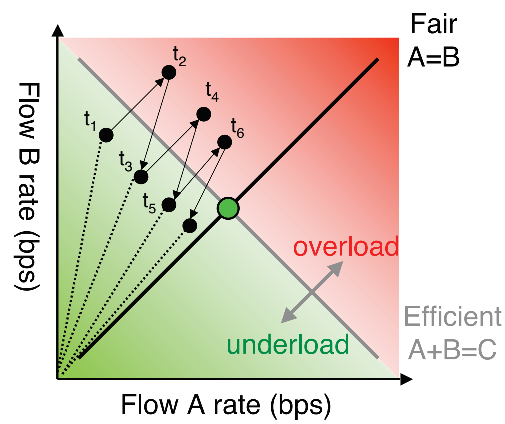
Lecture 10: TCP Friendliness, DCCP, NATs, and STUN

TCP Friendliness

Congestion Control

- TCP dynamically adapts its rate in response to congestion
- AIMD causes flows to converge to fair goodput
- But how do losses (e.g., bit errors) affect goodput?
- What about UDP?

Chiu Jain Phase Plots



Responding to Loss

- Set threshold to $\frac{cwnd}{2}$
- On timeout
  - Set cwnd to 1
  - Causes TCP to enter slow start
- On triple duplicate ACK (Reno)
  - Set cwnd to $\frac{cwnd}{2}$
  - Retransmit missing segment
  - Causes TCP to stay in congestion avoidance

Analyzing TCP Simply

- Assume all segments are MSS long
- Assume a packet loss rate p
- Assume a constant RTT
- Assume p is small (no timeouts)

Analysis

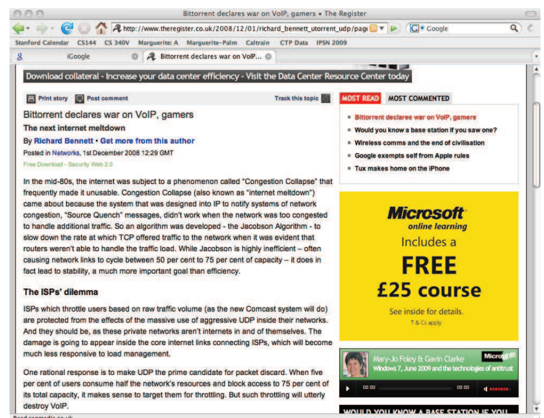
- **Window size W cuts to $\frac{W}{2}$ after a loss**
- **Grows to W after $\frac{W}{2}$ RTTs**
- **Goodput = $\frac{3}{4} \cdot W \cdot MTU \cdot \frac{1}{RTT}$**

Window Size

- $p = \frac{1}{(\frac{W}{2}+(\frac{W}{2}+1)+\ldots+W)}$
- $p \approx \frac{1}{\frac{3}{8}W^2}$
- $W \approx \sqrt{\frac{8}{3 \cdot p}}$
- **Goodput = $\frac{3}{4} \cdot \sqrt{\frac{8}{3 \cdot p}} \cdot MTU \cdot \frac{1}{RTT}$**
- **Goodput = $\frac{1.22 \cdot MTU}{RTT \cdot \sqrt{p}}$**
- **Constant factor changes based on delayed acks, etc.**

TCP Friendliness

- **Don't want other protocols to disrupt TCP**
- **UDP happily shuts down TCP flows**
- **"TCP friendliness:" obeying TCP congestion control as per prior goodput equation**
  - Does not imply acting like TCP
  - E.g., does not require abrupt window changes



Bittorrent declares war on VoIP, gamers • The Register

http://www.theregister.co.uk/2008/12/01/richard_bennett_utorrent_udp/page

Download collateral - Increase your data center efficiency - Visit the Data Center Resource Center today

Print story   Post comment   Track this topic

Bittorrent declares war on VoIP, gamers

The next internet meltdown

By Richard Bennett • Get more from this author

Posted in Networks, 1st December 2008 12:29 GMT

Free Download - Security Web 2.0

In the mid-80s, the internet was subject to a phenomenon called "Congestion Collapse" that frequently made it unusable. Congestion Collapse (also known as "internet meltdown") came about because the system that was designed into IP to notify systems of network congestion, "Source Quench" messages, didn't work when the network was too congested to handle additional traffic. So an algorithm was developed - the Jacobson Algorithm - to slow down the rate at which TCP offered traffic to the network when it was evident that routers weren't able to handle the traffic load. While Jacobson is highly inefficient – often causing network links to cycle between 50 per cent to 75 per cent of capacity – it does in fact lead to stability, a much more important goal than efficiency.

The ISPs' dilemma

ISPs which throttle users based on raw traffic volume (as the new Comcast system will do) are protected from the effects of the massive use of aggressive UDP inside their networks. And they should be, as these private networks aren't internets in and of themselves. The damage is going to appear inside the core internet links connecting ISPs, which will become much less responsive to load management.

One rational response is to make UDP the prime candidate for packet discard. When five per cent of users consume half the network's resources and block access to 75 per cent of its total capacity, it makes sense to target them for throttling. But such throttling will utterly destroy VoIP.

MOST READ   MOST COMMENTED

- Bittorrent declares war on VoIP, gamers
- Would you know a base station if you saw one?
- Wireless comms and the end of civilisation
- Google exempts self from Apple rules
- Tux makes home on the iPhone

Mary-Jo Foley & Gavin Clarke
Windows 7, June 2009 and the technologies of antitrust

Read regmedia.co.uk

ledbat WG

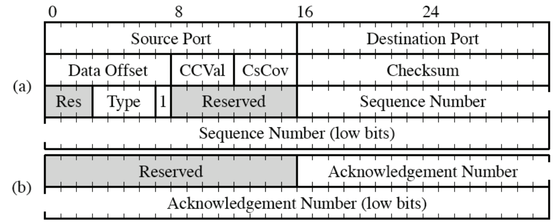
- **"The LEDBAT WG is chartered to standardize a congestion control mechanism that should saturate the bottleneck, maintain low delay, and yield to standard TCP."**
- **TCP-friendliness is insufficient for modern P2P applications**
  - Flow fairness, not application fairness
  - TCP fills queues
- **Elastic workloads vs. inelastic workloads**

DCCP

DCCP

- **Datagram Congestion Control Protocol (DCCP) provides congestion control for unreliable datagrams (RFC 4340)**
- **Connection-oriented protocol**
  - Request-response-ack establishment
  - Close-reset or CloseReq-Close-reset teardown
- **Counts packets, not bytes**

DCCP Segment



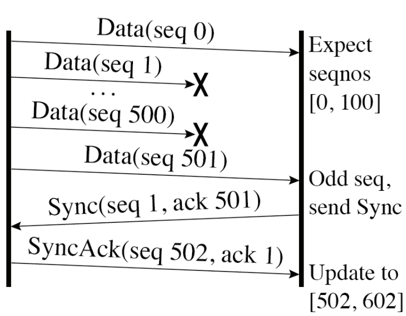
Sequence Numbers

- **Every DCCP packet uses a new sequence number**
  - Data
  - Acknowledgements
  - Control traffic
- **Acknowledgements are for *last packet received***
  - Not cumulative acknowledgements
  - Does not succinctly describe connection history
  - Options can give packet vectors

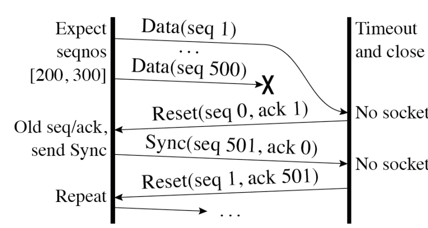
Synchronization

- **DCCP uses sequence number windows to protect from attacks**
- **Large bursts of losses cause packets to fall past windows**
- **Need to resynchronize**

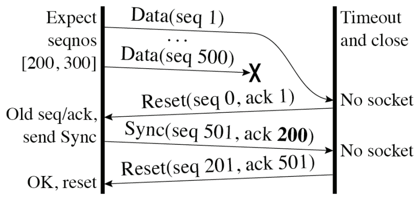
Synchronization Exchange



Synchronization on Reset Problem



Synchronization on Reset Solution



Congestion Control

- **Defines Congestion Control IDs (CCIDs)**
- **Negotiated with change/confirm L/R options**
- **Each half-connection can have different congestion control**
- **CCID 2: TCP congestion control (AIMD) (RFC 4941)**
- **CCID 3: TCP-friendly congestion control (RFC 4942)**

CCID 2

- **Uses TCP congestion control**
  - Maintains a `cwnd`, slow-start, etc.
- **Adds congestion control to acks**
  - Sender specifies an *AckRatio*, R
  - Ratio of data to ack packets (TCP with delayed ACKs is 2)
  - On detecting ack losses, double R
  - After $\frac{cwnd}{R^2-R}$ lossless congestion windows, decrement R

CCID 3

- **Uses TCP-friendly congestion control**
- **Uses a sending rate, rather than a congestion window**
- **Receiver sends feedback once per RTT, reporting loss rate**
- **If sender hears no feedback, halves sending rate**
- **Security issue with loss rate reporting: report *loss intervals*, rather than just a loss rate, verifiable with ECN nonces**

DCCP Today

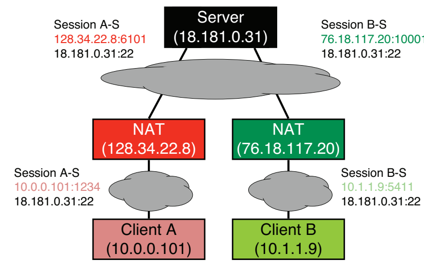
- **Numerous implementations**
- **IETF Standards Track**
- **Well suited to VoIP, Internet Gaming, etc.**
- **Sees very little use**

2-minute stretch

Network Address Translator

NAT

- **Network Address Translator**



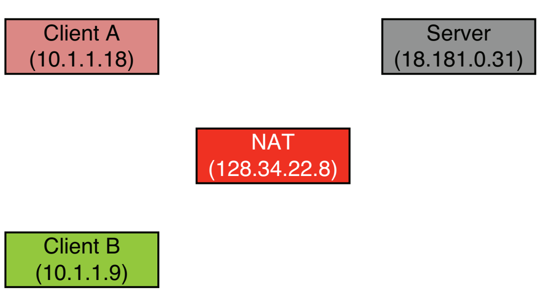
Motivations and Complications

- **There are only 2^{32} IP addresses**
- **Firewalls for security**
- **Breaks end-to-end (node does not know its external IP)**
- **Node might not even know if it's behind a NAT**
- **NAT needs to be able to dynamically assign mappings**

How a NAT Works

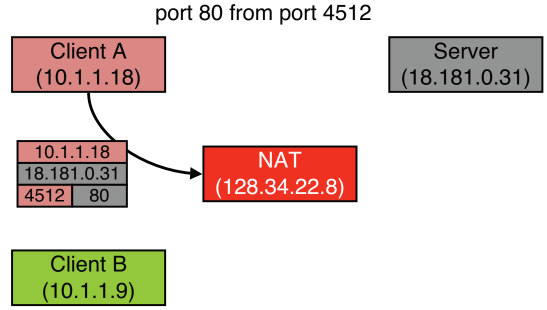
- **Maps between global and local (IP,port) pairs**
- **Requires knowledge of transport packet format**
- **UDP datagram, TCP SYN**
  - Can shut down TCP mapping on FIN+ACK
  - UDP requires timeouts ($>$ 2 minutes, unless IANA says otherwise)
- **RFC 4787/BCP 127 defines recommended behaviors**

NAT Example, Step 1



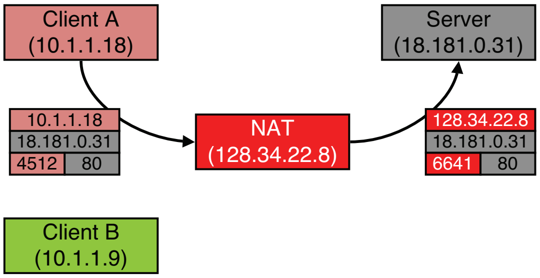
NAT Example, Step 2

Client A tries to open
a connection to Server
port 80 from port 4512



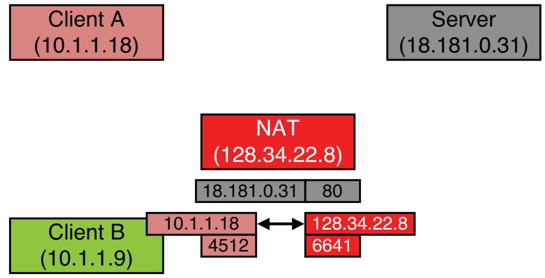
NAT Example, Step 3

NAT rewrites the source
address and port so it
seems it is the source



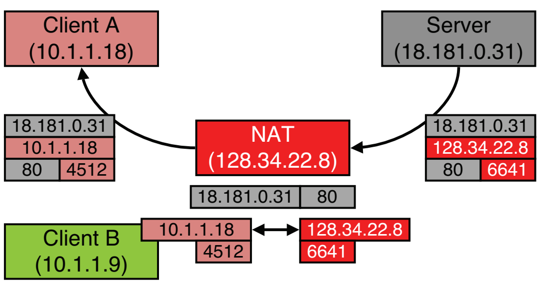
NAT Example, Step 4

NAT also creates a mapping
so it can transform future
packets correctly



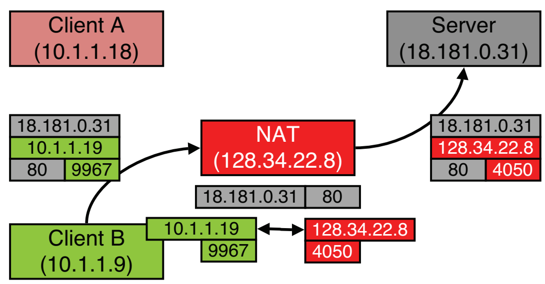
NAT Example, Step 5

Server responds to SYN
with SYN+ACK and NAT
rewrites it



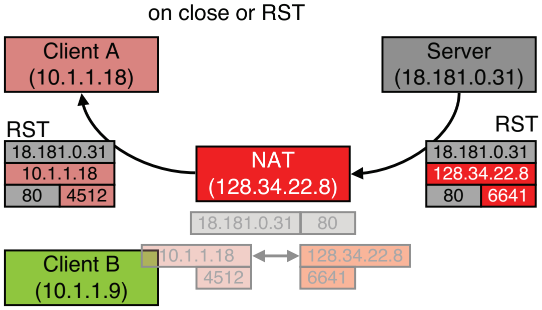
NAT Example, Step 6

Client B can also open
a connection, and the NAT
creates a mapping for it too



NAT Example, Step 7

In the case of TCP, NAT
can discard the mapping
on close or RST

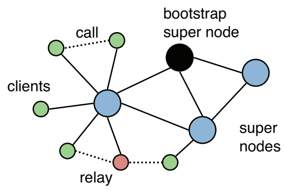


Types of NAT (RFC 3849)

- Full Cone: no ingress filter (single local-external mapping)
- Restricted Cone: ingress filter on address
- Port Restricted: ingress filter on address/port
- Symmetric: different mappings for different external destinations
- Teminology is imperfect (static port mappings, etc.)

NAT Problems

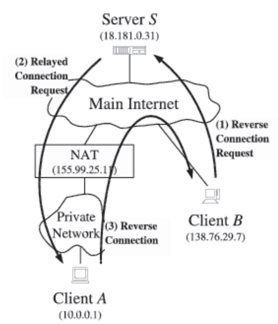
- Incoming TCP connections
- E.g., Skype



TCP Through NATs

- Server socket doesn't initiate traffic: NAT can't set up mapping
- Rendezvous servers (as in Skype)
- Connection reversal through rendezvous if only one is behind a NAT (rendezvous server asks un-NAT node to open a port so NAT node can connect)

TCP Reversal



More NAT Problems

- Port mapping: 0-1023 should map to 0-1023
- Port parity: even port → even port, odd port → odd port (RFC 3550: RTP uses even, RTCP uses odd))
- Hairpinning: packet from inside NAT can send packets to external (NAT) address and have portmap work

RFC 4787 and Hairpinning

- "NAT UDP Unicast Requirements"
  - F. Audet and C. Jennings (guest lecture last year)

```
More formally, a NAT that supports hairpinning forwards packets
originating from an internal address, X1:x1, destined for an external
address X2':x2' that has an active mapping to an internal address
X2:x2, back to that internal address, X2:x2.  Note that typically X1'
is the same as X2'.

Furthermore, the NAT may present the hairpinned packet with either an
internal (X1:x1) or an external (X1':x1') source IP address and port.
Therefore, the hairpinning NAT behavior can be either "External
source IP address and port" or "Internal source IP address and port".
"Internal source IP address and port" may cause problems by confusing
implementations that expect an external IP address and port.
```

Example confusion

- Two nodes (A, B) share a switch behind a NAT
- A sends TCP SYN to an external address for B
- B responds with SYN+ACK
- What can happen if NAT did not rewrite A's SYN to have an external address and port? Will B's packet traverse the NAT?

Simple Traversal of UDP through NATs

STUN (RFC 3849)

- **"Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)"**
- **Enables a node to**
  - Determine if it is behind a NAT, and if so, what kind
  - Obtain a public IP address/port pair
- **Client-server protocol, requires no changes to NATs**
- **STUN server coordinates**

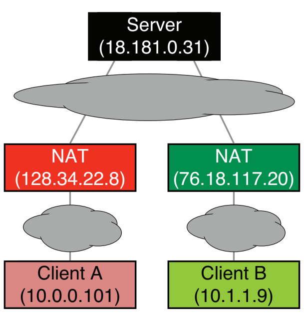
STUN Binding Requests

- **Node sends BR to STUN server**
- **STUN sends a response that has the address and port it sees**
  - If different than node's local address and port, it's behind a NAT
- **Node can probe to see what kind of NAT**
  - Ask STUN server to respond from different address: no response, address restricted; different response, symmetric
  - Ask STUN server to respond from different port: no response, port restricted
- **When does STUN not work?**

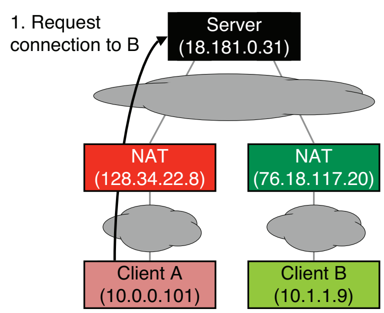
NAT Hole-Punching

- **STUN doesn't work when**
  - A or B are behind symmetric NAT
  - A and B are behind same NAT without hairpinning
  - What about restricted cone, port-restricted NATs?
- **Parallel approaches**
  - A and B report their local IP address to server Sx
  - Server tells the other the address/port pair (L, G)
  - A tries to send UDP packets to (L_B, G_B) using L_A
  - B tries to send UDP packets to (L_A, G_A) using L_B

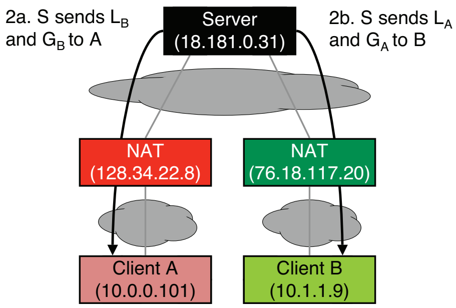
NAT Hole-Punching Example



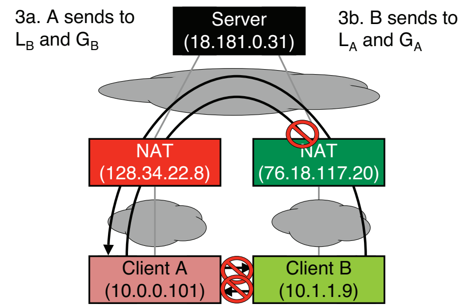
NAT Hole-Punching Example



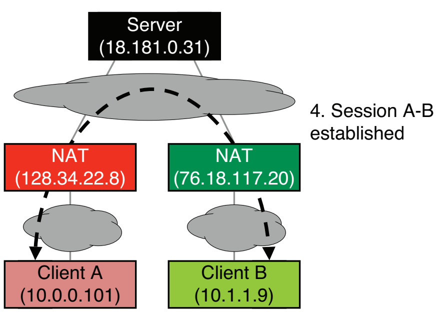
NAT Hole-Punching Example



NAT Hole-Punching Example

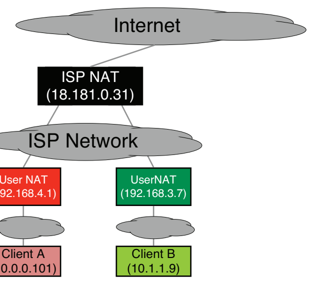


NAT Hole-Punching Example



Multiple NAT Layers

- **Common for consumer Internet: ISP has internal NAT, end user places another NAT**
- **Requires hairpinning at ISP NAT**



The New Hourglass