
# Effective DNS SRV for SIP

Torrey Searle, SIPNOC, April 25, 2011

# The Problem

- Our customers need a failover solution for high availability

# The Problem

- Our customers need a failover solution for high availability
- Our customers do not want to make custom modifications to their equipment

# The Problem

- Our customers need a failover solution for high availability
- Our customers do not want to make custom modifications to their equipment
- Our customers want to have no (or few calls) lost due to a failover

# The Problem

- Our customers need a failover solution for high availability
- Our customers do not want to make custom modifications to their equipment
- Our customers want to have no (or few calls) lost due to a failover
- We need a solution that can easily scale

# Original Solution: Options Ping

- Behavior of Options requests is not well defined for an SBC (should behave like a proxy to properly work, but often behaves like a UAS)
- Doesn't scale very well

**New Solution: DNS SRV**

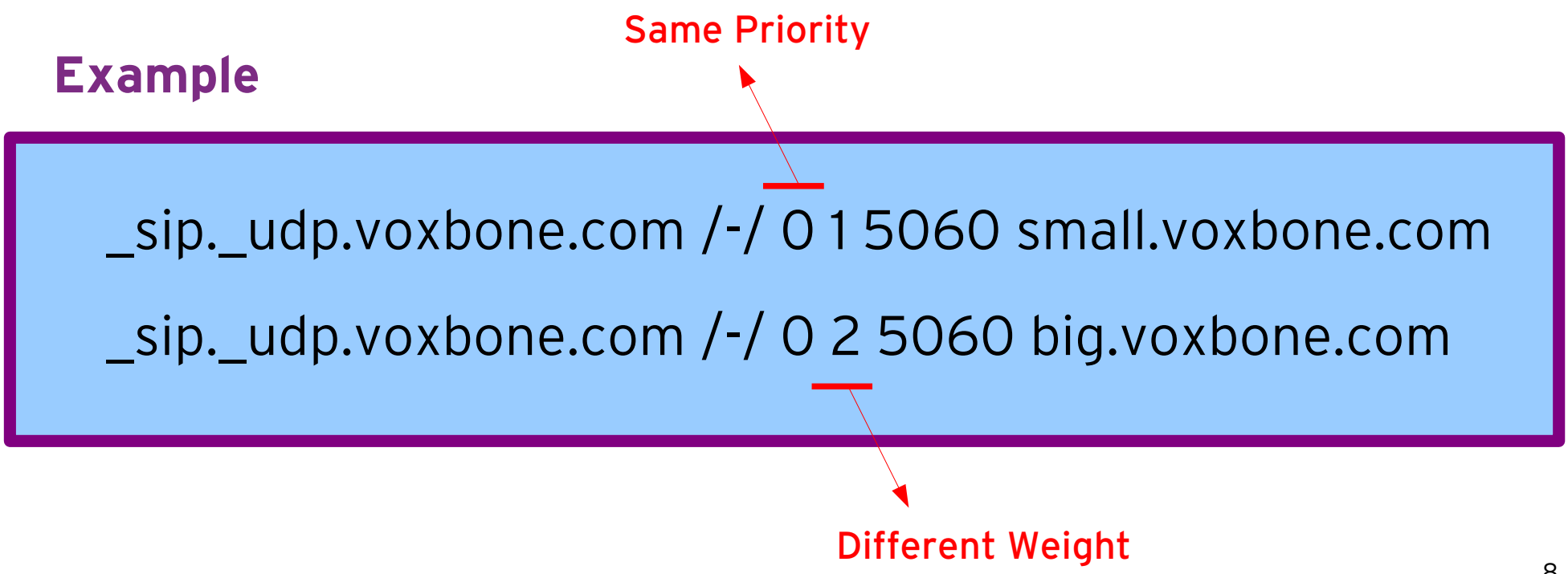
# Everybody knows, but just to be sure : What is DNS SRV?

- Allows definition of per-service DNS records
- Defined in RFC 2782
- Built-in mechanisms for load balancing and fail over

# Load balancing in DNS SRV

- All SRV records with the same Priority form a load balancing group
- Weight allows for traffic shaping : bigger weights get more traffic

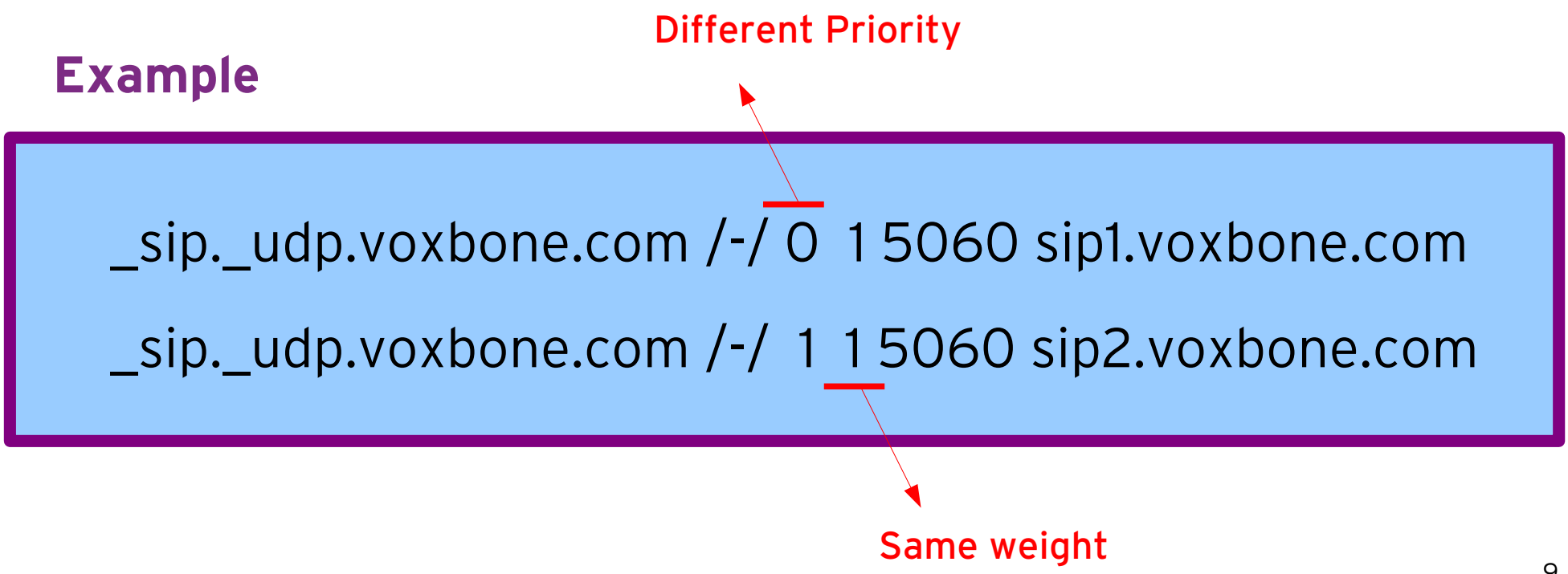
**Example**



# Failover in DNS SRV

- Records with a lower priority value are tried first
- Records with higher priority values are only tried if all records with a lower priority are considered unreachable

**Example**



# Use of DNS SRV failover for SIP

- Use of DNS SRV for SIP defined in RFC3263 (Locating SIP Servers)
- 3 cases are described for failover in UDP :
  - Transaction timeout (timer B or F)
  - ICMP error (Destination Port Unreachable)
  - 503 Server Error (Service Unavailable)

Case 1

# Transaction time-out

- Timer B (invite transactions) and Timer F (non-invite transactions) are set at 64xT1 and T1 = 0,5 sec.  Hence both timers are set at 32 seconds
- 32 seconds is too long for invites. Callers will normally hang up after 10-15 seconds

Case 2

# ICMP error

- ICMP responses require connect() to be called on the UDP socket

Case 2

# ICMP error

- ICMP responses require connect() to be called on the UDP socket
- Connect() causes all packets received from other hosts to be rejected – not practical for proxies to do

Case 2

# ICMP error

- ICMP responses require connect() to be called on the UDP socket
- Connect() causes all packets received from other hosts to be rejected – not practical for proxies to do
- Reception of the ICMP response is not guaranteed

Case 2

# ICMP error

- ICMP responses require connect() to be called on the UDP socket
- Connect() causes all packets received from other hosts to be rejected – not practical for proxies to do
- Reception of the ICMP response is not guaranteed
- In conclusion : very effective for TCP, but not for UDP (hence for most VOIP traffic)

Case 3

# 503 response

- 503 responses are fast
- Only downside : doesn't work when equipment has crashed completely

# Voxbone applies its own solution for delivery of calls to customers

- Given its effectiveness and speed,  Error 503 messages are taken into account
- Standard transaction timers and ICMP errors are discarded and replaced by a mechanism we refer to as **Timer N**

# The reception of a "100 Trying" indicates if a server is available

- A UAS (or Stateful Proxy) must generate either a 100 Trying or a final response withing 200ms (RFC3261 17.2.1)
- 100 Trying is hop by hop, not end to end

# A new timer N is implemented based on the reception of this message

- We implemented a new Timer (Timer N) on the application layer

# A new timer N is implemented based on the reception of this message

- We implemented a new Timer (Timer N) on the application layer
- The timer starts when an Invite request with DNS SRV routing is sent

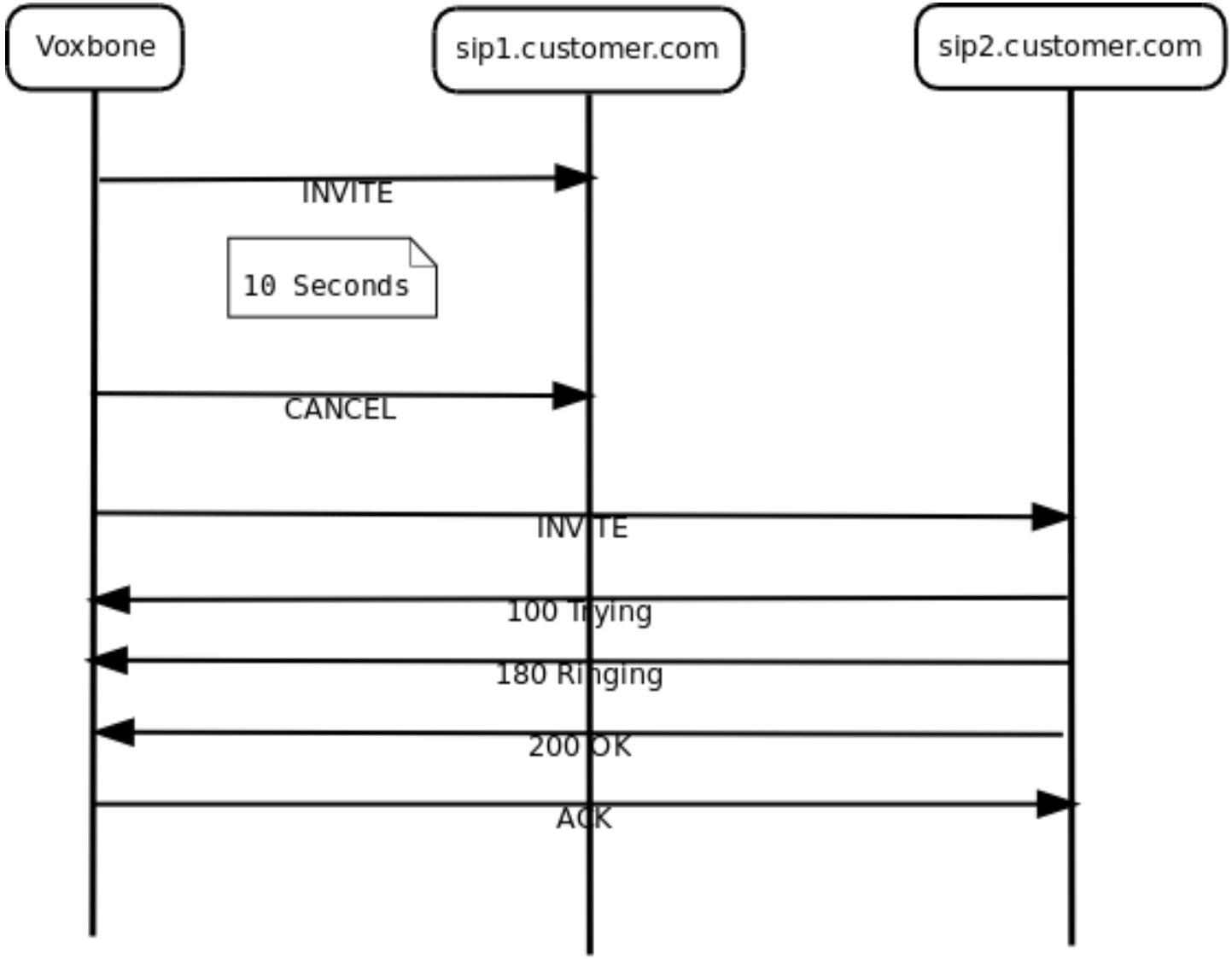
# A new timer N is implemented based on the reception of this message

- We implemented a new Timer (Timer N) on the application layer
- The timer starts when an Invite request with DNS SRV routing is sent
- Stopped when the initial 100 Trying response is received

# A new timer N is implemented based on the reception of this message

- We implemented a new Timer (Timer N) on the application layer
- The timer starts when an Invite request with DNS SRV routing is sent
- Stopped when the initial 100 Trying response is received
- If no 100 Trying response is received **within 10 seconds** the transaction is canceled and a new Invite is sent to the next DNS SRV record

# Call flow illustration



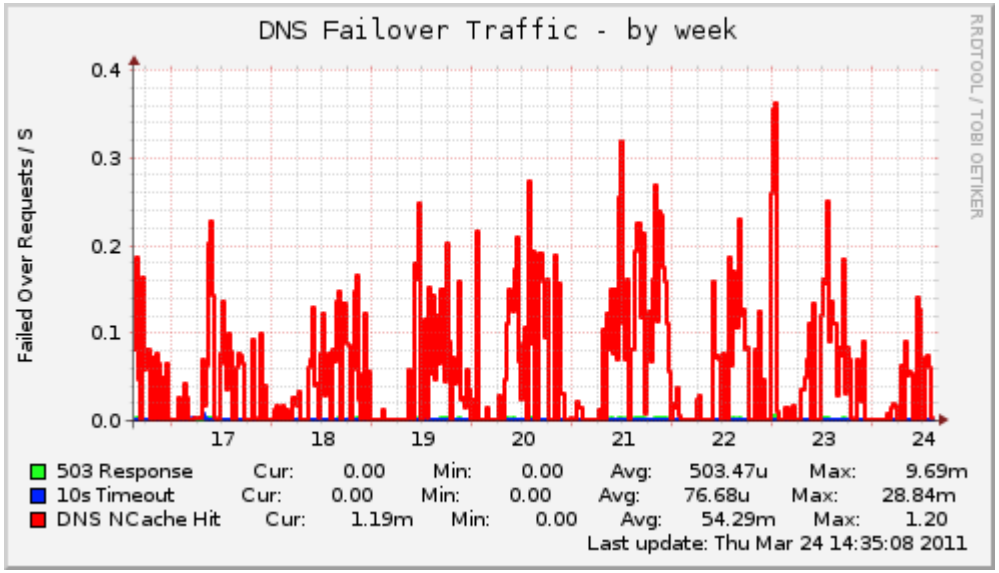
# Negative caching is used to keep the call establishment time as low as possible

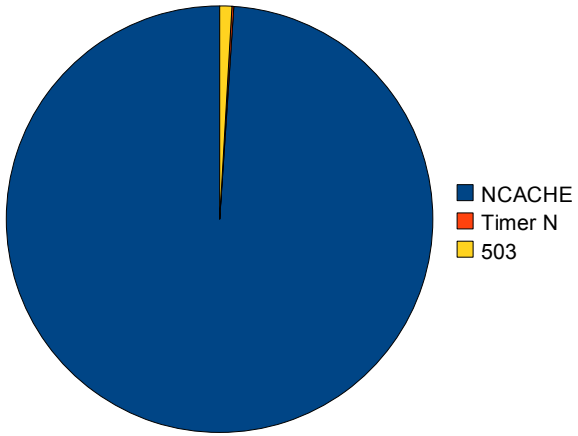
- 10 seconds is still a long time to wait for call establishment as long as a server is unavailable
- Negative caching resolves this as follows :
  - When a Failover occurs, the IP address of the failed node is saved in the "negative cache" for 5 minutes
  - IP addresses in this cache are removed from the DNS look up results

    *Exception: if all records for a domain are negative cached, the application will continue to try to set up the call with the last entry*

# The DNS Failover mechanism has two positive impacts

1. Call establishment rate increased by .7%



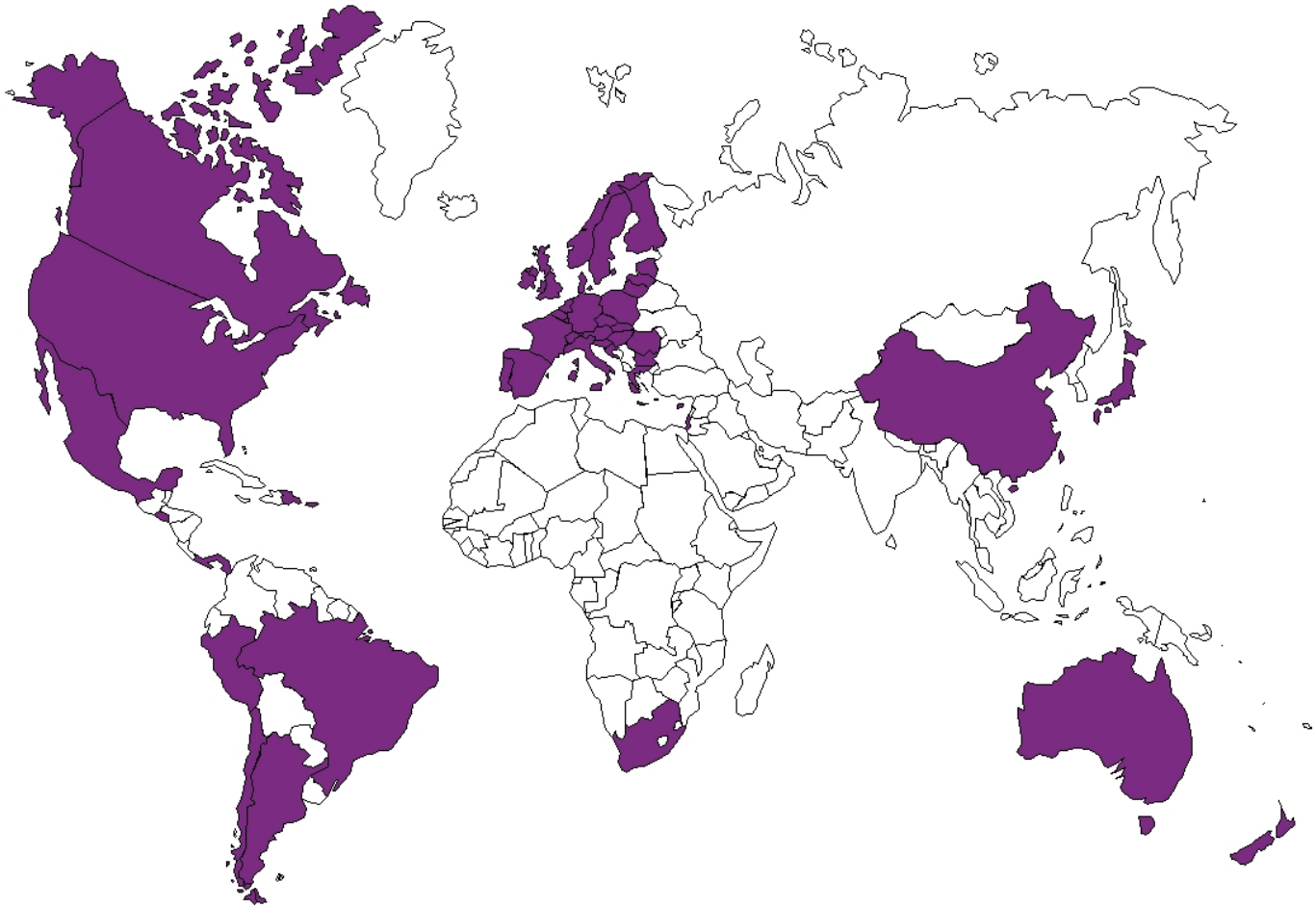




2. Only a very small % of traffic has slow call establishment time

# Conclusions

- RFC3263 is good, but failover mechanisms defined are not sufficient
- Negative Caching dramatically improves call establishment times and rates during an outage
- The current value of the timer, 10 seconds, may still be too long for effective failover.  The optimum value of the timer is to be investigated

# Thank you.  Questions?



**Inbound geographical and toll free phone numbers from 50 countries and 4000 cities**