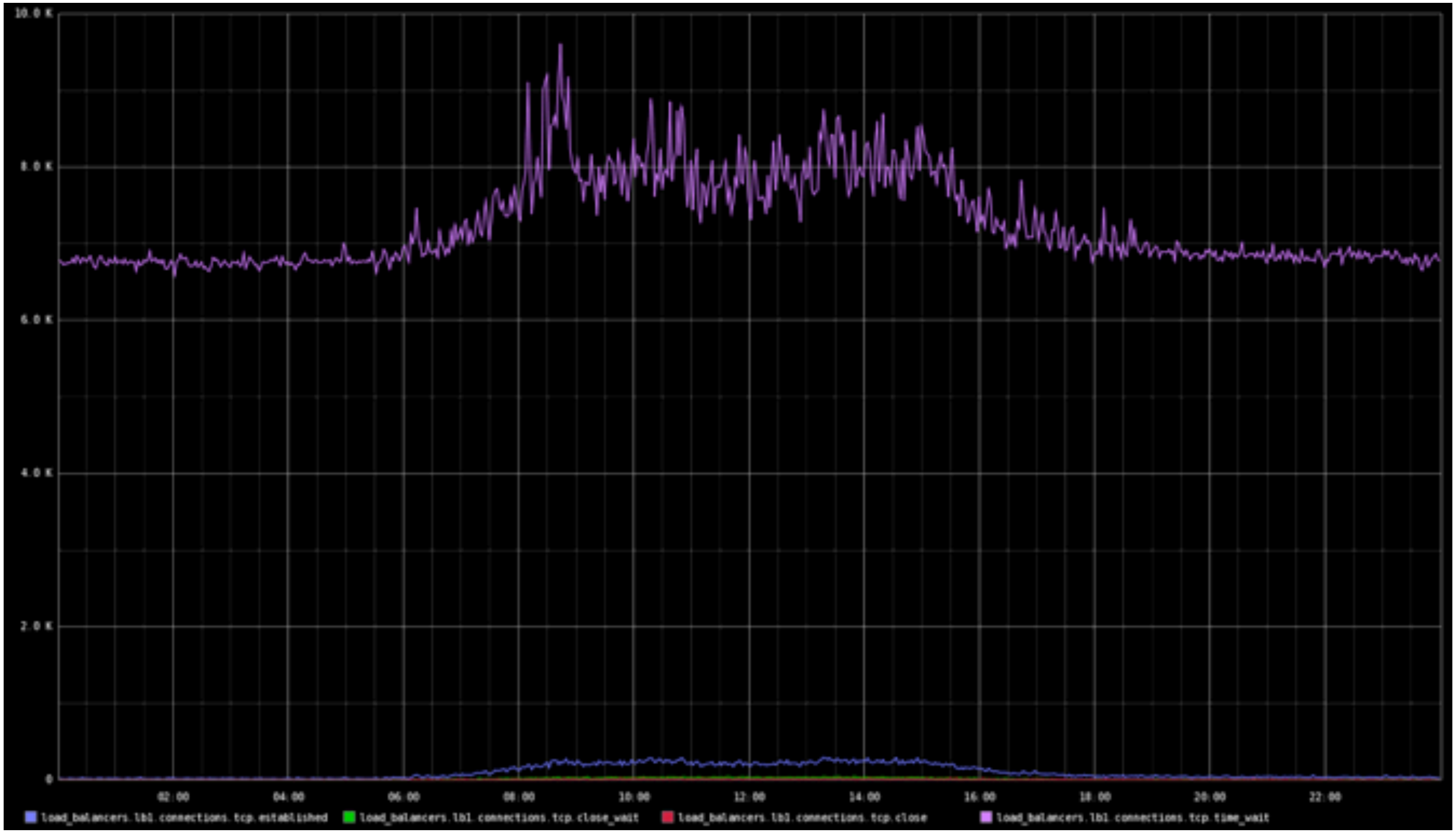
HTTP Traffic: One Week



TCP Connections



TCP Connections: + TIME_WAIT



Thank You for Attending LOPSA-East '13

Please fill out the Trainer Evaluation
http://lopsa-east.org/2013/training-survey

Rate LOPSA-East '13
http://www.lopsa-east.org/2013/rate-lopsa-east-13

LOPSA
the league of professional system administrators

@Ryan_Frantz
rfrantz@etsy.com
github.com/RyanFrantz
www.ryanfrantz.com