Transparent HTTP with Apache Traffic Server

networkGEOGRAPHICS

Speaker

- Alan M. Carroll, PMC
  - Started working on Traffic Server in summer 2010.
  - Implemented
    - Transparency
    - IPv6
    - Other stuff
  - Works for Network Geographics
    - Provides ATS and other development services

Goal

- A starting point for deploying ATS as a transparent HTTP proxy
- Provide sample scripts
- Help you understand what the commands in the scripts are intended to accomplish
- Guide to useful tools

Outline

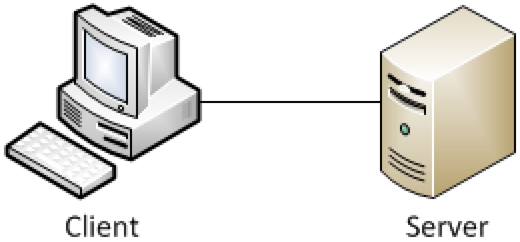
- Basic theory
- Drill down to increasing detail for deployment
- Trouble shooting
- Commands not discussed directly
  - Not really helpful
  - You can look ahead to appendix scripts and ask questions on them that relate to slides

What are we trying to do?

BASIC THEORY

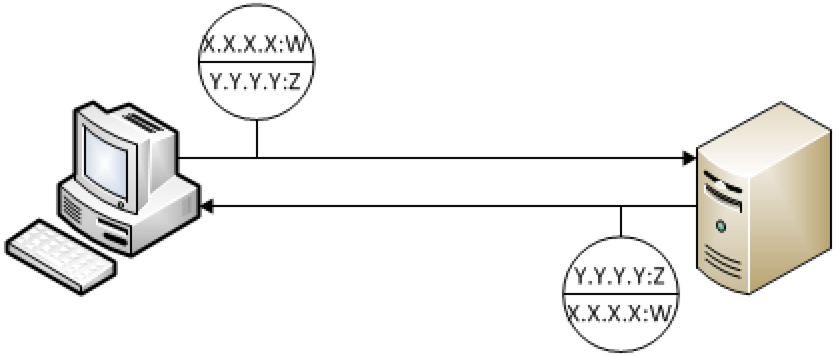
Transparency Basics

- Quick review to avoid misunderstandings
- Use standard client / server terminology
  - Client initiates connection
  - Server receives connection
  - Users think this is how it works:
  - Transparency contributes to this illusion in the presence of proxies



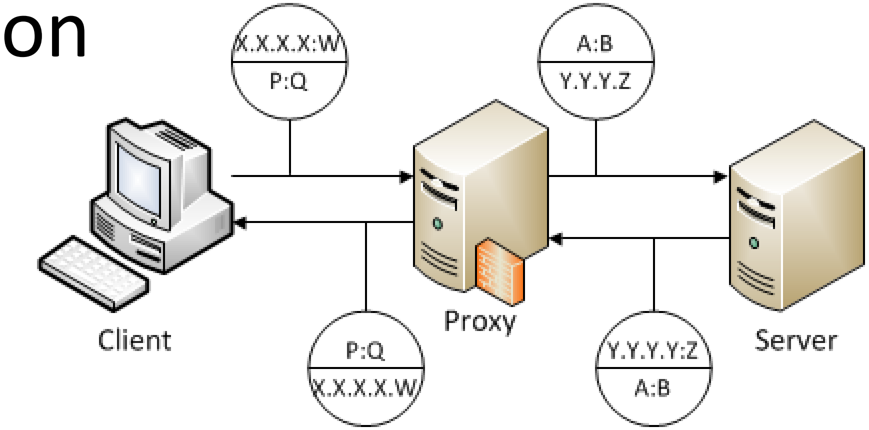
Two Halves make a whole

- Slightly more sophisticated view
  - Two half connections
  - Identified by 5-tuple, but we'll presume TCP
  - So it's a 4-tuple for us
    - Local IP address:port, Remote IP address:port
  - "Local" and "Remote" are viewpoint based



Proxying

- You want to modify network traffic
  - Use a proxy to intercept connections
  - If just monitor and track, not modify, use a sniffer. Much easier.
- Basic proxy operation



Proxying makes two

- A proxied connection is really two connections
  - Two <u>independent</u> connections
    - Client <-> Proxy
    - Proxy <-> Server
  - They only look related because the proxy is clever
- Proxy address:port pairs (P:Q and A:B)
  - Proxy types are simply terms for how these pairs are selected

Proxy Types

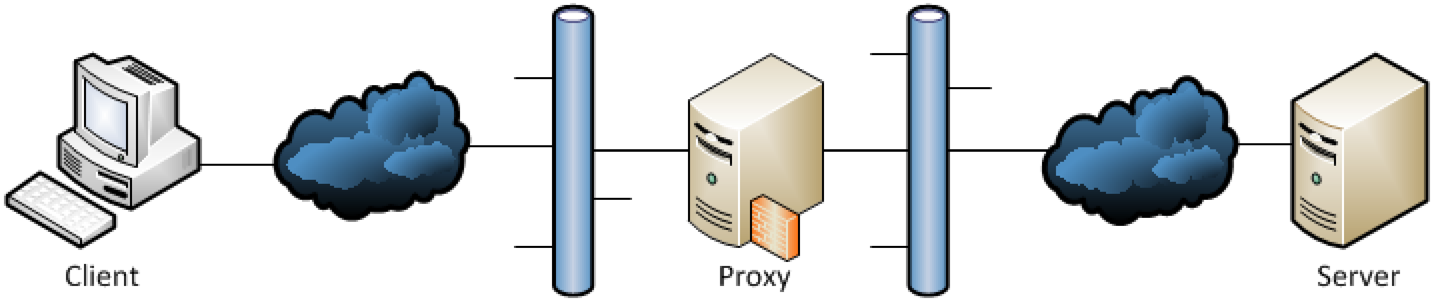
Address used by proxy	Client connects to Proxy Address	Client connects to Server Address
Server accepts from Proxy Address	Explicit Proxy (Not transparent)	Inbound transparent
Server accepts from Client Address	Outbound transparent	Fully transparent

HTTP Proxies

- ATS is an HTTP proxy/cache
- To modify traffic proxy must understand traffic
  - Other traffic must be handled as opaque data
- ATS understands HTTP
  - Can modify/cache headers as well as content
  - Can rely on data present in HTTP headers
- ATS does not understand HTML
  - But your plugin can

Putting the Proxy in your network

- Proxy goes between the client and the server



Proxy Topologies

- Routed
  - Proxy is between different networks
- Bridged
  - Same network on both sides of the proxy
- WCCP (Cisco routers only)
  - Router intercepts for proxy elsewhere
  - Enables pass through failover
  - IPv4 only

Why Transparency

- Transparency makes a proxied topology look like the simple client / server topology
- Should you use transparency?
  - From whom do you want to hide the proxy?
    - Hide from clients?
    - Hide from server?
- Pick from four basic types of transparency

Proxy Types

Address used by proxy	Client connects to Proxy Address	Client connects to Server Address
Server accepts from Proxy Address	Explicit Proxy (Not hidden)	Inbound transparent (hidden from clients)
Server accepts from Client Address	Outbound transparent (hidden from servers)	Fully transparent (hidden from clients and servers)

Examples / Use Cases

- Explicit proxy
  - The original way, everyone knows there's a proxy
  - Used primarily when there is no other choice.
- Outbound transparent
  - CDN
    - Clients connect to explicit (advertised) proxy address
    - Server addresses are hidden from clients, servers could use non-routable addresses
    - Servers can still see client address on connection

Examples / Use Cases

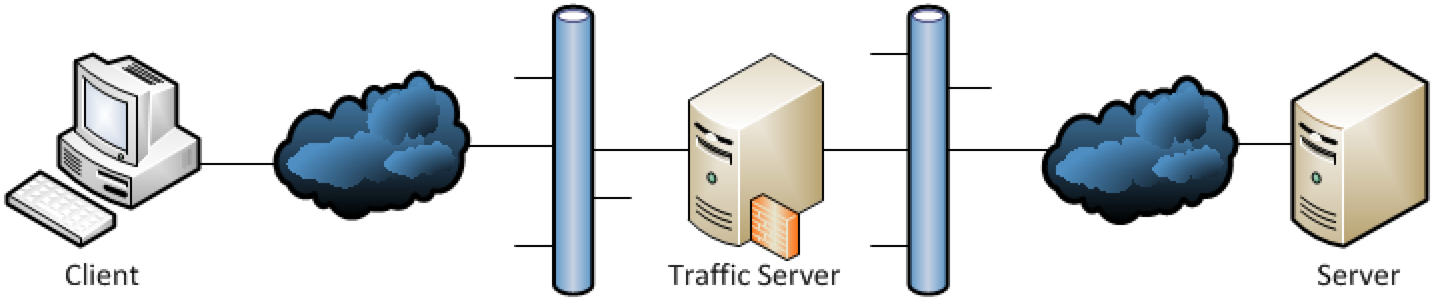
- Inbound transparent
  - Corporate: hide internal addresses behind proxy without client configuration
- Fully transparent
  - Proxy is not visible to clients or servers - no changes required for clients or servers, they still see each others’ addresses
  - Corporate use
    - Need to proxy
    - Need to have servers see distinct IP addresses for clients
    - Infeasible to configure clients for explicit proxy

Putting ATS in your network

DEPLOYMENT

Deploying

- Routed and Bridged require ATS inline
- WCCP requires intercepting router to be inline
- Packets <u>must</u> pass through intercepting box
- Simplified required topology looks like



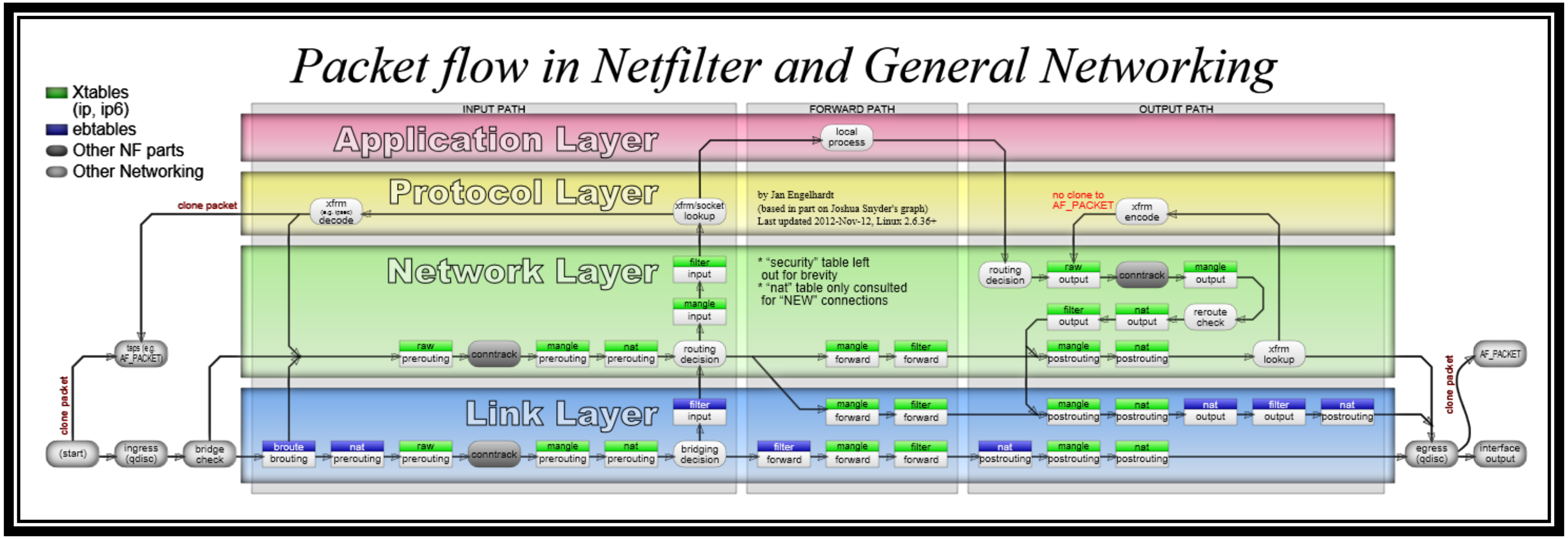
Adapt ATS to your network

- Because ATS can work in various modes you should pick the mode that works best in your network
- No mode is “better” than another, the modes are more or less appropriate for your network

Generic Deployment

- Normal packet flow is through ATS box
- Need to divert specific flows to ATS
  - Use iptables/ebtables to mark packets
  - Use routing table to re-route packets to ATS
  - Configure ATS to handle those packets
  - Tweak host OS
- See appendices for detailed commands

Simplified Linux Packet Handling



Packet flow in Netfilter and General Networking

by Jan Engelhardt
(based in part on Joshua Snyder's graph)
Last updated 2012-Nov-12, Linux 2.6.36+

* "security" table left out for brevity
* "nat" table only consulted for "NEW" connections

Now that I've scared you, let's look at just what we need to know for HTTP transparency

TPROXY

- Short for “Transparent PROXY”
- Linux kernel feature to support binding foreign IP addresses
- Accessible through iptables and socket options
- Should be present in modern Linux kernels

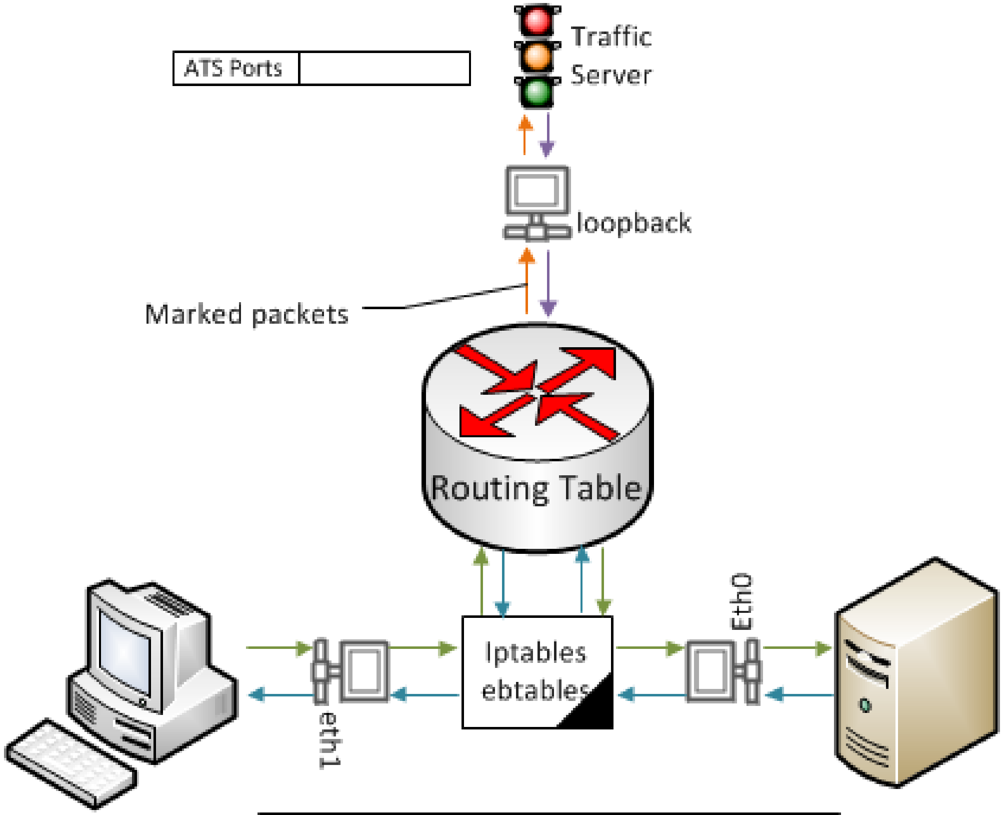
Building ATS

- Transparency will be enabled by default if possible
  - Can forced with `--enable-tproxy=force` option Uses built in values
  - Also with `--enable-tproxy=19` to force a value (e.g. 19) for the sockopt parameter
  - Need Linux Kernel 2.6.31.13 or later
- Requires POSIX capabilities, `libcap-devel`

Generic Pre-Deployment

- Decide on ATS options
  - Type of transparency
  - Routed, bridged, WCCP
- Enumerate server intercept ports
- Pick firewall mark
- Select inbound, outbound interfaces
- Select ATS proxy port(s)
- Verify clients can connect to server

Generic Setup



ATS Ports	

Client Interfaces	
Server Interfaces	
Server Ports	
Firewall Mark	/

Pre-deployment cautions

- ATS box is in line so all other traffic will pass through it
- Firewall mark and ATS proxy ports are arbitrary and local so select to avoid interference with other activity on the host

ebtables

- Break packets out of layer 2 bridge
- Packets then processed as in other cases
- Can do both IPv4 and IPv6 with ebtables

iptables

- Set firewall mark to enable special routing
  - Can use entire mark or a bit range and value
  - Only need 1 bit
  - Mark based on server port and host interface
- Mark `TPROXY` for inbound transparent
  - Required for ATS to accept connection with foreign destination address
- Redirect to ATS proxy port
- Use `ip6tables` for IPv6

iptables

- iptables is used for many things, including firewalling
- Lots of potential cross interference
- ATS uses the `mangle` table only
- Default iptables configuration will block ATS operation – test client to server connectivity through ATS host without ATS

Routing table

- Use policy routing to force table for packets with ATS firewall mark
  - Add table for intercepted packets
  - Table sends everything to loopback
- Side tables mean no direct interaction with normal routing table

ATS Configuration

- Create proxy port(s) marked transparent as needed
- ATS proxy port must agree with iptables redirection for inbound transparent

ATS Transparency Options

- Transparency mode
  - `tr-in` = inbound transparent
  - `tr-out` = outbound transparent
  - `tr-full` = fully transparent
    - Can also use `tr-in:tr-out`
- `tr-pass` = transparent pass through
- Options other than `ipv6,ipv4` may collide with transparency

Host OS Configuration

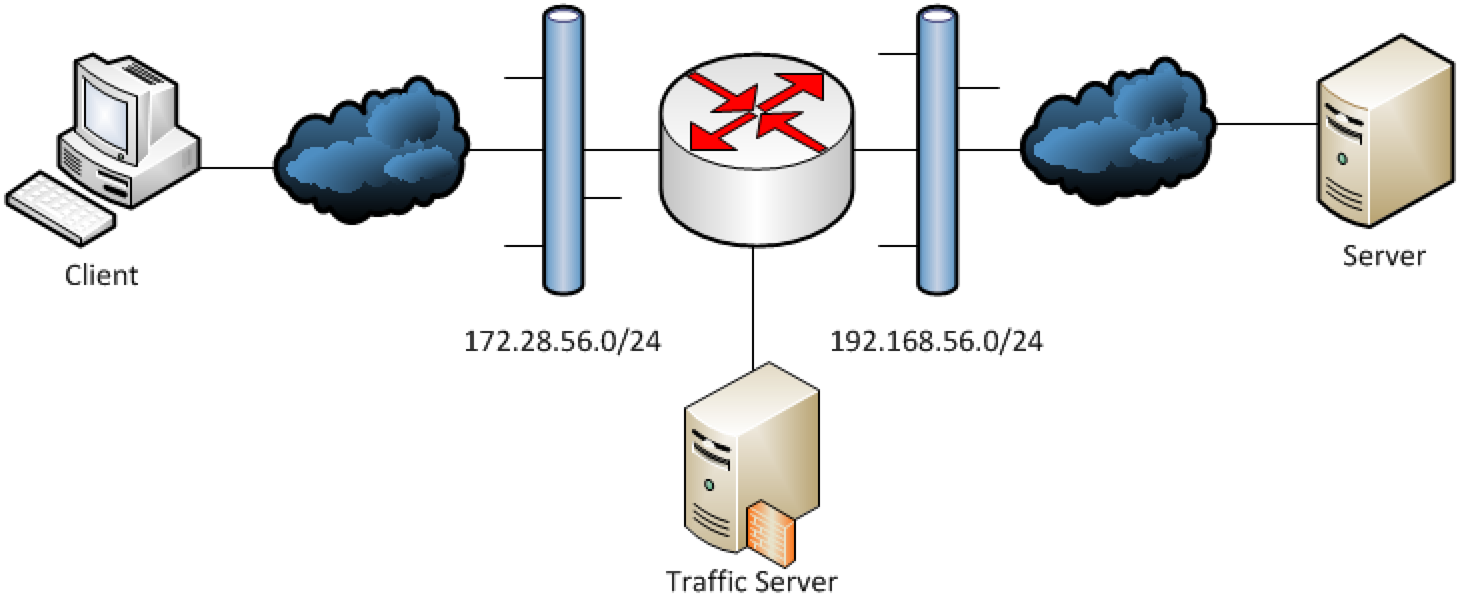
- Enable packet forwarding
- Disable reverse path check (rp_filter) on transparent physical interfaces
- Do routing or bridged configuration
  - But that's already done because <u>of course</u> you've checked for connectivity before deploying ATS

Specific Deployment Cases

- Appendix has scripts for each case
- Main script for standard routed/bridged cases
  - Can use full case for either half case
- Example on using `NAT` instead of `TPROXY`
- Discussion on using WCCP

WCCP Topology

- Router does packet interception for ATS



WCCP

- Past end of life Cisco protocol, still in use
  - Significant parts undocumented, no support
- Effectively remote control policy routing
  - Heartbeat to allow bypass on cache failure
- Best with 3 (or more) interfaces
- Can be done with 2 interfaces using tunnels
- Pointless if not inbound transparent

WCCP ATS Configuration

- Configuration values
  - proxy.config.wccp.addr STRING <IPv4 address>
  - proxy.config.wccp.services STRING <path>
- Services file describes WCCP services for ATS
  - Need two groups – inbound and outbound
  - Must match router config
- Other configuration as for previous cases

WCCP Host Configuration

- For L2 use routed transparent case
- For tunnel use 2 firewall bits
  - One for packets from tunnel (TPROXY marked)
  - One for packets from ATS to put in tunnel
- Two interface router requires use of tunnel
- Tunnel requires disabling PMTU discovery

ATS Plugins

- Can control outbound transparency per connection
- Can control server address per connection
- Still a few bugs on URLs because HTTP headers are different

Making it work

TROUBLE SHOOTING

Trouble Shooting

- Step One: Make it work without ATS
  - Cannot over emphasize this
  - Always see Step One

Trouble Shooting Tools

- Tcpdump
  - Almost always installed
  - Requires only text interface
  - Directly or to make capture files for Wireshark
- Wireshark
  - Graphical interface, very powerful
- iptables, ebtables hit counts
- `netstat --tcp --listen -n`

Trouble Shooting – ATS logging

- Debug messages have a tag
- Turn on with
  `proxy.config.diags.debug.enabled INT 1`
- Set output tags with
  - `traffic_server` command line option
    `-T “tag1|tag2”`
  - `records.config` value
    `proxy.config.diags.debug.tags STRING “tag1|tag2”`
- Value for tags is regular expression
  - So “`host`” matches tags like “`host`”, “`host_db`”

Troubleshooting Checklist

- □ Remove ATS ebtables, iptables, routing – do you have connectivity?
- □ Enable ATS – are all the processes running?
- □ Check ATS logs to verify startup success. Look for error messages!
  - □ `traffic.out`
  - □ `error.log`
  - □ `dmesg`

Troubleshooting Checklist

- ❑Review configuration
  - ❑Verify iptables target port, ATS proxy port match
  - ❑Check iptables for packet / connection filtering
  - ❑Bridge mode – ebtables set up?
  - ❑Policy routing in place?
  - ❑Check OS tweaks (ip_forward, rp_filter)

Troubleshooting Checklist

- ❑ Capture client side
  - ❑ SYN-ACK from ATS?
  - ❑ Connection / request sent?
- ❑ Capture loopback
  - ❑ Client SYN packets redirected?
  - ❑ SYN-ACK from ATS?
- ❑ Capture server side – packets outbound?
- ❑ Check ATS logs for connections

Trouble Shooting Notes

- Be careful using IP addresses to determine packet sources – the whole point of transparency is to fiddle with those
- Each packet has a MAC address which is useful for determining original source
- Can also use the “IP id” value to trace packet sources (shown in some tools)

ATS Logging

- Enable debugging out
  - -T “tags”
  - Edit `records.config` values
    - proxy.config.diags.debug.enabled INT 1
    - Proxy.config.diags.debug.tags STRING “tag1|tag2”
- Useful tags
  - “`hostdb`”, “`dns`” – see outbound connections
  - “`http_accept`” – see inbound connections
  - “`tproxy`” - extra TPROXY related events
- Output to `etc/trafficserver/traffic.out`

WCCP Trouble shooting

- Router: `show ip wccp`
- ATS tag “wccp”
- Look for heartbeat packets via packet capture
- Check for redirected packets
- Check that both service groups are working

Issues

- Potential problems from field experience
  - Origin server address resolution
  - Port transparency
  - Proxy port address binding
  - Keep Alive
  - HTTPS
  - Non-HTTP tunneling
  - IP family lock
  - Currently limited to Linux variants

Origin Server Address

- Origin server resolved twice – client, ATS
- If server has RR DNS these may differ
- Can cause problems (MS Windows Update)
- Inefficient (two lookups per access)
- May complicate local DNS server setup
- Can override to use client supplied address
  - `proxy.config.http.use_client_target_addr INT 1`
  - But lose some control (trust client to resolve correctly)

Proxy Port Address Binding

- Transparent ports can’t bind to local address
- Inbound must de facto bind to ANY_ADDR
- Outbound must bind to client source address
- This leads to binding to loopback interface, not any physical interface
- Proxy port options `ip-in,ip-out` can conflict

Port Transparency

- By default server connection can have a different client port than actual client
- Can configure ATS to use the client port
  - `proxy.config.http.use_client_source_port INT 1`
- Requires outbound transparency
- Can lead to port jamming via Keep-Alive (TS-1424)
- Linux kernel shares port space for port binding to foreign addresses -> ~64K connection limit

Keep Alive

- ATS doesn’t always match keep alive between client side and server side.
- Can cause “port shift”
- In practice seems to matter only rarely

HTTPS

- HTTPS proxying requires certificates
  - ATS must terminate the connections
  - Easy for CDN situations
  - Can’t just slap on `ssl` proxy port option
- HTTPS can be blind tunneled
  - Can still check IP addresses but little else

Non-HTTP Tunneling

- There exist protocols that use port 80 and HTTP like headers but are not HTTP
- By default ATS rejects the connection
- TS-1423 patch enables this – use with caution

IP Family lock

- ATS handles cross IP family connections
  - E.g. IPv4 client connection, IPv6 server connection
- Not possible with transparency
  - Preserving the address implies preserving family
- TS-1307 – DNS lookup for outbound transparent forces family
- Proxy port option `ip-resolve` is ignored, forced to `client` if outbound transparent

Remapping

- In general remapping “works”
- Be careful – client and ATS will differ on the IP address for server
- Currently explicitly inhibited if ATS uses server address from client connection
  - Not sure now why I did that…
- Can do more sophisticated things in plugin

Linux Required

- Depends on TPROXY, iptables, policy routing
- Requires POSIX capabilities or equivalent
  - Transparent binding is a privileged operation
- Want it to work on other operating systems? Volunteers always appreciated!

Script Kiddies

- For inbound transparent ISP case, script kiddies probing for open servers
- ATS will accept connections to any foreign IP address:port
- Script thinks everything has an open port at intercept ports
- Can have an impact on ATS loading

Scripts and Resources

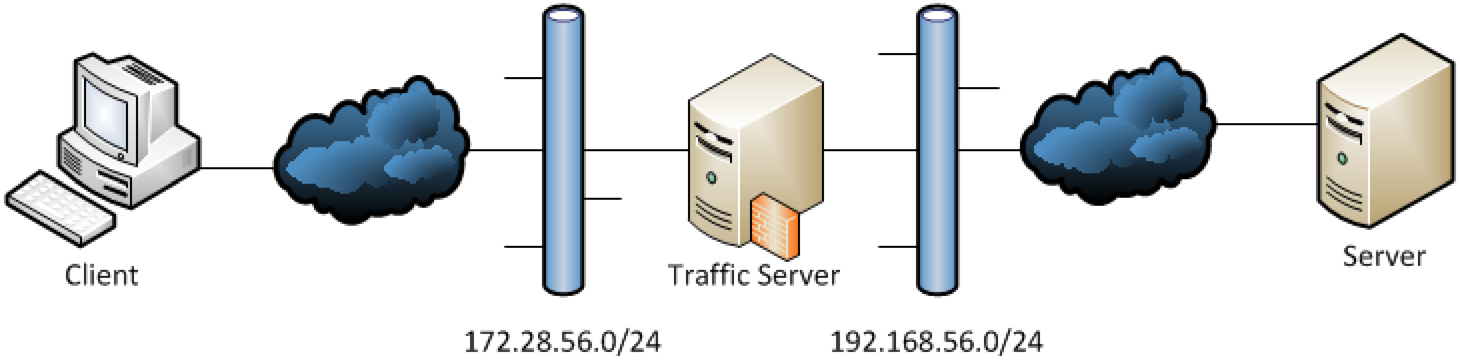
APPENDIX

A beginning…

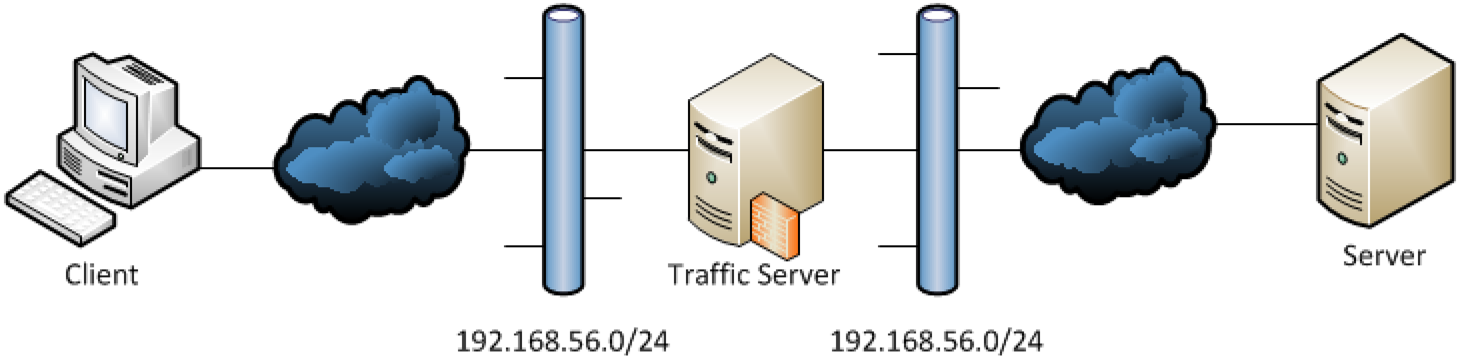
- These scripts are just starting points
  - Customize for local conditions
  - Illustrate essential commands and basic options
  - Plenty of other documentation for commands

Environmental Concerns

- Scripts presume
  - Client interface is eth1
  - Server interface is eth0
  - Server side network is 192.168.56.0/24
  - Client side network is 172.28.56.0/24
  - ATS proxy port is 8080



Routed Topology



Bridged Topology

ATS Configuration Examples

- As of ATS 3.2 use the configuration value `proxy.config.http.server_ports STRING` for all proxy ports
- Each proxy port has a descriptor string of colon separated values
  - Two proxy ports, at 8080 for IPv4 and IPv6 `8080:ipv4:tr-full,ipv6:8080:tr-full`
  - Outbound transparent at 9090, IPv4, passthrough `tr-out:tr-pass:9090`

Setup Script

- Script for bridged and routed cases
- Set shell variables to control setup
- Works from a cold start
  - Pick out pieces for less intrusive operation
- File name ‘`acna-universal.sh`’

```
#!/bin/sh

### Universal version for all 6 cases (bash)
# Set these to control the script operations
TOPOLOGY='BRIDGED'
# TOPOLOGY='ROUTED'
# Transparency. Set both to 1 for full.
INBOUND=1 # set to 0 for not inbound transaprent
OUTBOUND=1 # set to 0 for not outbound transaprent

## System tweaks
# Enable IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
# Disable RP filter. Oddly, not needed on loopback
echo 0 > /proc/sys/net/ipv4/conf/eth0/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/eth1/rp_filter

if [ $TOPOLOGY = 'BRIDGED' ] ; then
    ## Set up the bridge interfaces
    # Update cluster interface if set to a subsumed interface
    brctl addbr br0
    ifconfig br0 up
    brctl stp br0 off
    brctl addif br0 eth0
    brctl addif br0 eth1

    # Turn off addresses on physical interfaces to avoid confusion
    ifconfig eth0 0 0.0.0.0
    ifconfig eth1 0 0.0.0.0

    ## Put an address on the bridge virtual interface
    #ifconfig br0 192.168.56.11 netmask 255.255.255.0 up
    ## Or use DHCP:
    # Shutdown current DHCP client operation, terminate any leases.
    # This avoids problems with subsumed interfaces holding addresses
    dhclient -r
    # Start DHCP client daemon for bridge interface
    dhclient br0

    ## Do the same for IPv6 if needed
    #ip -6 addr add fc01:192:168:56::11/64 dev br0
fi

## Set up policy routing for redirected packets
#  Clear any existing rules.
ip rule delete fwmark 1/1 > /dev/null 2>&1
ip -6 rule delete fwmark 1/1 > /dev/null 2>&1
#  Add new rules
ip rule add fwmark 1/1 table 1
ip -6 rule add fwmark 1/1 table 1
#  Set routes to use rules
ip route add local 0/0 dev lo table 1
ip -6 route add local ::/0 dev lo table 1

if [ $TOPOLOGY = 'BRIDGED' ] ; then
    # Routing tables need to have a default route via br0 and not via one of the
    # physical interfaces. The latter seems to break anything that goess off the local
    # network. Sometimes you have to delete those routes explicitly.
    # ip route delete default via 192.168.56.1 dev eth0

    # br0 must have an address on the same network as the default gateway addr
    ip route add default via 192.168.56.1
    #ip -6 route add default via fc01:192:168:56::11
fi
```

```
## Iptables setup
#  IPv4
#  Brutal - get rid of everything else in the mangle table and put our stuff in
iptables -t mangle --flush PREROUTING
if (( $INBOUND )) ; then
    iptables -t mangle -A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j TPROXY --on-ip 0.0.0.0 --
on-port 8080 --tproxy-mark 1/1
fi
if (( $OUTBOUND )) ; then
    iptables -t mangle -A PREROUTING -i eth0 -p tcp -m tcp --sport 80 -j MARK --set-mark 1/1
fi

# Be sure we're not filtering packets before they go to ATS (default on Linux)
# This disables *all* firewall protection. Don't do this if you want to preserver
# any filtering! In that case verify the filter rules don't break connectivity.
# I use this because the default installed rules are a problem.
iptables -t filter --flush FORWARD
ip6tables -t filter --flush FORWARD
iptables -t filter --flush INPUT
ip6tables -t filter --flush INPUT
# You might need to flush the mangle table as well, if there's cruft there.

if (( $INBOUND )) ; then
    ip6tables -t mangle -A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j TPROXY --on-ip :: --on-
port 8080 --tproxy-mark 1/1
fi
if (( $OUTBOUND )) ; then
    ip6tables -t mangle -A PREROUTING -i eth0 -p tcp -m tcp --sport 80 -j MARK --set-mark 1/1
fi

if [ $TOPOLOGY == 'BRIDGED' ] ; then
    ## EBTables setup - bounce all port 80 TCP traffic to iptables (layer 3 routing)
    #  Flush the table - again, you'll need to do more testing if this isn't viable
    ebtables -t broute -F
    if (( $INBOUND )) ; then
            #  enable routing for traffic to web server
            ebtables -t broute -A BROUTING -p IPv4 --ip-proto tcp --ip-dport 80 -j redirect --
redirect-target DROP
            ebtables -t broute -A BROUTING -p IPv6 --ip6-proto tcp --ip6-dport 80 -j redirect -
-redirect-target DROP
    fi

    if (( $OUTBOUND )) ; then
            #  do the same from traffic from web server
            ebtables -t broute -A BROUTING -p IPv4 --ip-proto tcp --ip-sport 80 -j redirect --
redirect-target DROP
            ebtables -t broute -A BROUTING -p IPv6 --ip6-proto tcp --ip6-sport 80 -j redirect -
-redirect-target DROP
    fi
fi
```

Transparency with NAT

- Can use the iptables NAT capability for inbound transparent
- ATS proxy port is <u>not</u> marked inbound transparent!
- For outbound transparent could use TPROXY but then why use NAT inbound?
- Must resolve server address in ATS, the client resolved server address is destroyed by NAT
- IMHO only useful for inbound transparent case to avoid TPROXY entirely
- See appendix script for implementation details

NAT style

- Proxy ports “8080”
- iptables

  ```
  iptables –t nat –A PREROUTING –i eth1 –p tcp –m
  tcp --dport 80 –j REDIRECT --to-port 8080
  ```

- Enable IP forwarding

  ```
  echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

WCCP Setup

- Example router config
  - Tunnel addresses are on the 10.28.56.0/24 network
  - ATS host shares outside interface network (192.168.56.0/24)

```
no ip source-route
ip wccp check services all
ip wccp 51 password apache
ip wccp 52 password apache

interface Tunnel0
ip address 10.28.56.1 255.255.255.0
tunnel source 192.168.56.12
tunnel destination 192.168.56.11

interface FastEthernet0/0
ip address 192.168.56.12 255.255.255.0
ip wccp redirect exclude in
ip wccp 52 redirect in

interface FastEthernet0/1
ip address 172.28.56.12 255.255.255.0
ip wccp 51 redirect in
```

• ATS Host config for WCCP (example)

```
# Man page says 'delete' but that doesn't work. Must use 'del'
ip tunnel del wccp-tunnel > /dev/null 2>&1
ip tunnel add wccp-tunnel mode gre remote 192.168.56.12 local 192.168.56.11 nopmtudisc
ip link set wccp-tunnel up # must be up or route add will complain
ip addr add 10.28.56.2/24 dev wccp-tunnel
ip route add 10.28.56.0/24 dev wccp-tunnel

ip route add 172.28.56.0/24 dev eth0 via 192.168.56.12

# Clear out old cruft. Really should parse the output of
# ip rule list. Someday...
ip rule delete fwmark 1/3 > /dev/null 2>&1
ip rule delete fwmark 1/1 > /dev/null 2>&1
ip rule delete fwmark 2/3 > /dev/null 2>&1
ip rule delete fwmark 2/2 > /dev/null 2>&1

ip rule add fwmark 1/3 table 1
ip rule add fwmark 2/3 table 2

if [ ! -z "$(ip route show table 1)" ] ; then
  ip route delete table 1;
fi
if [ ! -z "$(ip route show table 2)" ] ; then
  ip route delete table 2;
fi
ip route add local 0.0.0.0/0 dev lo table 1
ip route add default dev wccp-tunnel via 10.28.56.2 table 2

# Clear current iptables
iptables -t mangle --flush
# Bypass local network traffic
iptables -t mangle -A PREROUTING -s 192.168.56.0/24 -i eth0 -j ACCEPT
iptables -t mangle -A PREROUTING -d 192.168.56.0/24 -i eth0 -j ACCEPT

iptables -t mangle -A PREROUTING -i wccp-tunnel -p tcp -m tcp -j TPROXY --tproxy-mark 1/3 --on-port 8080

iptables -t mangle -A OUTPUT -p tcp -m tcp --sport 80 -j MARK --set-mark 2/3

echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc
```

- WCCP services file

```
security = {
  key = "apache";
  option = "MD5";
};

services = (
  {
    name = "ATS Client";
    description = "Capture packets from client.";
    id = 51;
    type = "DYNAMIC";
    priority = 240;
    protocol = 6;
    primary-hash = ( "src_ip" );
    ports = ( 80 );
    assignment = ( "hash" );
    forward = ( "gre" );
    return = ( "gre" );
    routers = ( "172.28.56.12" );
  },
  {
    name = "ATS Server";
    description = "Capture packets from origin server.";
    id = 52;
    type = "DYNAMIC";
    priority = 240;
    protocol = 6;
    primary-hash = ( "dst_ip" );
    ports = ( 80 );
    port-type = "src";
    assignment = ( "mask" );
    forward = ( "gre" );
    return = ( "gre" );
    routers = ( "172.28.56.12" );
  }

);
```

Resources

- ATS has online documentation, a wiki, mailing lists, bug tracker, and IRC channel. Access these via
  - http://trafficserver.apache.org
- NG Consulting services