Covert Channels inside DNS

Lucas Nussbaum

Introduction

On many networks, to access the Internet:

- you have to pay (airports, train stations, hotels)
- or you have to log in (universities networks)

Only service available to all users: DNS

Question

How can one access the whole Internet (without logging in) using only DNS?

DNS - Domain Name System

- “phone book” for the Internet
- UDP, port 53

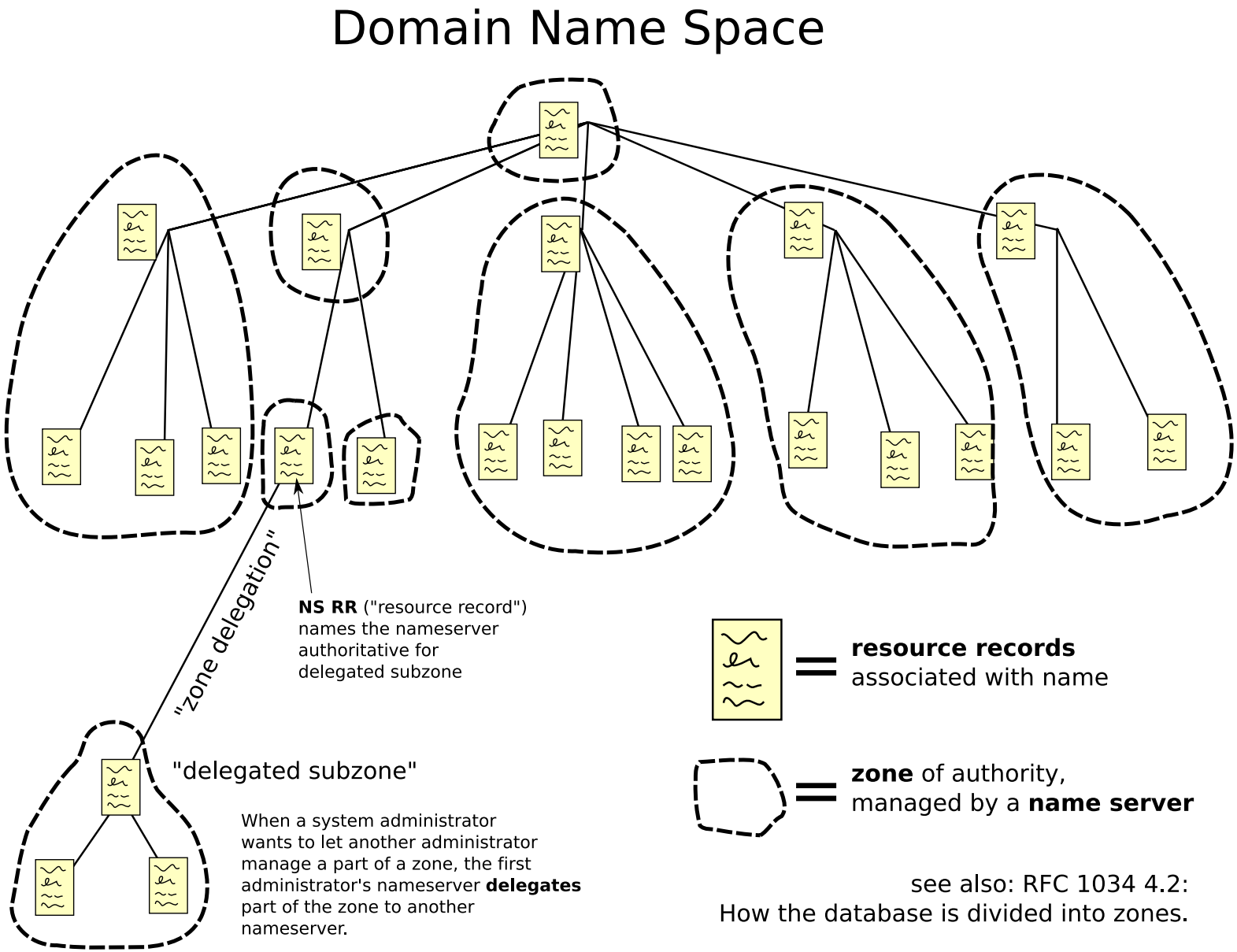
Types of records:

- A: karok.imag.fr ⇒ 129.88.69.11
- CNAME: www-id.imag.fr ⇒ karok.imag.fr
- TXT: text
- ...

$ dig -t txt imag.fr

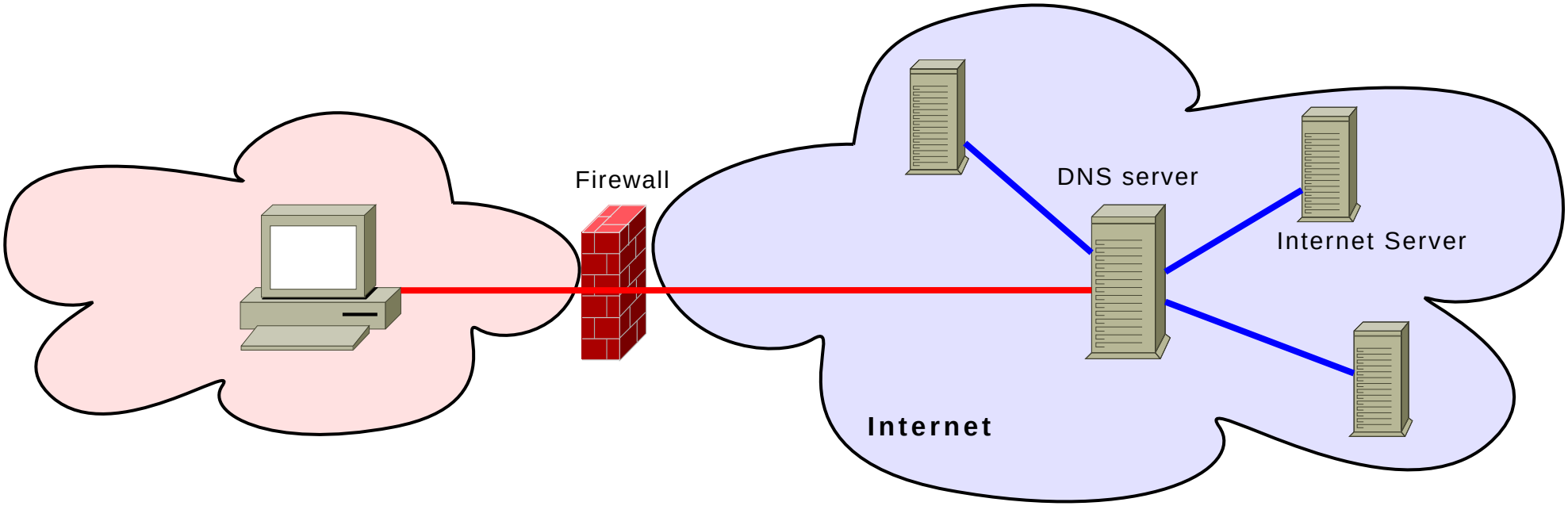
```
Institut d'Informatique et de Mathematiques Appliquees
IMAG  BP 53  F-38041 GRENOBLE Cedex 9 (France)
or IMAG 46 Av. Felix Viallet  F-38031 GRENOBLE Cedex 1
v=spf1 mx ip4:129.88.34.204 ?all
```

DNS - Hierarchy and Delegation



Tunneling data inside DNS - Principle

- encapsulation of data in DNS packets
- use a rogue DNS server on the Internet to decode data and send it to other servers



Covert Channels

Definition

parasitic communication channel that draws bandwidth from another channel in order to transmit information without the authorization or knowledge of the latter channel's designer, owner or operator.

- Storage channels - communicate by modifying a stored object
- Timing channels - transmit information by affecting the relative timing of events

Examples: Steganography, IP over HTTP, IP over ICMP

Covert Channels inside DNS - implementations

NSTX:

- IP over DNS
- uses base64 to upload
- uses TXT records without encoding to download
- unmaintained, buggy code (segfaults, mem leaks)
- splits IP packets in chunks

OzymanDNS:

- encapsulates a TCP stream in DNS packets (typical use: SSH over DNS)
- implements a TCP-like machine to reorder/retransmit packets
- uses TXT and EDNS0
- crashes very frequently; not able to run performance measurements

Covert Channels inside DNS - implementations

Iodine:

- better codebase than NSTX
- uses base32 or base64
- uses NULL records and EDNS0
- splits IP packets into chunks

TUNS

Features:

- IP over DNS
- uses only CNAME records (both for upload and download)
- doesn't split IP packets, reduces MTU instead (1-1 mapping)
  no need to bother with TCP state machine, reordering / retransmitting
- prototype written in Ruby (perf. problems with pNet::DNS!)
- tries to be RFC-compliant

TUN/TAP

- virtual network interfaces
  - tap: layer 2 (ethernet frames)
  - tun: layer 3 (IP packets)
- userspace application can associate with them and send/receive packets to/from the OS network stack
- used by openvpn, vtun, bochs, qemu, virtualbox
- available on all common platforms (incl. MS Windows)

Base32? Base64?

Goal: encode binary stream as text

- Base32: 32 values (5 bits) per character [A-Z][2-7]
- Base64: 64 values (6 bits) per character [A-Z][a-z][0-9]-+
- DNS (RFC 1034):

```
<label> ::= <letter> [ [ <ldh-str> ] <let-dig> ]
<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>
<let-dig-hyp> ::= <let-dig> | "-"
<let-dig> ::= <letter> | <digit>
<letter> ::= any one of the 52 alphabetic characters A through Z in
upper case and a through z in lower case
<digit> ::= any one of the ten digits 0 through 9

Note that while upper and lower case letters are allowed in domain
names, no significance is attached to the case.  That is, two names
with the same spelling but different case are to be treated as if
identical.
```

Base32? Base64?

- Base64: 64 characters
- DNS: 63 characters [A-z][a-z][0-9]-

nstx: [A-Z][a-z][0-9]-_

Sending and receiving data

Sending from client to server: easy.

- Just encapsulate data in a DNS packet as soon as it is received.

Receiving from server: harder.

- Client polls server with nearly empty requests.
- Server answers with data.

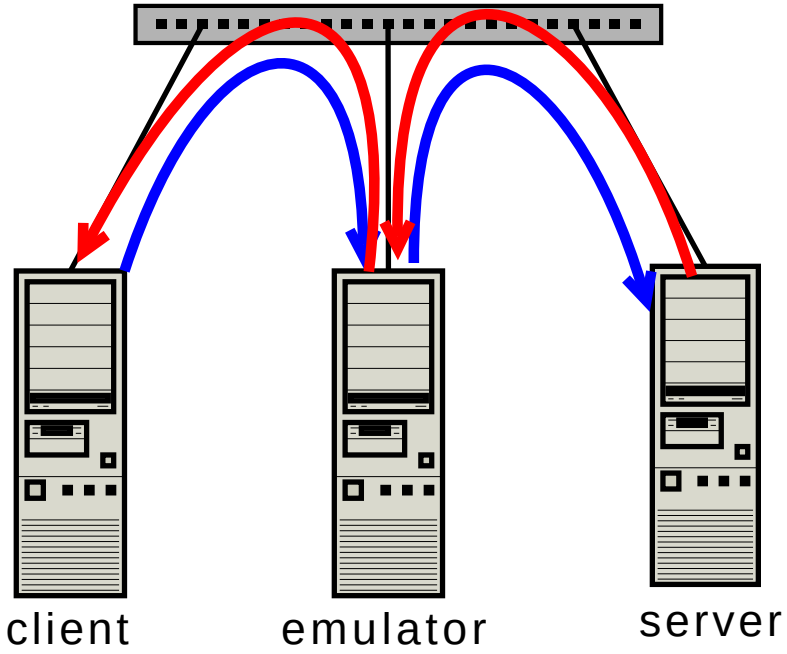
Improvement:

- Server keeps requests for a little while if there's nothing to send.

Performance?

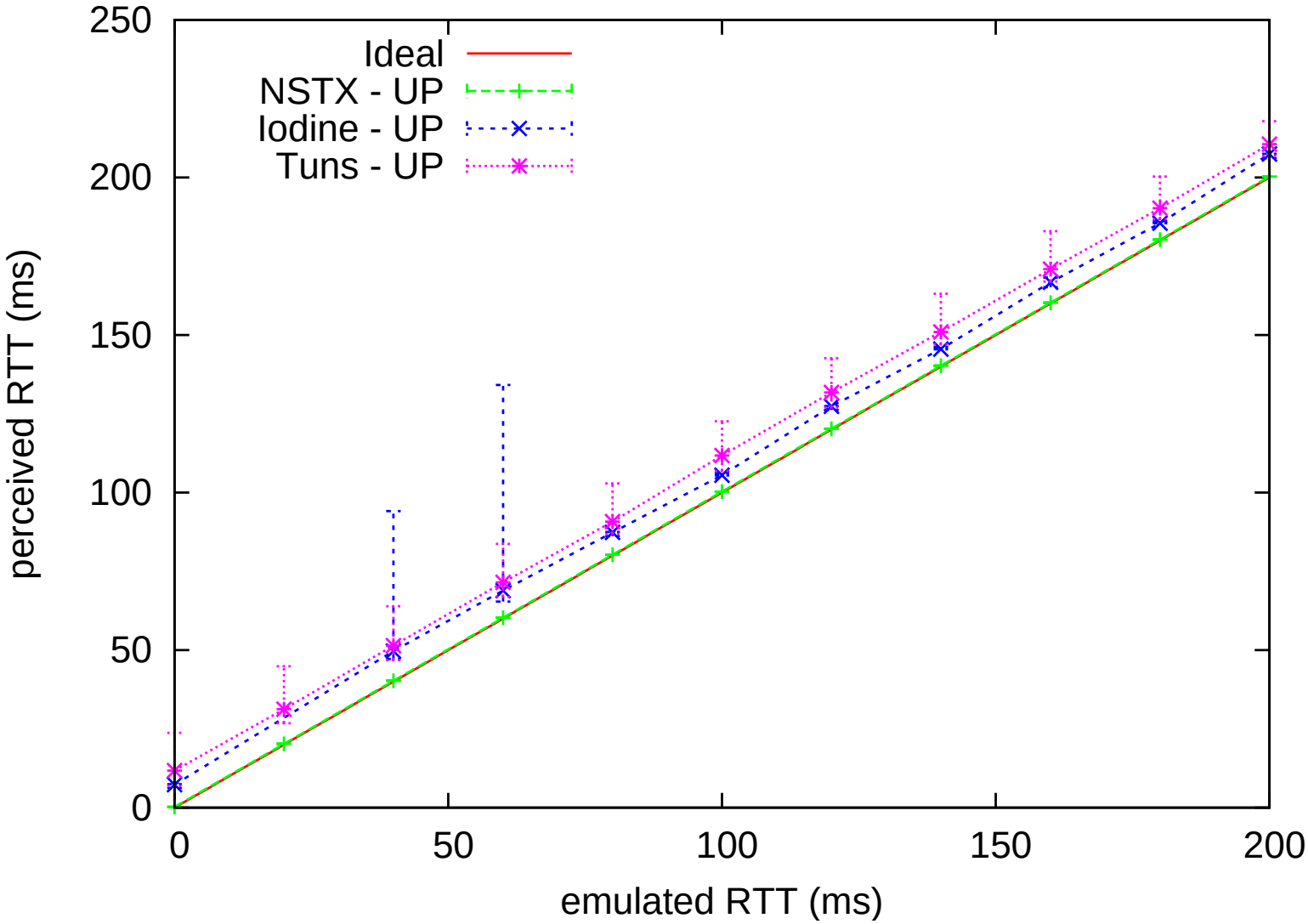
- Compare NSTX, Iodine, TUNS (OzymanDNS is too buggy)
- Latency?
- Bandwidth?
- How does performance evolve when the network conditions degrade?

Experimental setup

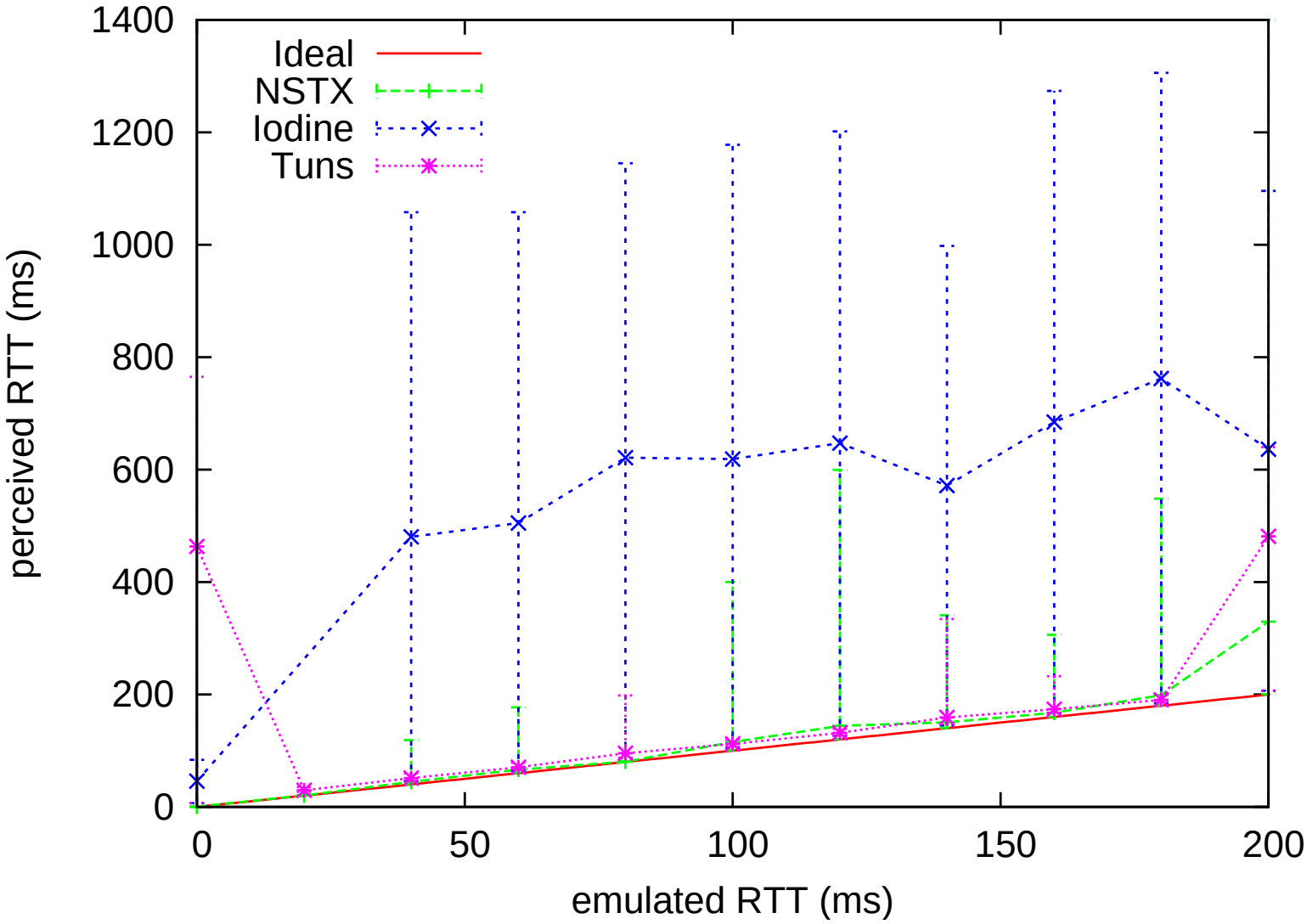


- Using 3 Grid'5000 nodes (Helios)
- Running on Linux 2.6.22 - i686, not amd64
- Using Linux TC/Netem for network emulation
- Emulation parameters applied when packets exit the emulator node
- For all experiments: output bandwidth of emulator node limited to 1 Mbps

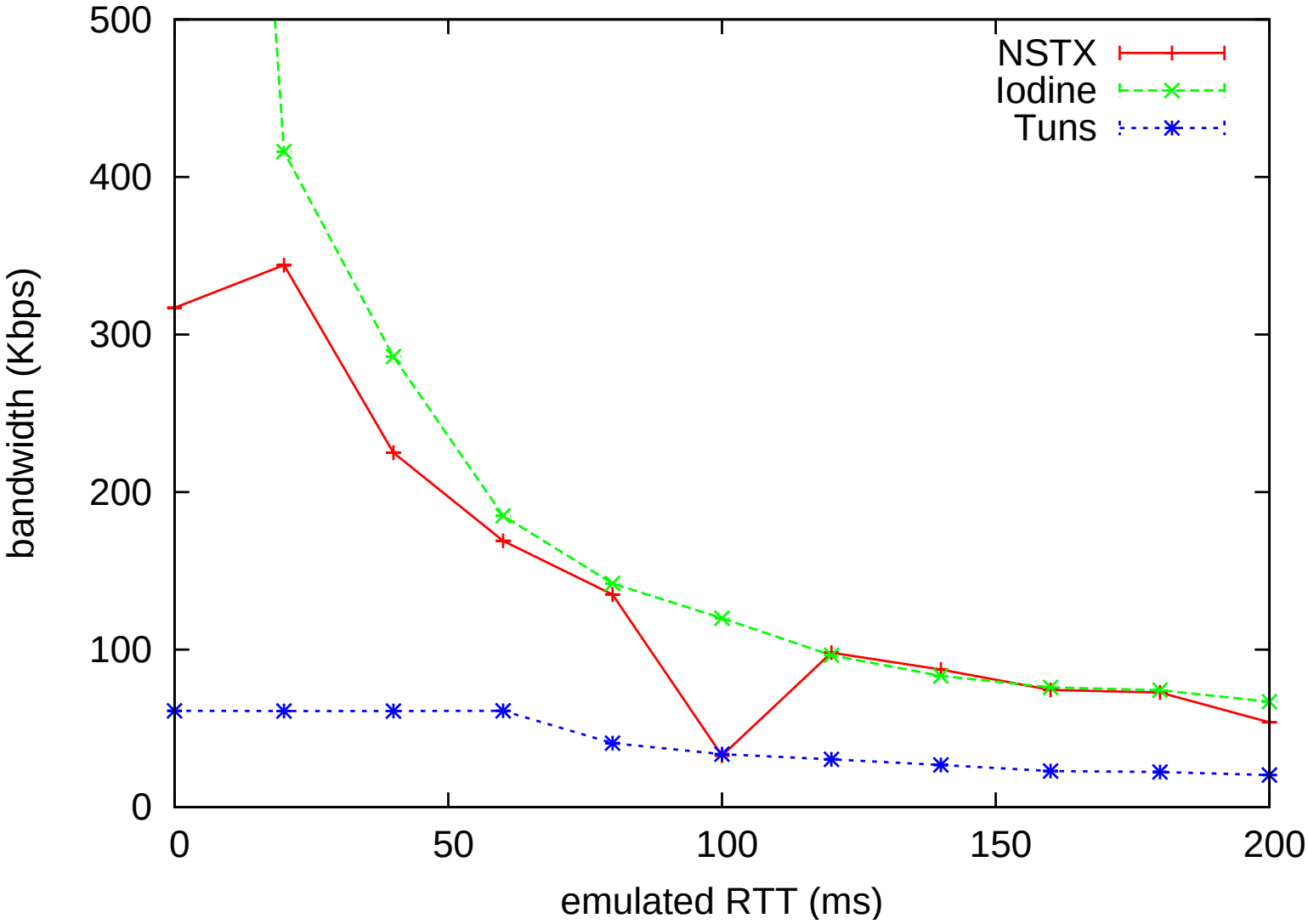
Latency - ping started by client (-i 1)



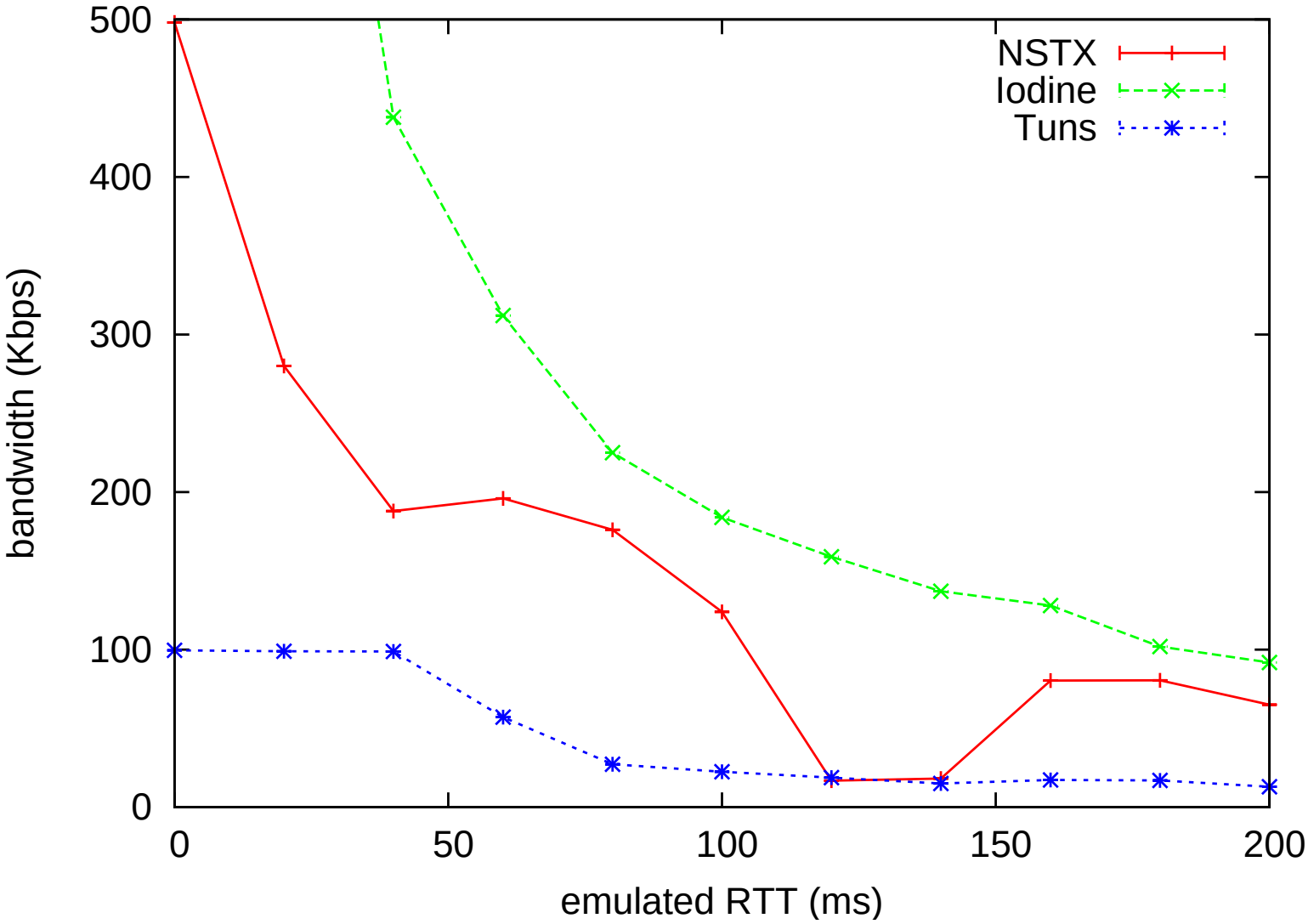
Latency - ping started by server (-i 1)



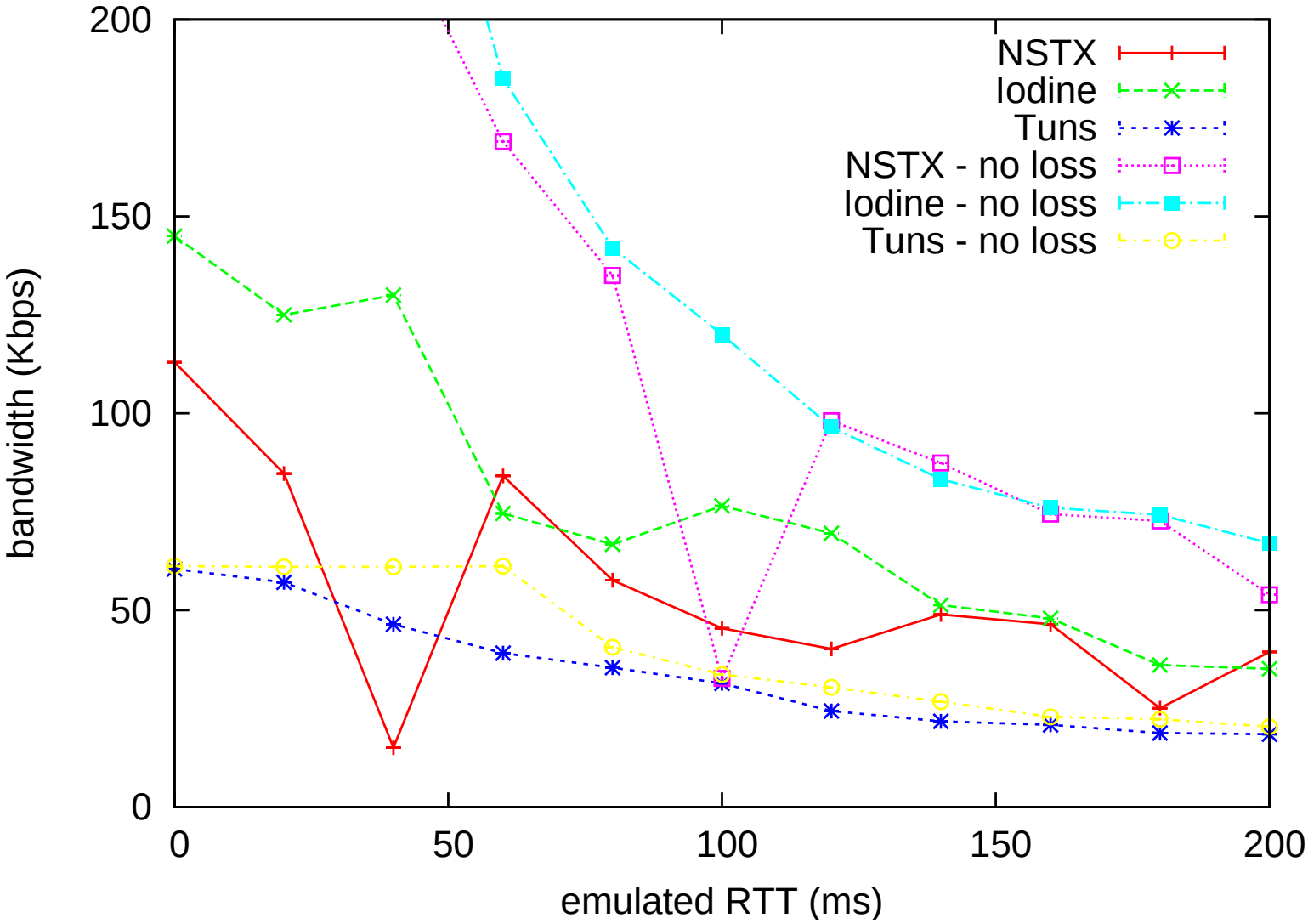
Bandwidth - upload (client to server)



Bandwidth - download (server to client)



Bandwidth under 2% loss - upload



Demo

- IMAG-VPN?
- Accessing the Internet from Grid'5000?

Conclusion

- IP over DNS works!
- KISS (Keep It Simple, Stupid) works
  - Tuns: about 700 LOC
- Performance?
  - Ruby: not as fast as C code (big surprise!)
    - pNet::DNS is SLOW
    - But good design can replace fast code (having both is better, of course)
- NSTX and Iodine perform a lot better
  - But often don't work in real networks
  - In experiments: no intermediate DNS servers

Legal issues

- Code pénal, article 323-1:
  Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 30000 euros d'amende.
  Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 45000 euros d'amende.
- Code pénal, article 323-3-1:
  Le fait, sans motif légitime, d'importer, de détenir, d'offrir, de céder ou de mettre à disposition un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour commettre une ou plusieurs des infractions prévues par les articles 323-1 à 323-3 est puni des peines prévues respectivement pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.