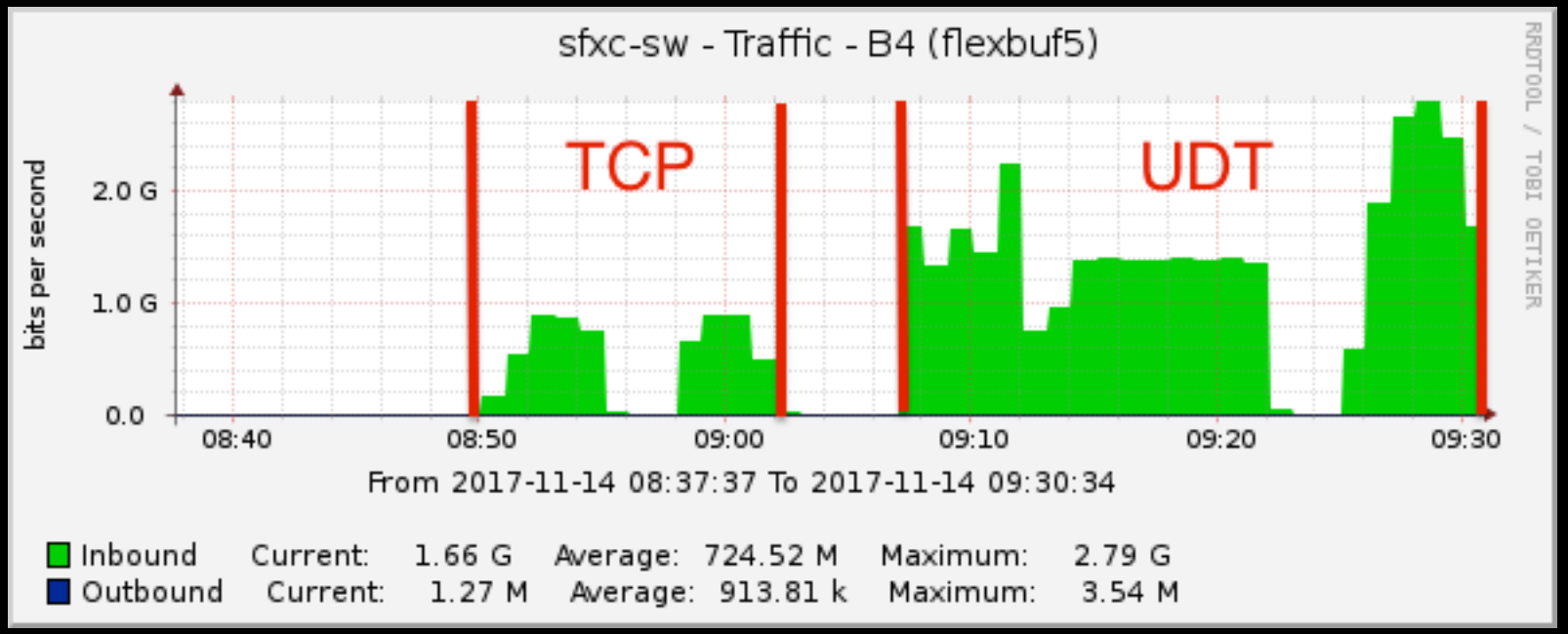
```
$> scp server.atnf.au:hugefile .
...
```

```
$> scp server.asterics.au:hugefile .
...
```

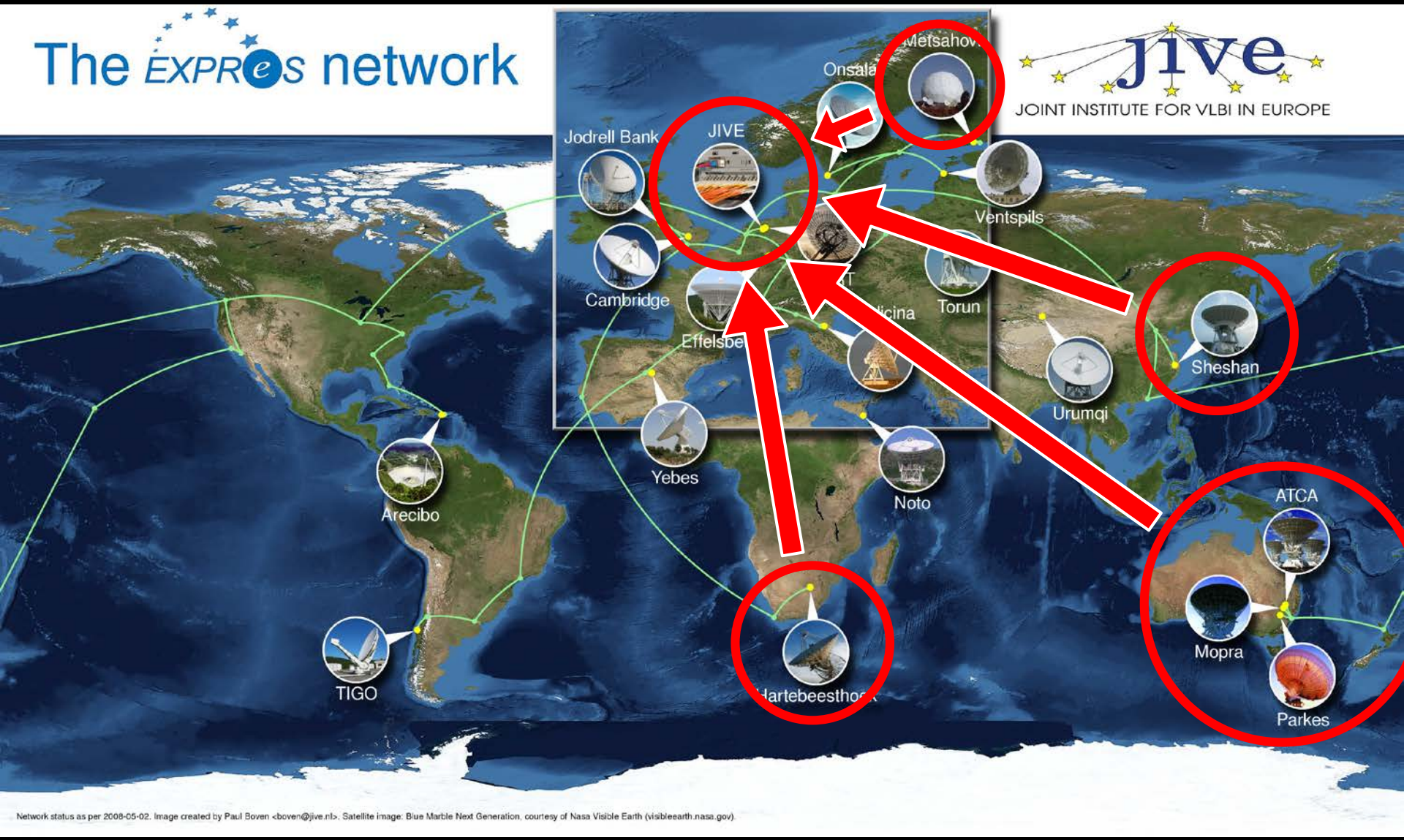
```
$> ftp server.asterics.au:hugefile .
...
```



https://github.com/jive-vlbi/etransfer

Thanks!

https://github.com/jive-vlbi/etransfer


The EXPReS network
JIVE
JOINT INSTITUTE FOR VLBI IN EUROPE
Jodrell Bank
JIVE
Onsala
Ventspils
Cambridge
Effelsberg
Torun
Yebes
Noto
Sheshan
Urumqi
Arecibo
TIGO
ATCA
Mopra
Parkes
Network status as per 2008-05-02. Image created by Paul Boven <boven@jive.nl>. Satellite image: Blue Marble Next Generation, courtesy of Nasa Visible Earth (visibleearth.nasa.gov)

The root cause is NOT the tool


≤ 140 characters

2006: Twitter starts, based on SMS

2006: Twitter starts, based on SMS

S(hort) M(essage) S(ervice)

- Length ≤ 160 characters

2006: Twitter starts, based on SMS

S(hort) M(essage) S(ervice)

- Length ≤ 160 characters
- 20 reserved for Twitter header

2006: Twitter starts, based on SMS

Send two SMSes in a row:

2006: Twitter starts, based on SMS

Send two SMSes in a row:

- no guarantee which will arrive

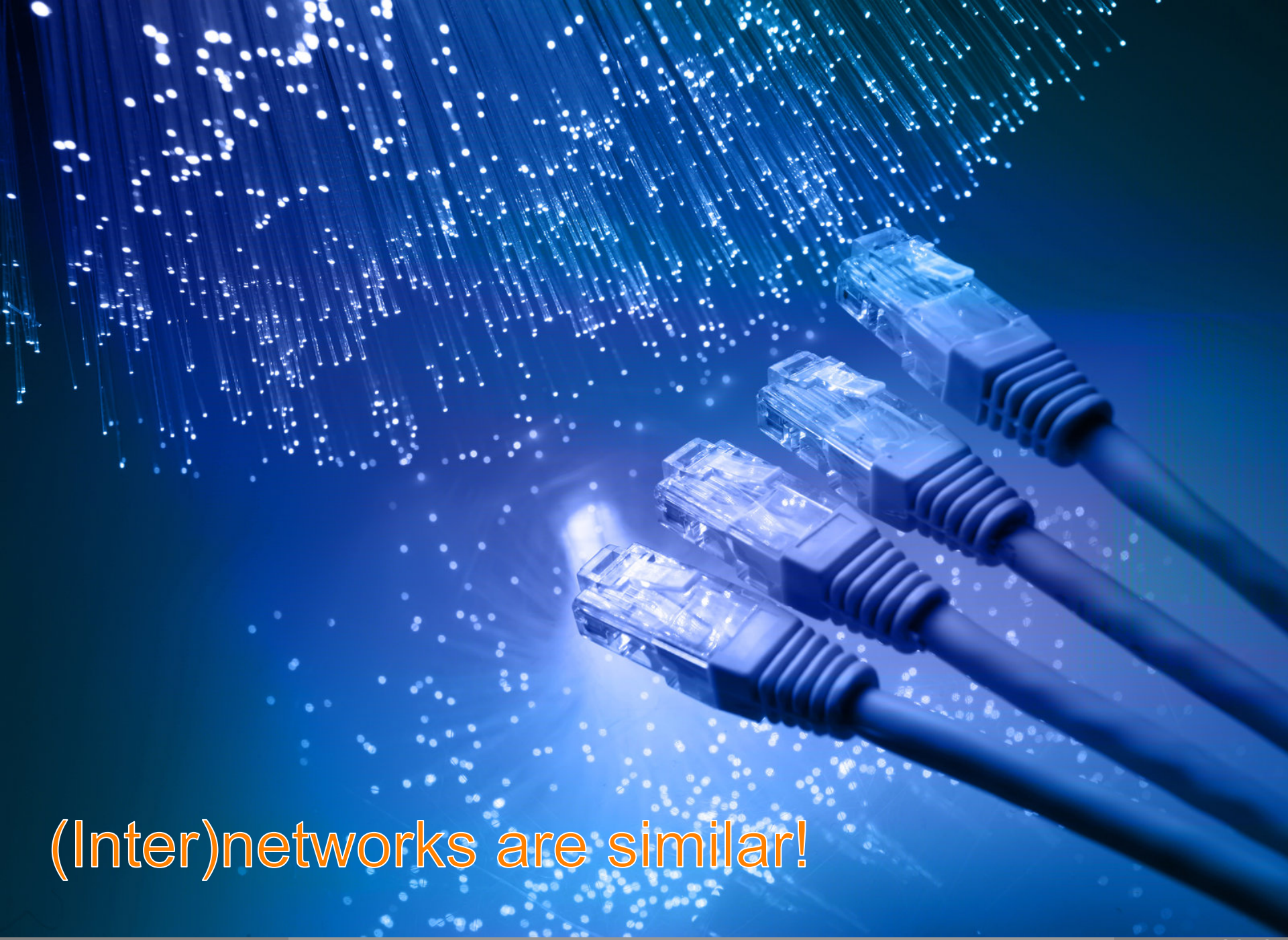
2006: Twitter starts, based on SMS

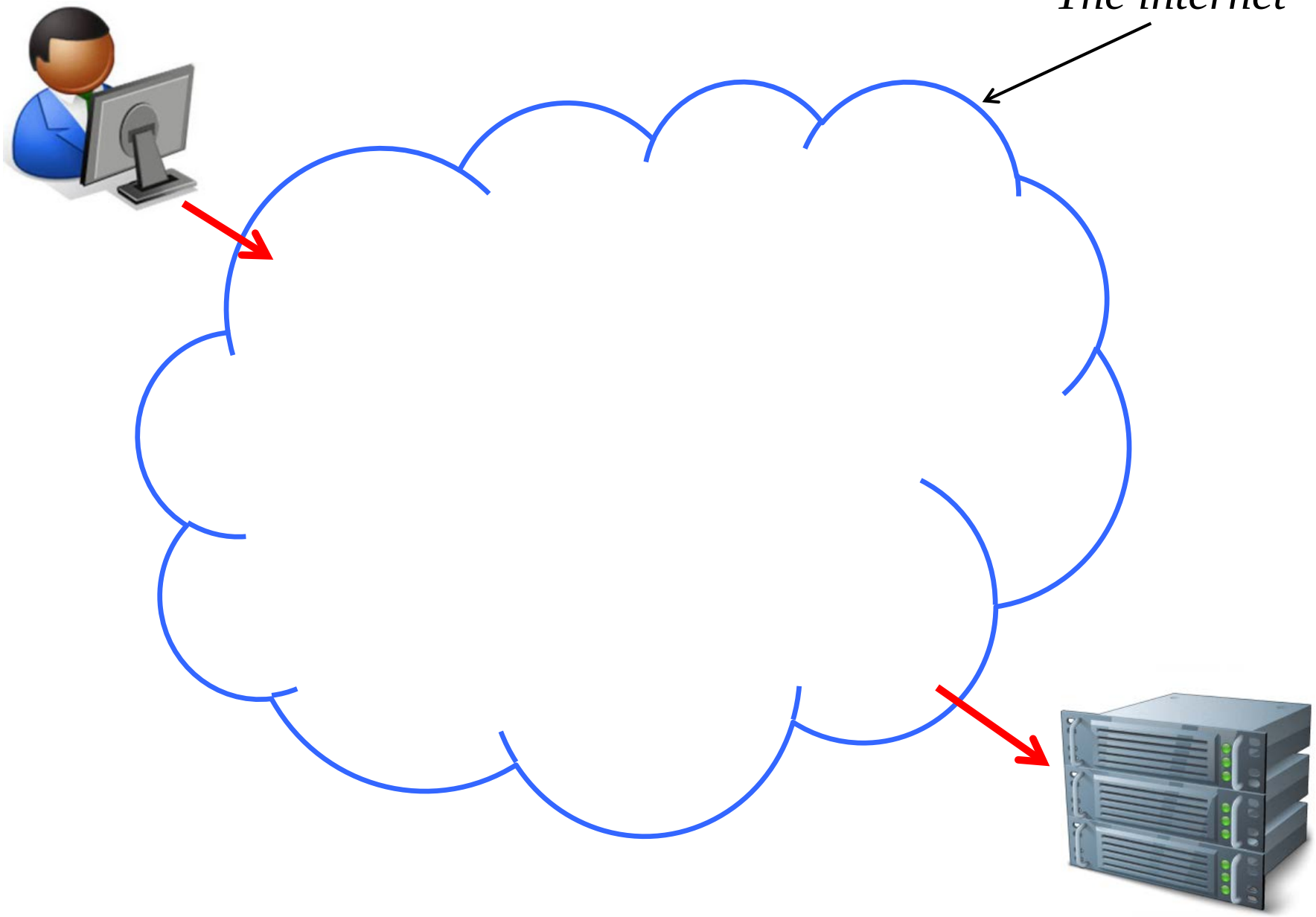
Send two SMSes in a row:

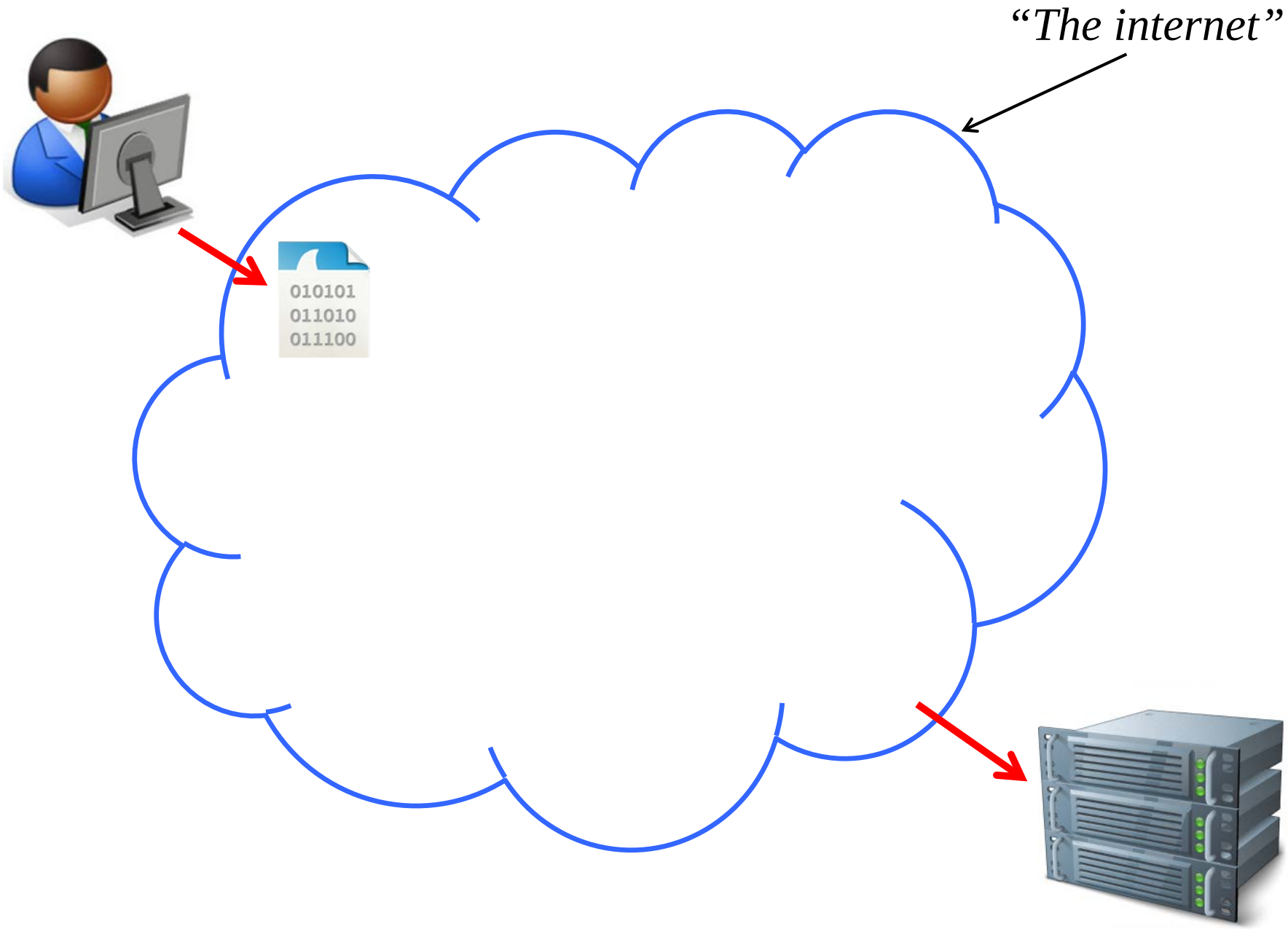
- no guarantee which will arrive
- no guarantee in which order

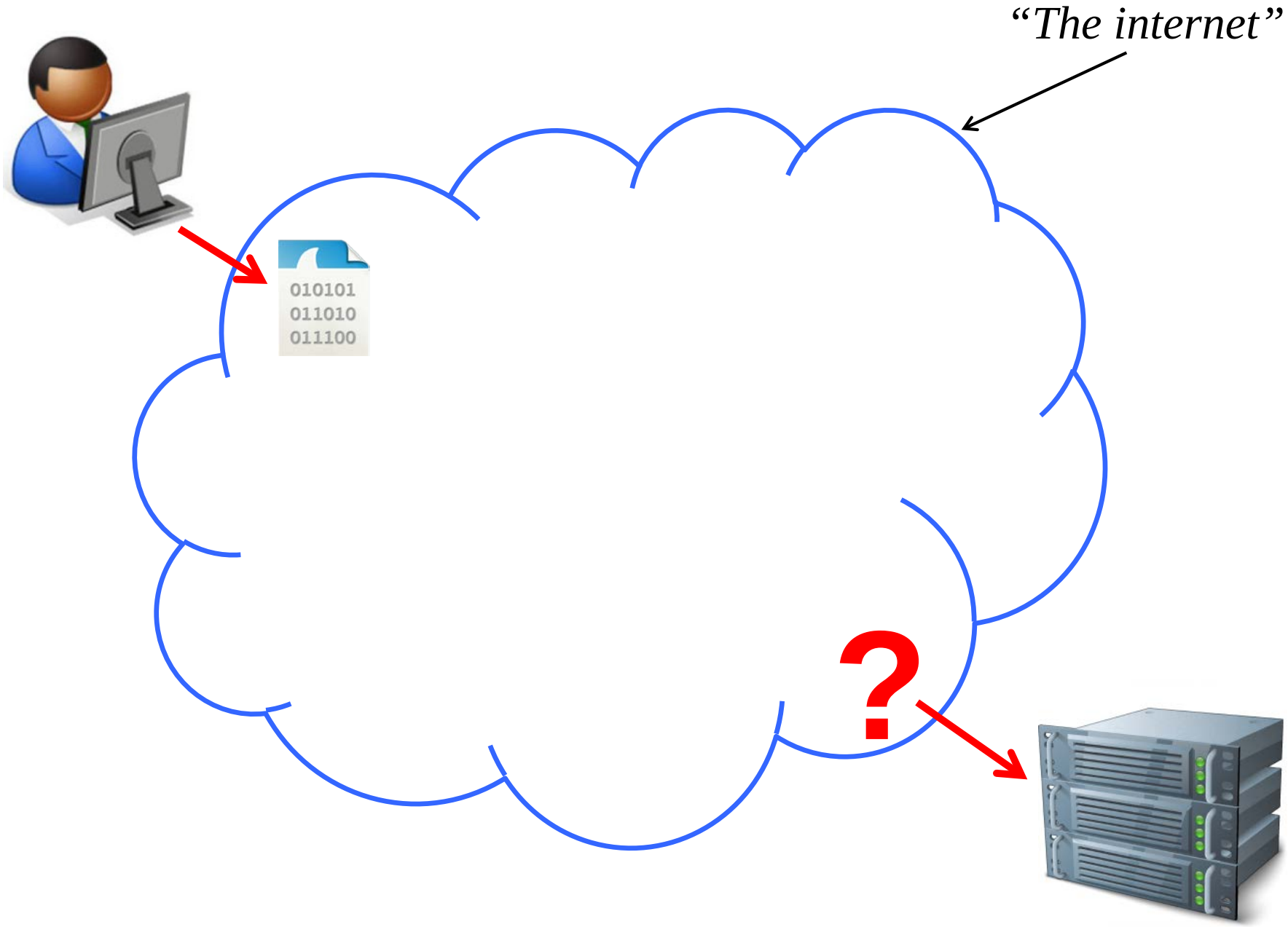
2006: Twitter starts, based on SMS

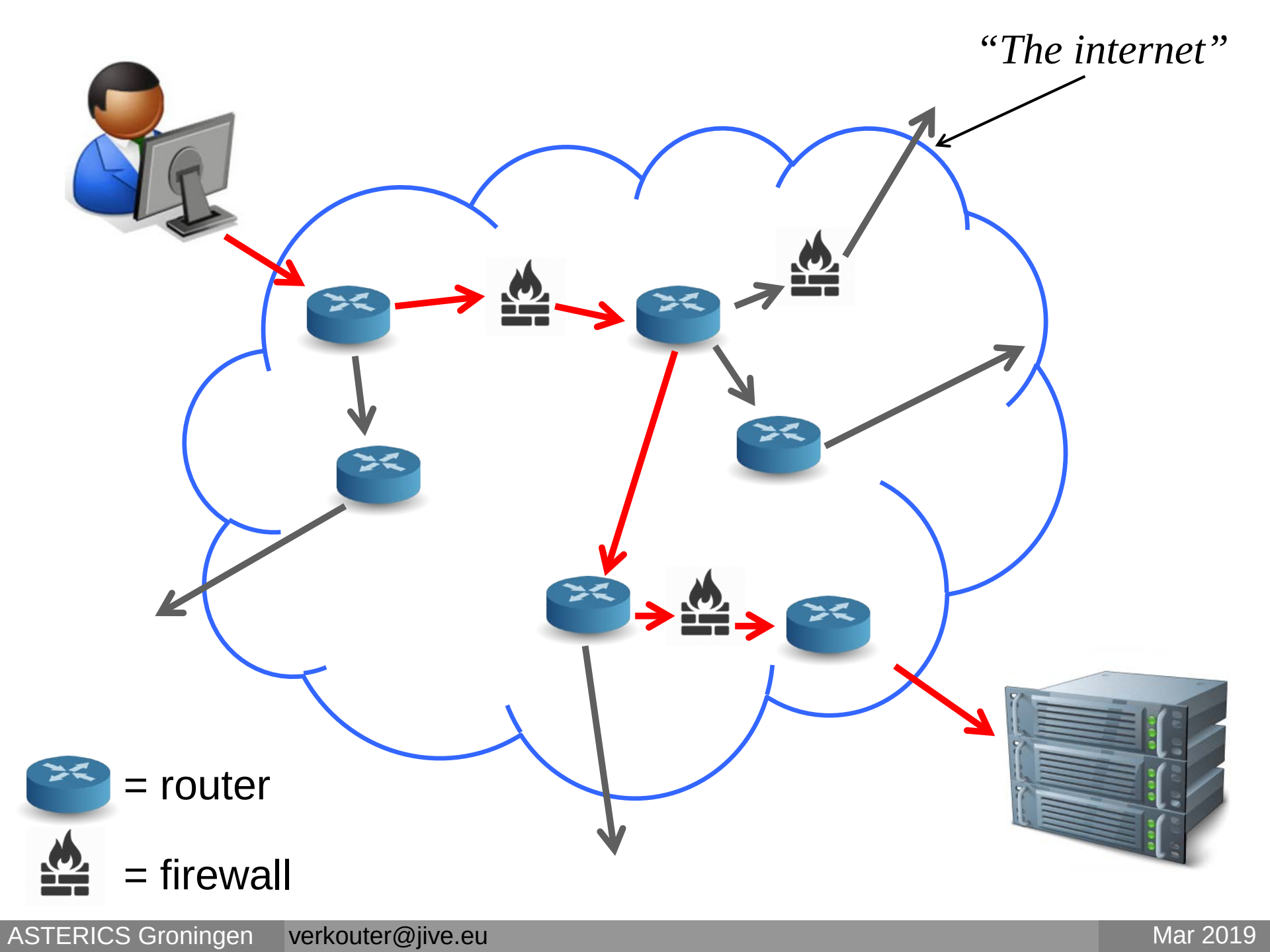
Can only send 140 characters atomically


(Inter)networks are similar!


“The internet”


"The internet"
010101
011010
011100


"The internet"
010101
011010
011100
?


“The internet”
= router
= firewall

Summarizing (Inter)network:

- packets dumped on network
- no guarantees
  - which will arrive
  - which order

RFC: 793

TRANSMISSION CONTROL PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

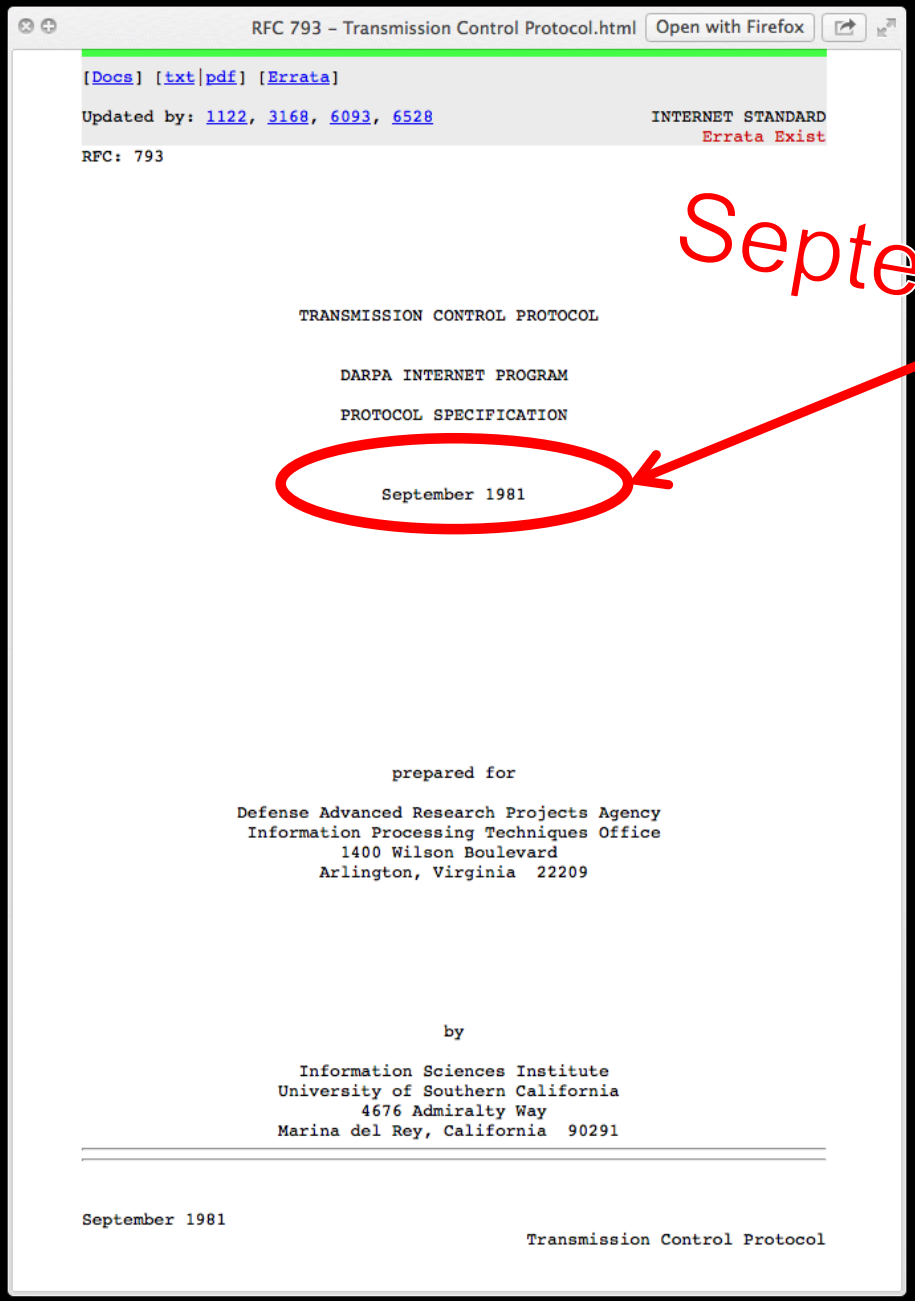
September 1981

prepared for

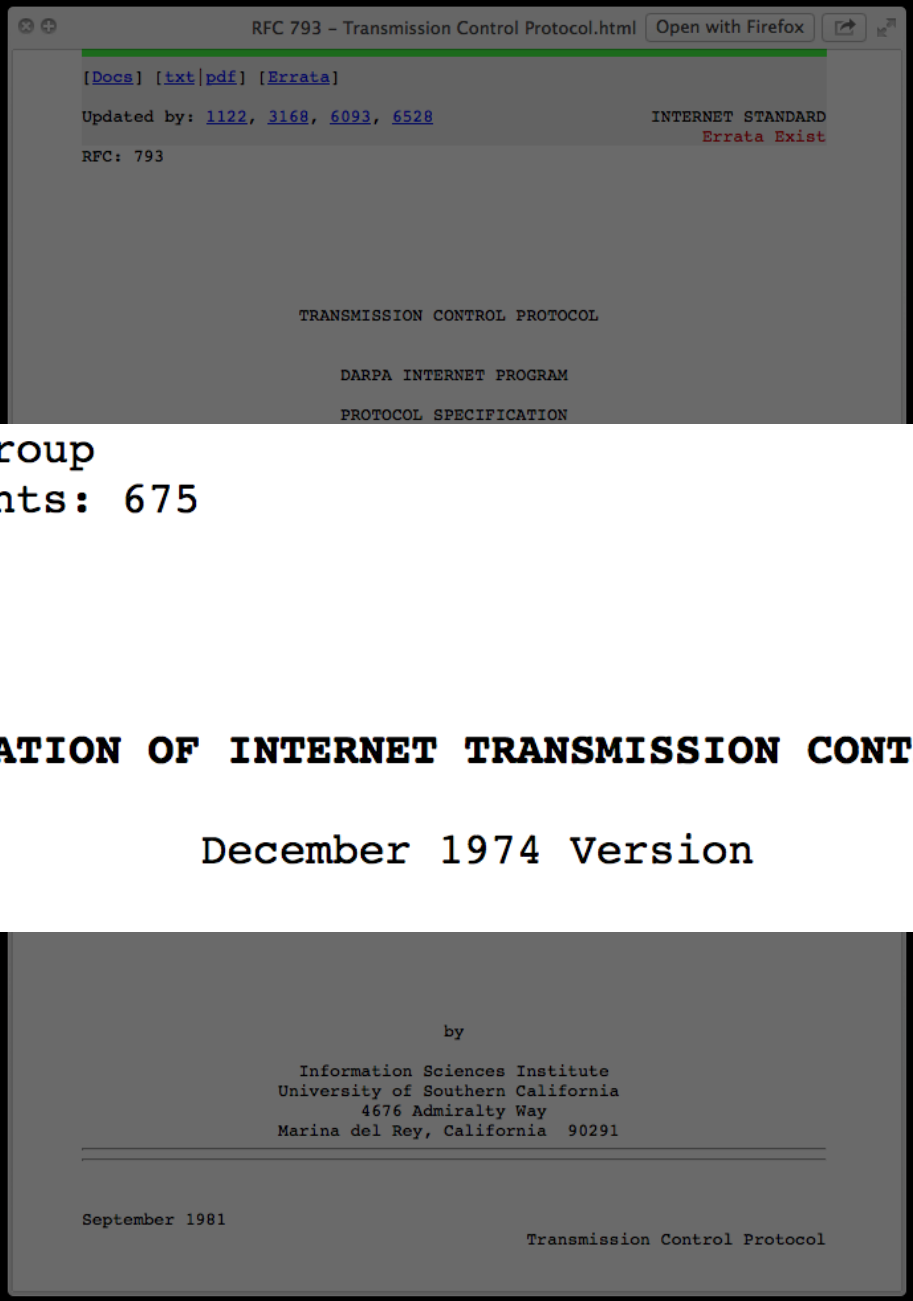
Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291



September 1981



Network Working Group — Vinton Cerf
Request for Comments: 675 — Yogen Dalal
NIC: 2 — Carl Sunshine
INWG: 72 — December 1974

SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM

December 1974 Version

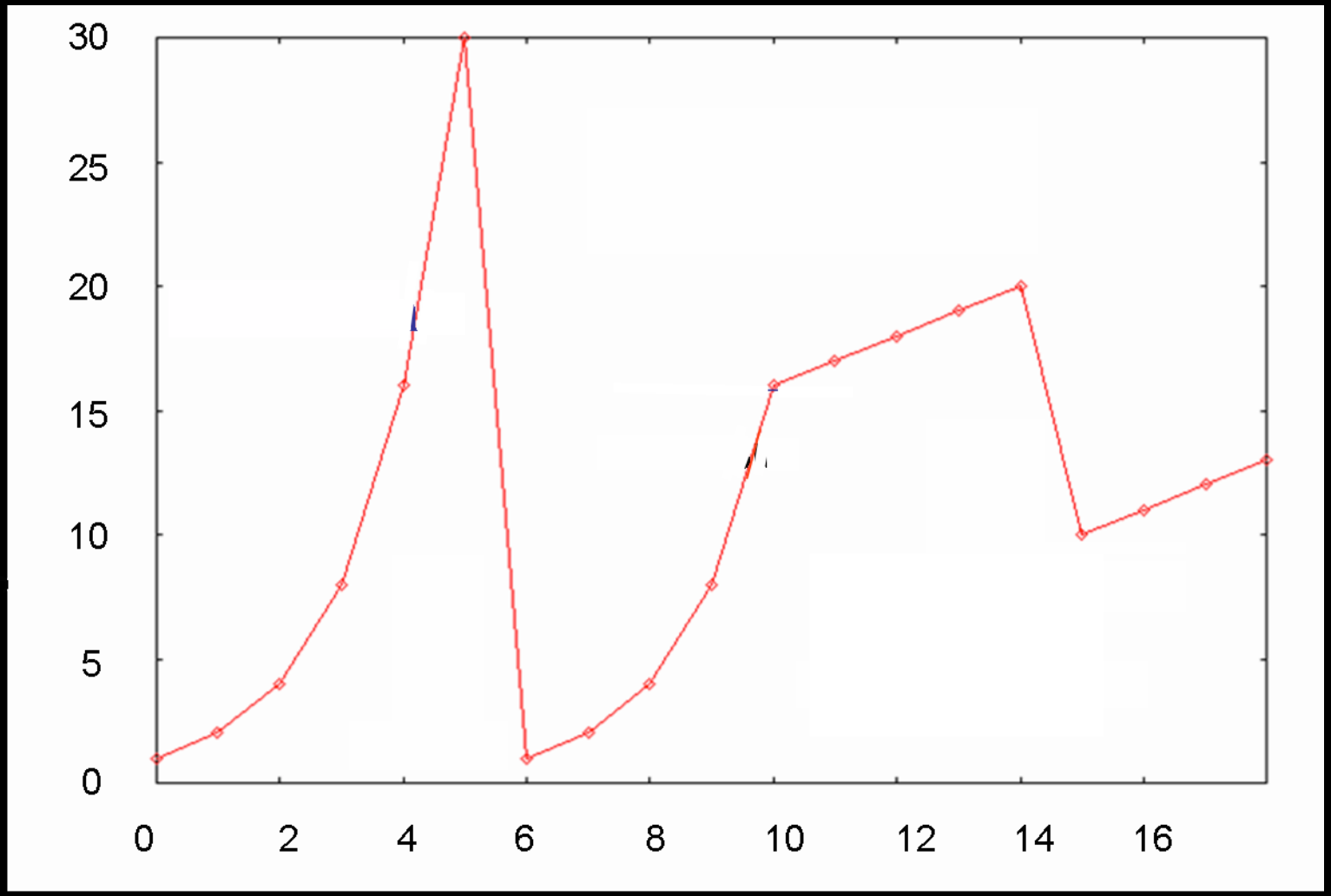
In those days:

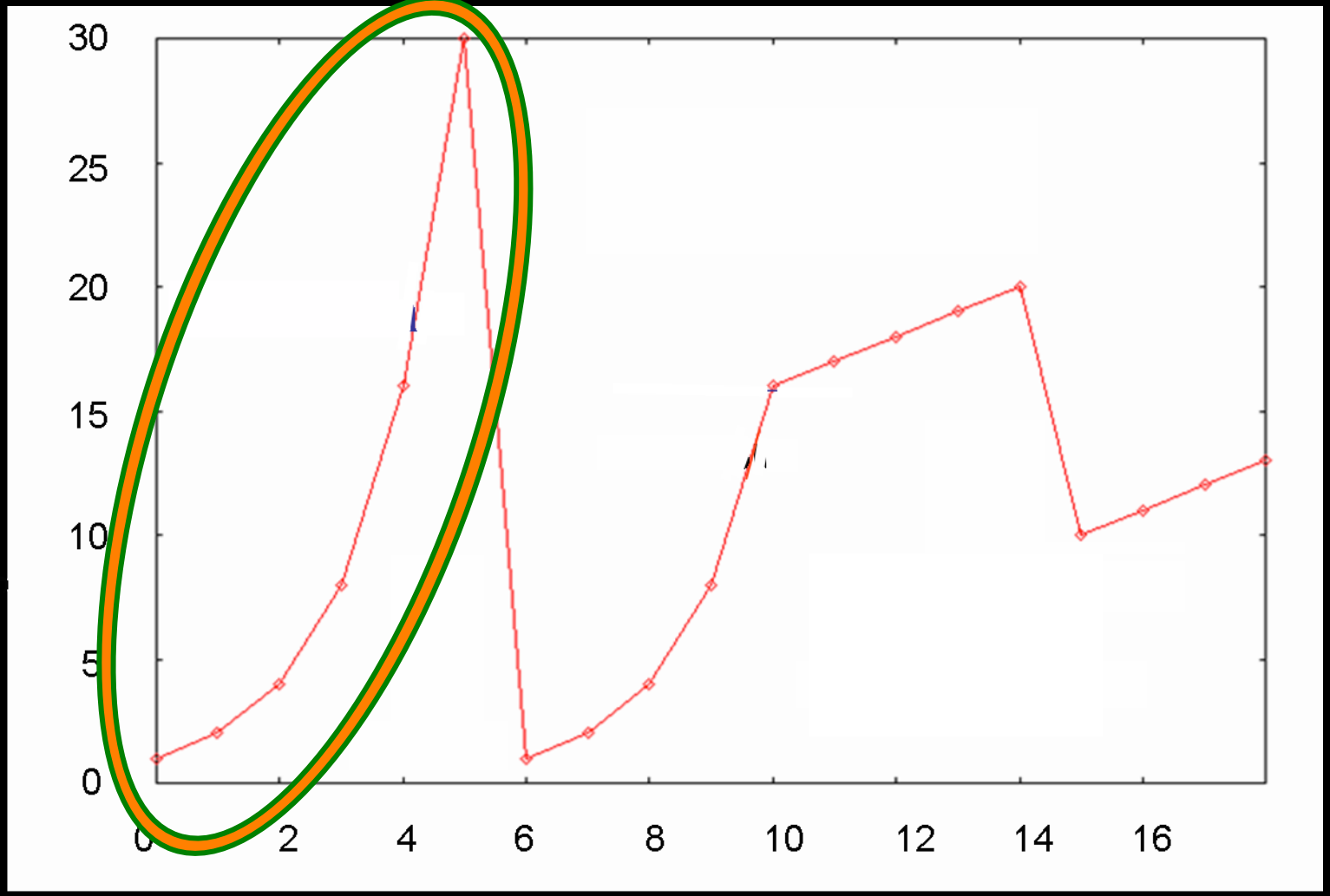
- 1500 bytes were a lot*
- protocol taxing on CPU + memory
- very sensitive to packet loss
  - assumption: network is congested
  - still true, btw
- type faster than your connection ...

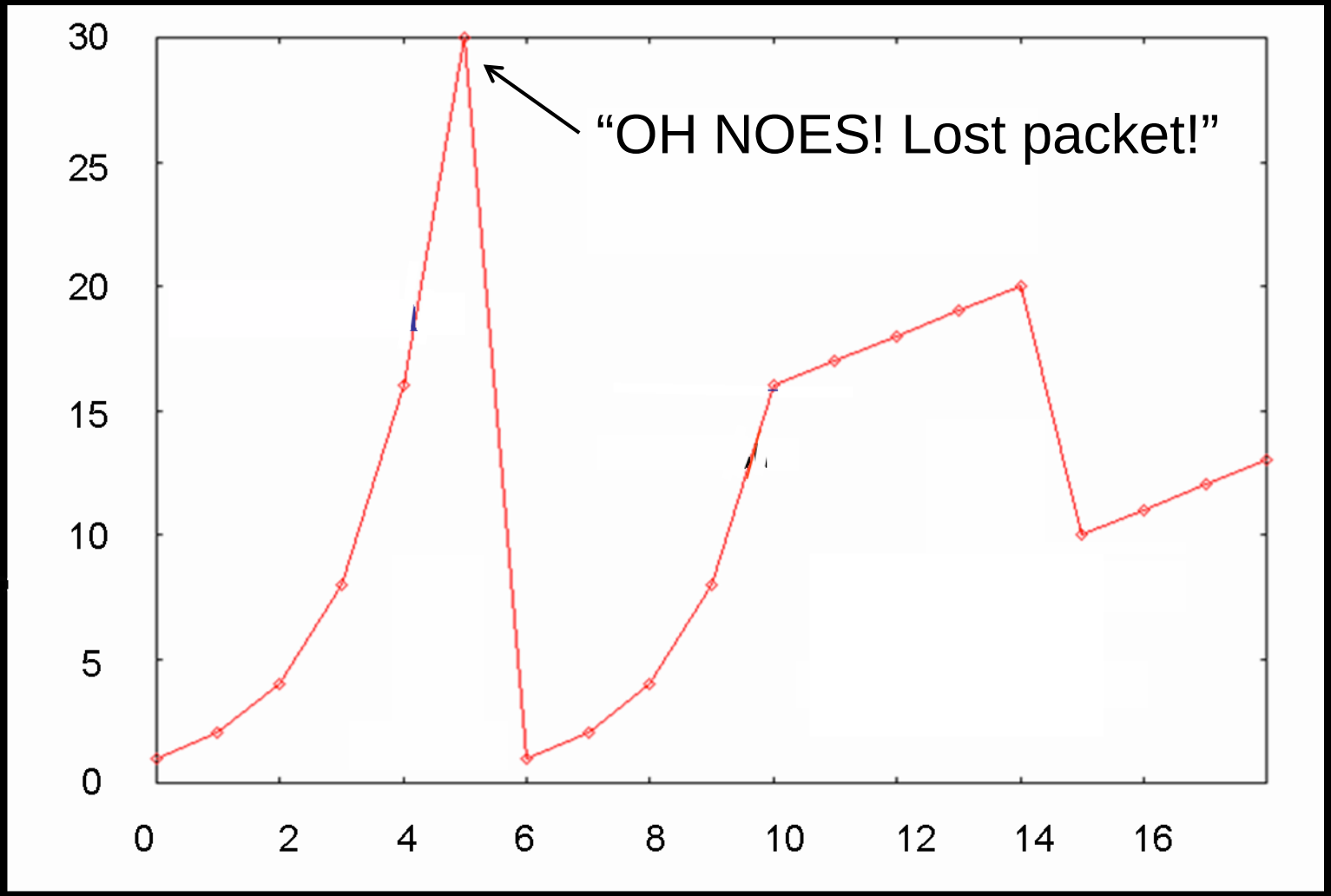
(*) ethernet Maximum Transmission Unit = ‘packet’

Due to design on
low speed, memory scarce systems

does not scale
linearly
to long, fat lines like global 10Gbps links


Speed
30
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
Time

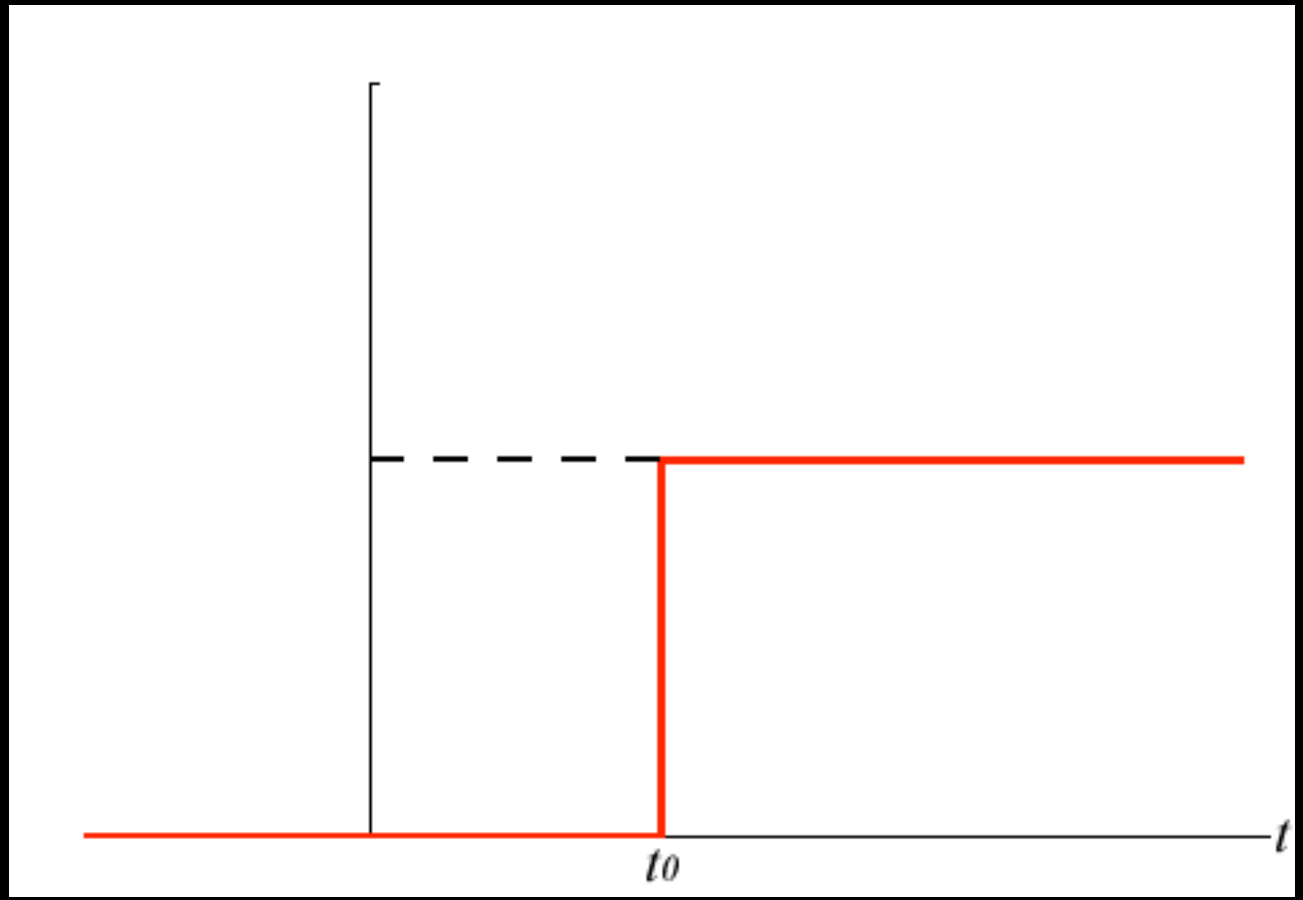

Speed
30
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
Time


Speed
30
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
“OH NOES! Lost packet!”
Time

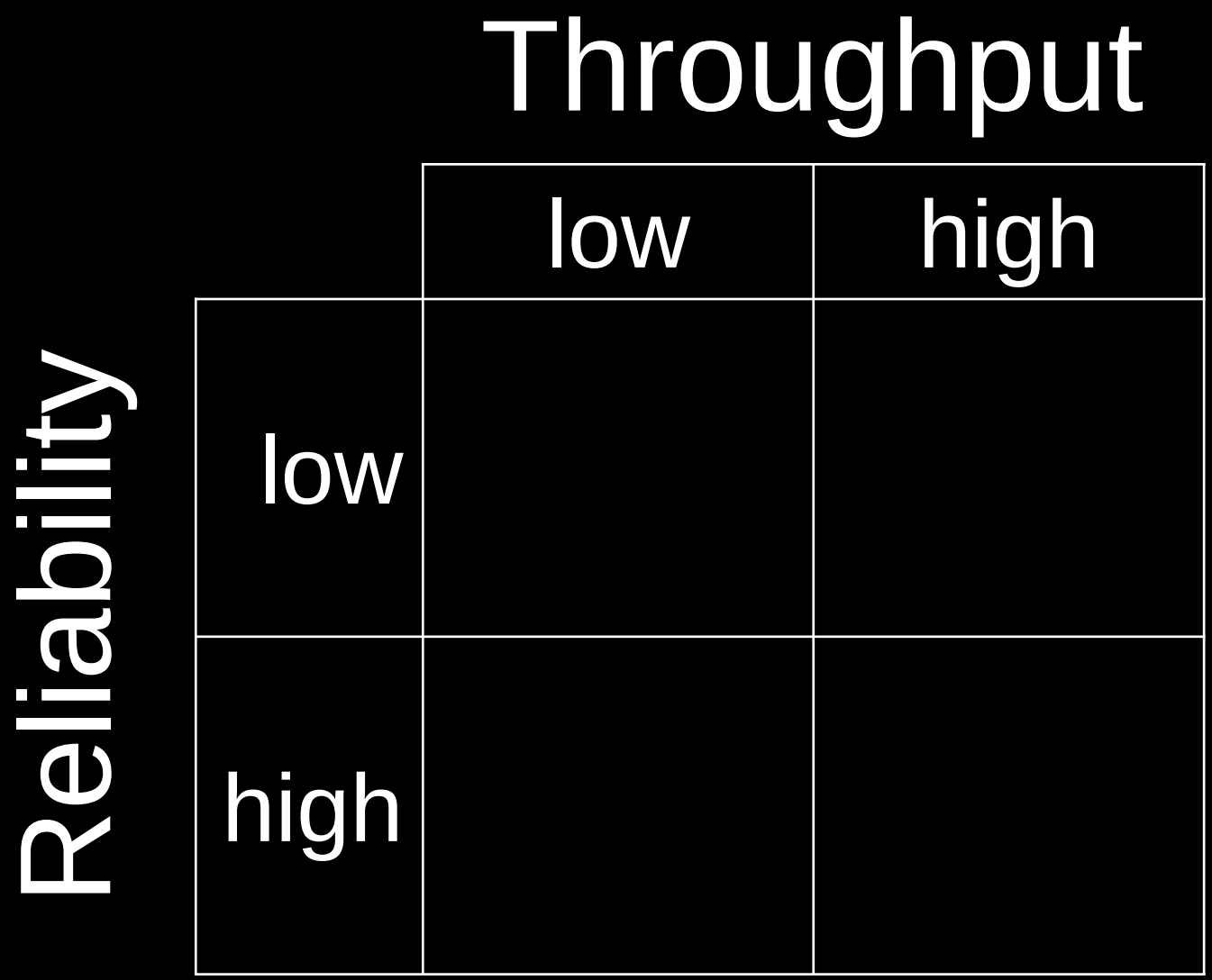
TCP designed to be polite

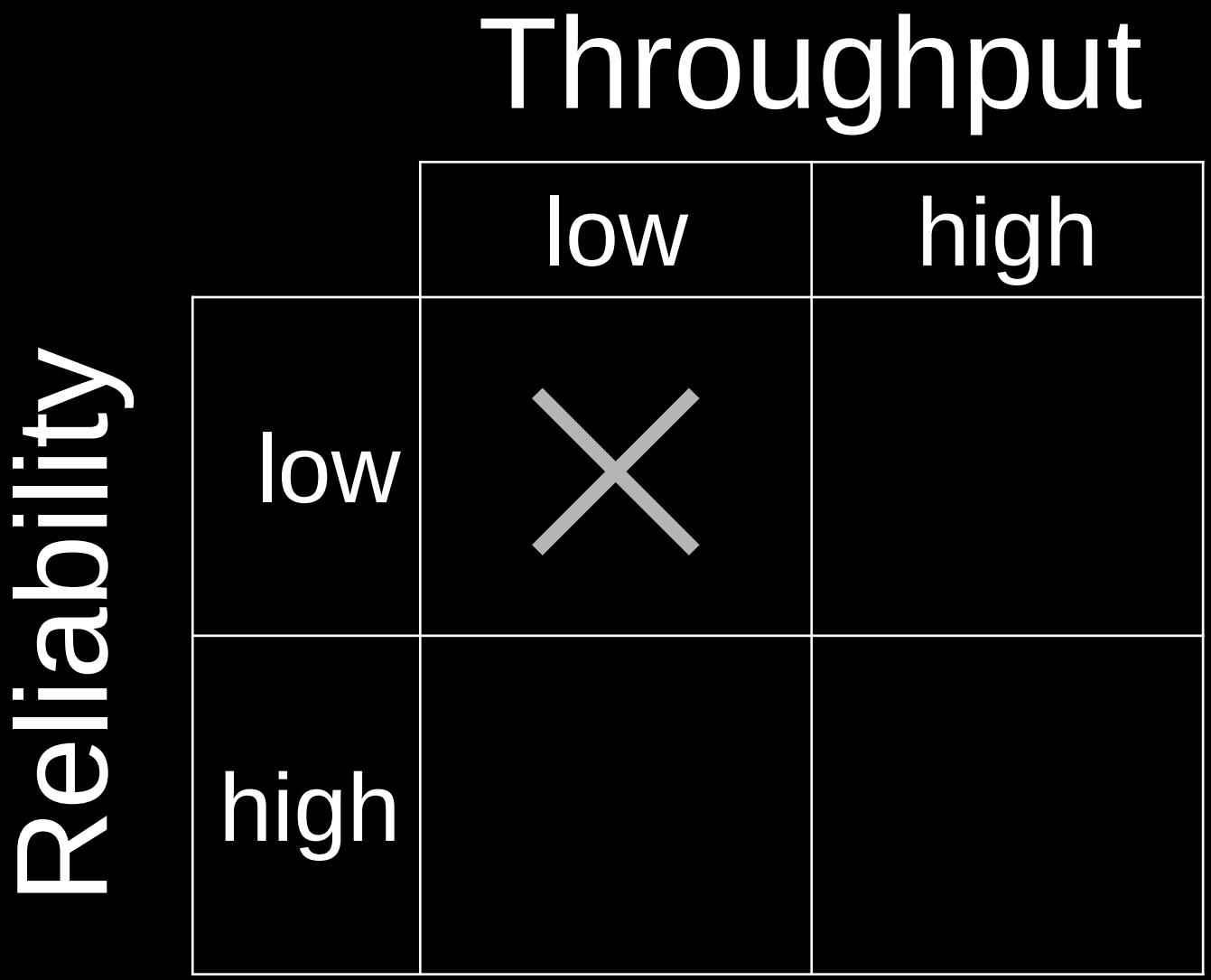
- packet lost? assume congestion
- back off by large amount

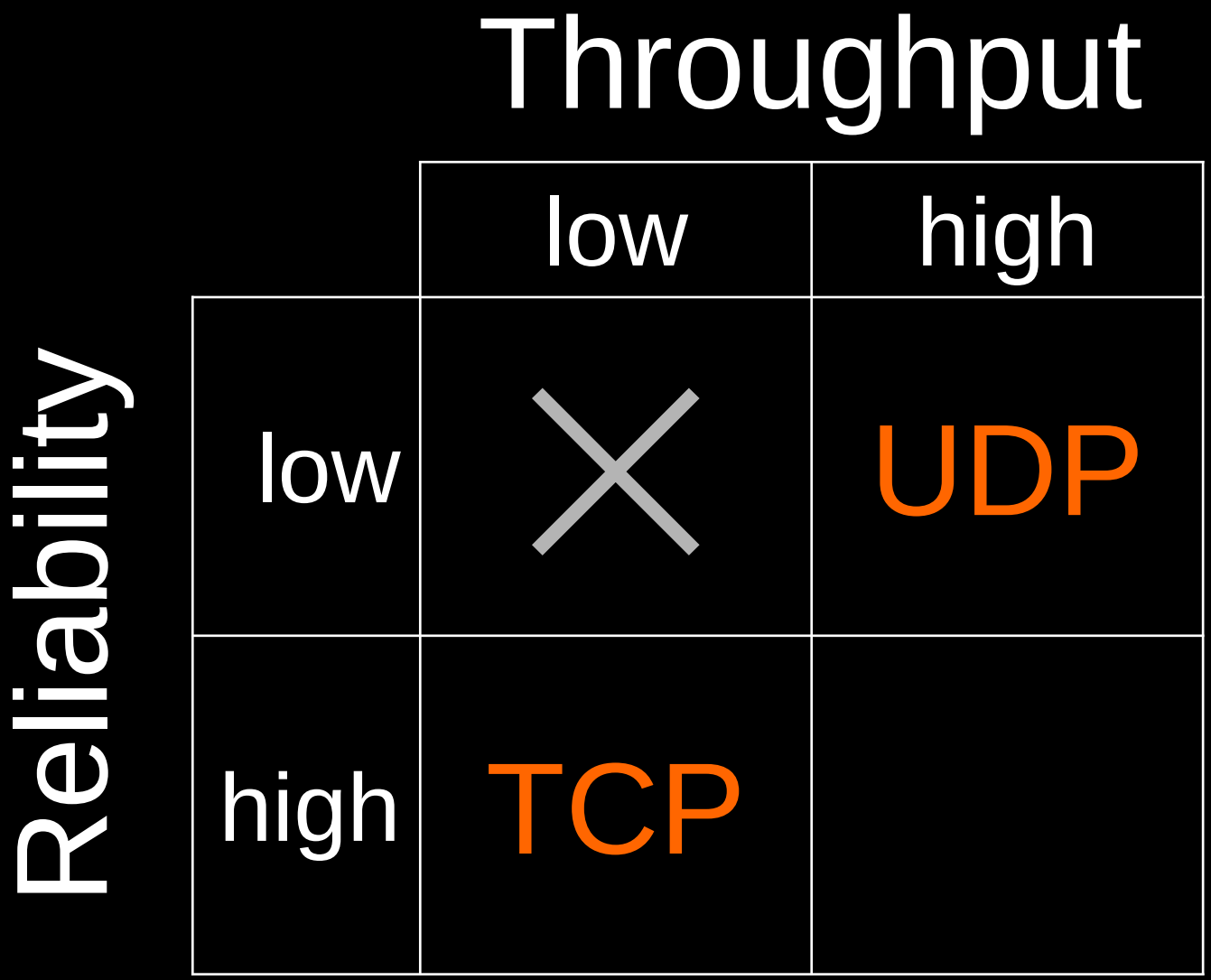
Speed

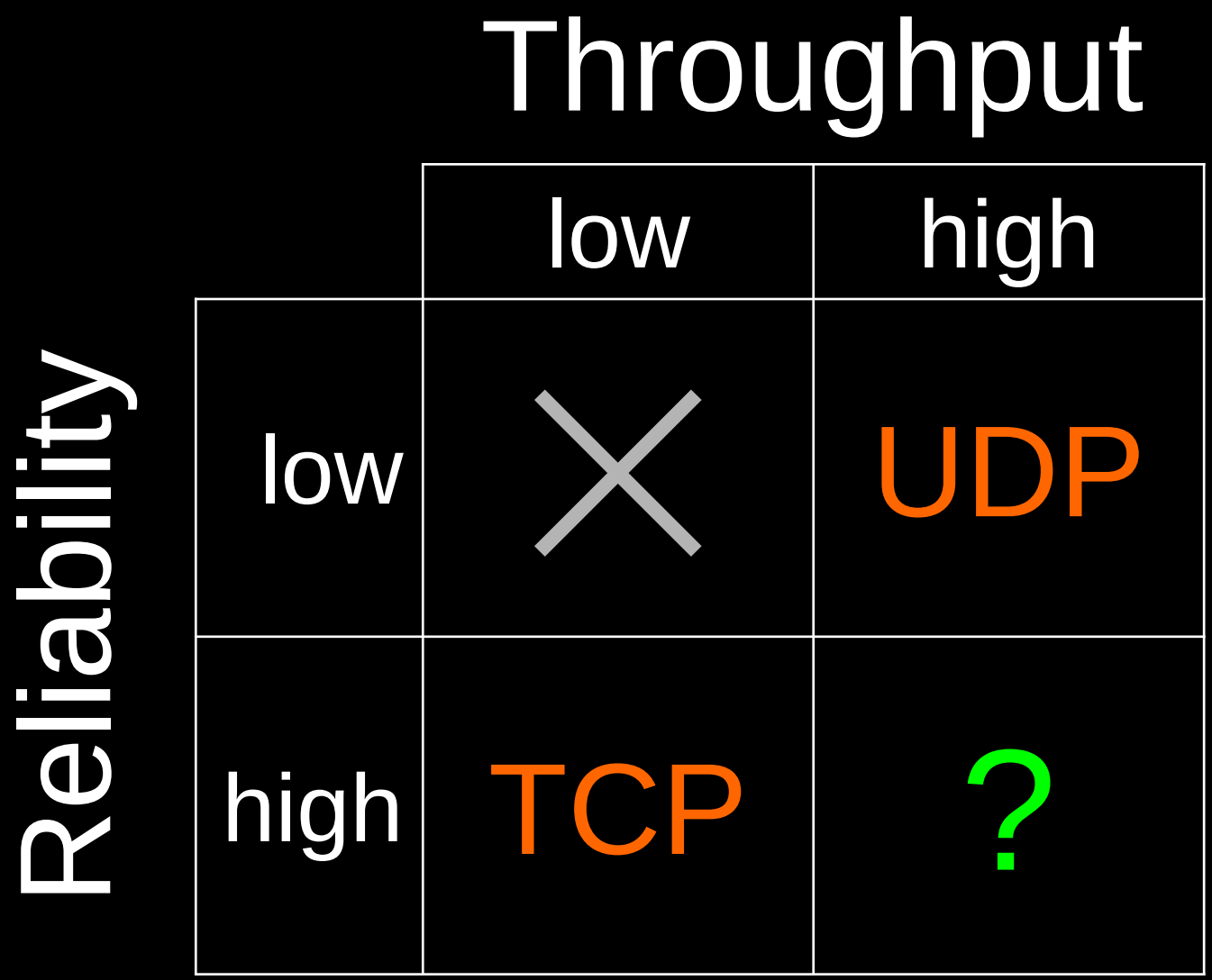


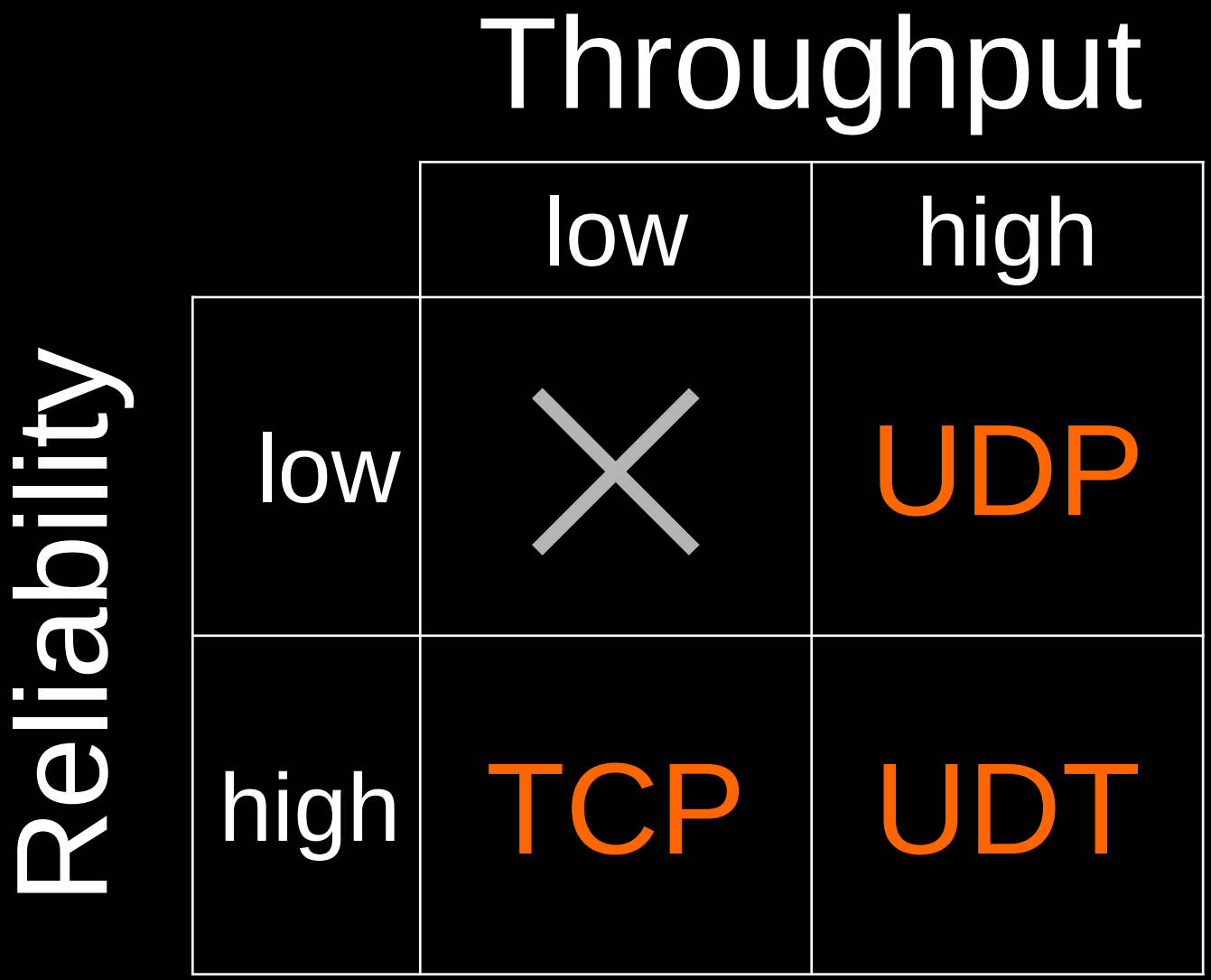
Time


Throughput
low
high
Reliability
low
high


Throughput
low
high
Reliability
low
high


Throughput
low
high
Reliability
low
UDP
high
TCP


Throughput
low
high
Reliability
low
high
UDP
TCP
?


Throughput
low
high
Reliability
low
UDP
high
TCP
UDT

UDT

UDP based Data Transport

- combines reliability + speed
- software library in user space
- is *just a protocol*
- needs an application to use it
  - e.g. `jive5ab`
- several years positive experience at JIVE

`http://udt.sourceforge.net/`

CLEOPATRA

Workpackage 5

Connecting
Locations of
ESFRI
Observatories and
Partners in
Astronomy for
Timing and
Real-time
Alerts

”Further development of existing data streaming software, building on the success of previous e-VLBI projects, and providing tools for robust and efficient data dissemination ...”

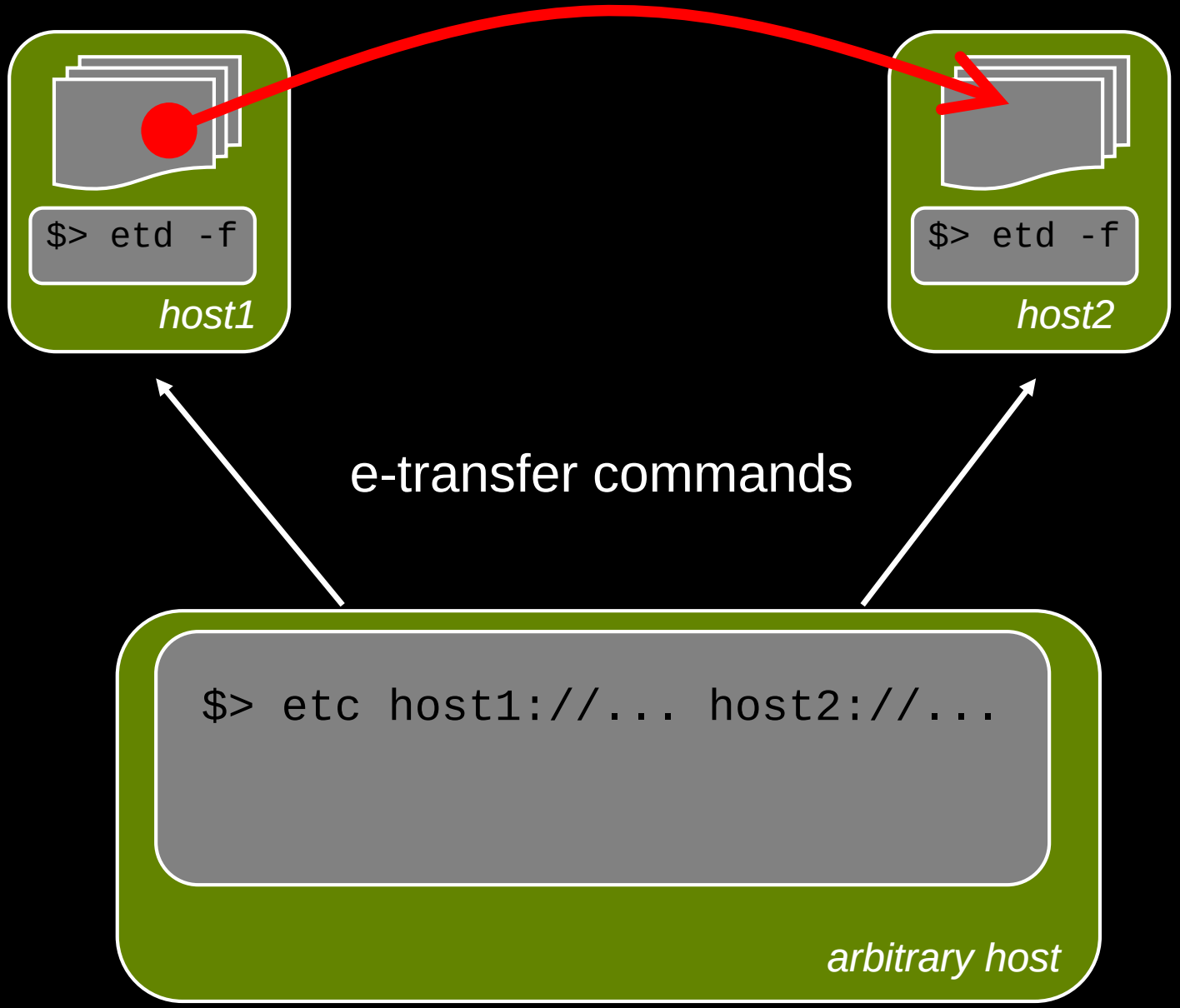
Funding to *do it right* this time!

Design

- simple to administer
- transfer big files
- support remote wildcards in file names
- transfer must be restartable, resumable, skippable
  - cf. `VLBI application` options
- UDT option for data channel
  - separate control- and data channels
- support remote-to-remote transfers
  - not readily found in other transfer tools
- proper server/client architecture
  - for both control AND data channels this time
  - server + client negotiate data channel

Design

- simple to administer
- transfer big files
- support remote wildcards in file names
- transfer must be restartable, resumable, skippable
  - cf. `VLBI application` options
- UDT option for data channel
  - separate control- and data channels
- support remote-to-remote transfers
  - not readily found in other transfer tools
- proper server/client architecture
  - for both control AND data channels this time
  - server + client negotiate data channel


$> etd -f
host1
$> etd -f
host2
e-transfer commands
$> etc host1://... host2://...
arbitrary host

Design

- transfer big files
- support remote wildcards in file names
- transfer must be restartable, resumable, skippable
  - cf. `m5copy` options
- UDT option for data channel
  - separate control- and data channels
- support remote-to-remote transfers
  - not found in any other transfer tool
- proper server/client architecture
  - for both control AND data channels this time
  - server + client negotiate data channel
- proper UNIX daemon
- ...

Prototype implementation

Design verification using `Python`

- explore client/server protocol
- validation of feasibility of ideas
- including UDT (`PyUDT` is ‘of course’ available ...)

Implementation in C++11

‘new’ C++ standard offers much for day-to-day use:

- multithreading support
- scoped locks, memory
  - auto release when leaving scope!
- lambdas + closures!
  - not the namby-pamby Python wannabees: the real thing!
- variadic templates
  - type safe `varargs`!, type safe forwarding of args
- introduces `std::move(...)` semantics
  - e.g.: move a locked mutex into another scope
  - elide copying of large temporary objects

Actual features of v0.1

- list remote files
- can transfer multiple files
- remote wildcards
- *can* do remote to remote
  - ^C not yet supported correctly here ☹
- one server can accept connections on:
  - multiple control channels
  - multiple data channels
- client will try data channels in order
  - server admin can set ‘priority’
- TCP: IPv4 and IPv6
- UDT: IPv4 and IPv6

e-transfer daemon (etd)

```
$> etd -m 3 --run-as some_user
     --command tcp://:4004
     --data udt://:8008
     --data tcp://:8008
```

Daemonize + substitute uid to some_user
Listen on all interfaces
One command channel
Two data channels, try UDT first, if not connectable, use TCP

e-transfer daemon (etd)

```
$> etd -m 3 -f
      --command tcp://:4004
      --command tcp6://:4004
      --data udt6://:8008
      --data udt://:8008
```

Do not daemonize

Listen on all interfaces

Two TCP command channels – over IPv4 and IPv6

Two UDT data channels – over IPv4 and IPv6

by listing `udt6` before `udt`: try that one first, else fall back to IPv4

e-transfer daemon (etd)

```
$> etd -m 3
      --command tcp://10.88.0.50:4004
      --data tcp://:8008
$> etd -m 3
      --command tcp://192.42.120.42:4004
      --data udt://:8008
```

Run two daemons
If connection made to local IF: use tcp data channel (faster)
If connection made to external IF: use UDT data channel

e-transfer daemon (etd)

```
$> etd -m 3
      --command tcp://10.88.0.50:4004
      --data tcp://:8008
$> etd -m 3
      --command tcp://192.42.120.42:4004
      --data udt://:8008
```

Run two daemons
If connection made to local IF: use tcp data channel (faster)
If connection made to external IF: use UDT data channel

e-transfer daemon (etd)

```
$> etd -m 3
      --command tcp://10.88.0.50:4004
      --data tcp://:8008
$> etd -m 3
      --command tcp://192.42.120.42:4004
      --data udt://:8008
```

Run two daemons

If connection made to local IF: use tcp data channel (faster)

If connection made to external IF: use UDT data channel

e-transfer client (`etc`)

```
$> etc --list tcp6://flexbuf4.jive.eu:/tmp/f*
```

List file names matching `/tmp/f*` on flexbuf4 @JIVE
Connect using TCP/IPv6

e-transfer client (etc)

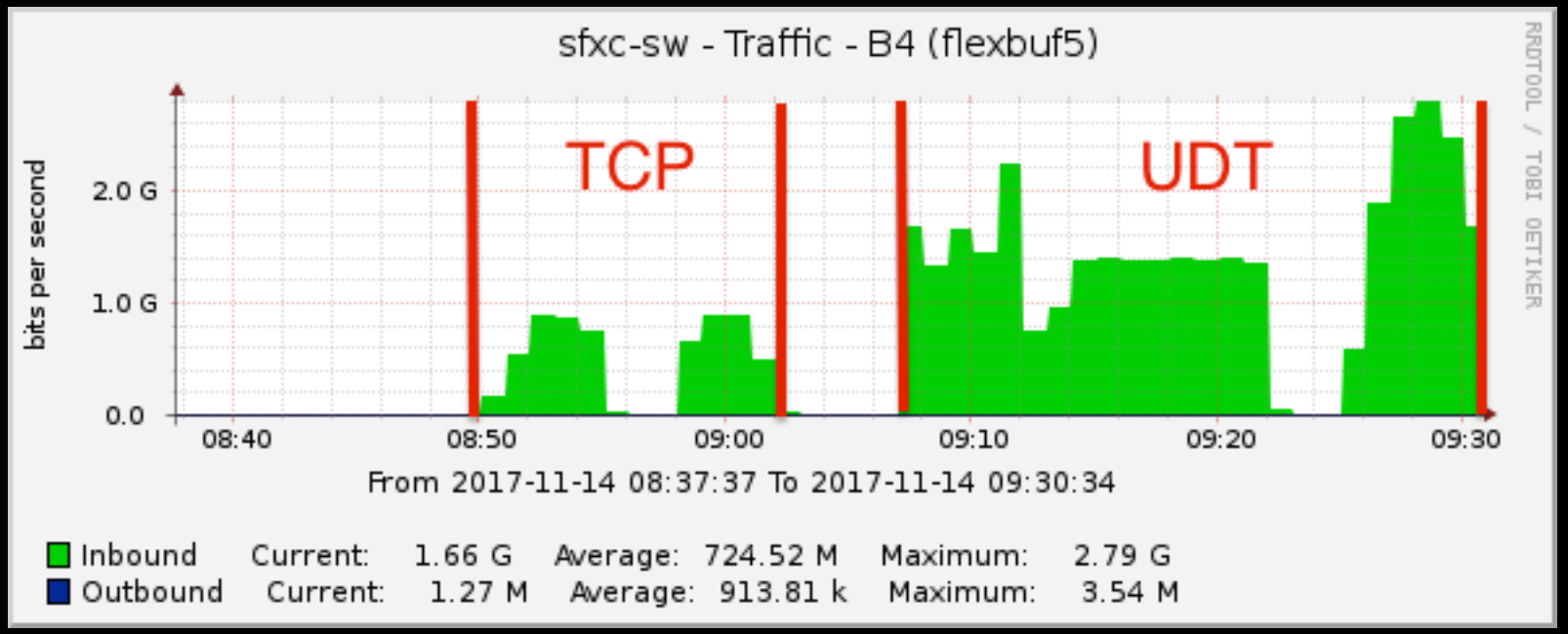
```
$> etc -m 3
     flexbuf4.jive.eu:/tmp/f*
     /mnt/data/fb4/
```

Copy all files matching `/tmp/f*` on flexbuf4 to local directory
Connect using TCP/IPv4 (the default)

e-transfer client (etc)

```
$> etc –m 3
     flexbuf4.jive.eu:/tmp/f*
     io13.mpifr-bonn.mpg.de:/data/f/
```

Copy all files matching `/tmp/f*` on flexbuf4 to remote directory
Connect using TCP/IPv4 (the default)



Transfer throughput from Netherlands to New Zealand

- 10 Gbps all the way
- 2 attempts
  - TCP #fails, never finishes
  - UDT just finishes, whilst adapting to ‘network weather’

Thanks!

Harro Verkouter

Joint Institute for VLBI in Europe

https://github.com/jive-vlbi/etransfer