ENTERPRISES, WILLIAM GIBSON & THE ART OF INTERNET MEASUREMENT

Vern Paxson, Co-Founder and Chief Scientist, Corelight Inc.

Professor of the Graduate School, UC Berkeley

vern@corelight.com

ACM IMC 2023 | October 24, 2023

ENTERPRISES, WILLIAM GIBSON & THE ART OF INTERNET MEASUREMENT

1) **Institutional Networks**
2) **Undertakings by individuals/groups**

Scientist, Corelight Inc.

Berkeley

ENTERPRISES, WILLIAM GIBSON & THE ART OF INTERNET MEASUREM

> The future is already here – it's just not evenly distributed.
>
> —William Gibson

How widespread is IPv6?

Do we have pervasive encryption everywhere à la Zero Trust?

Has QUIC taken over the world?

Is TLS opaque thanks to v1.3?

Does anyone still use clear-text HTTP?

… and what has happened that we *didn't* anticipate?

ENTERPRISES, WILLIAM GIBSON & THE ART OF INTERNET MEASUREMENT

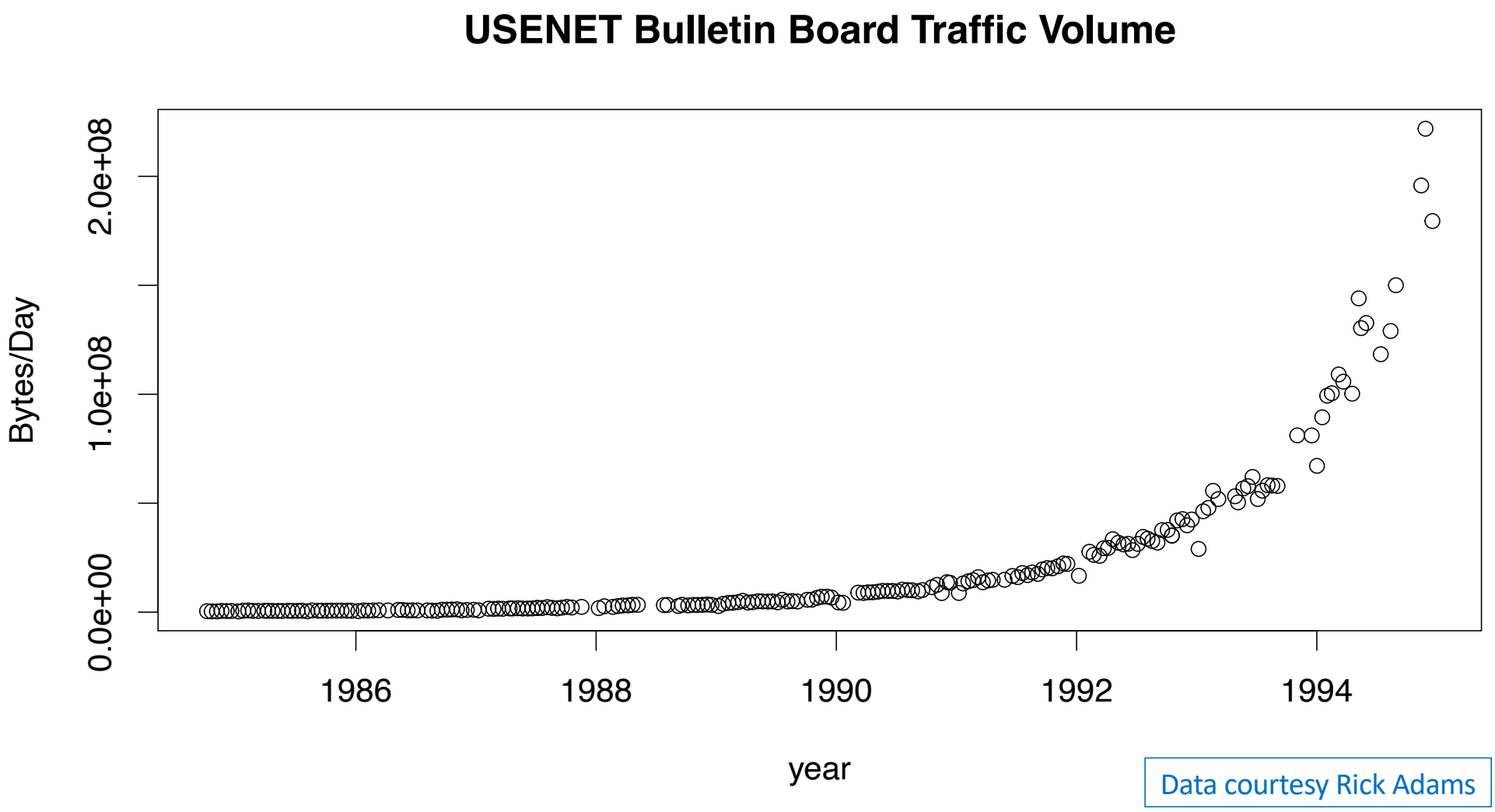
Not:
How to perform effective Internet measurements

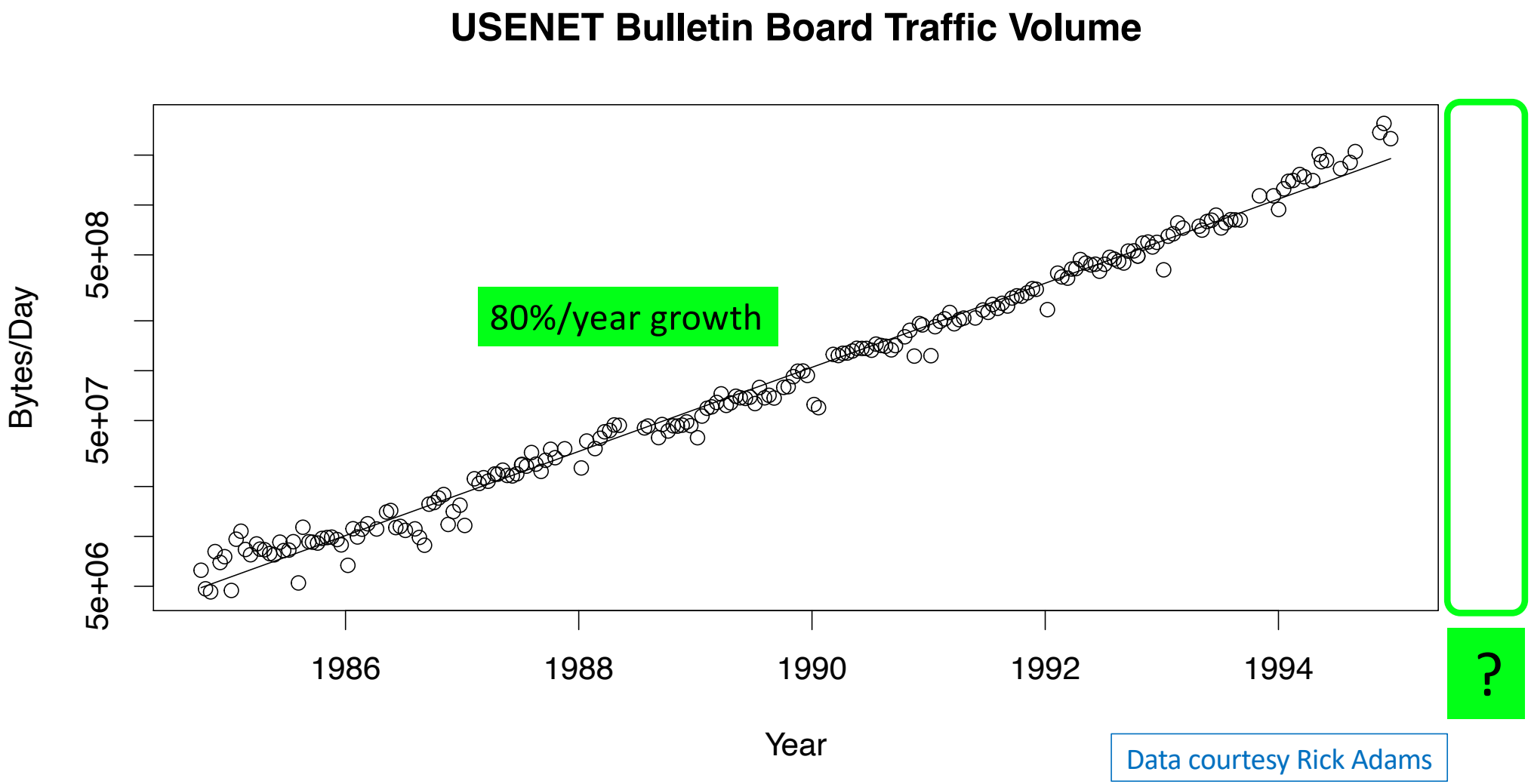
Scientist, Corelight Inc.
Berkeley

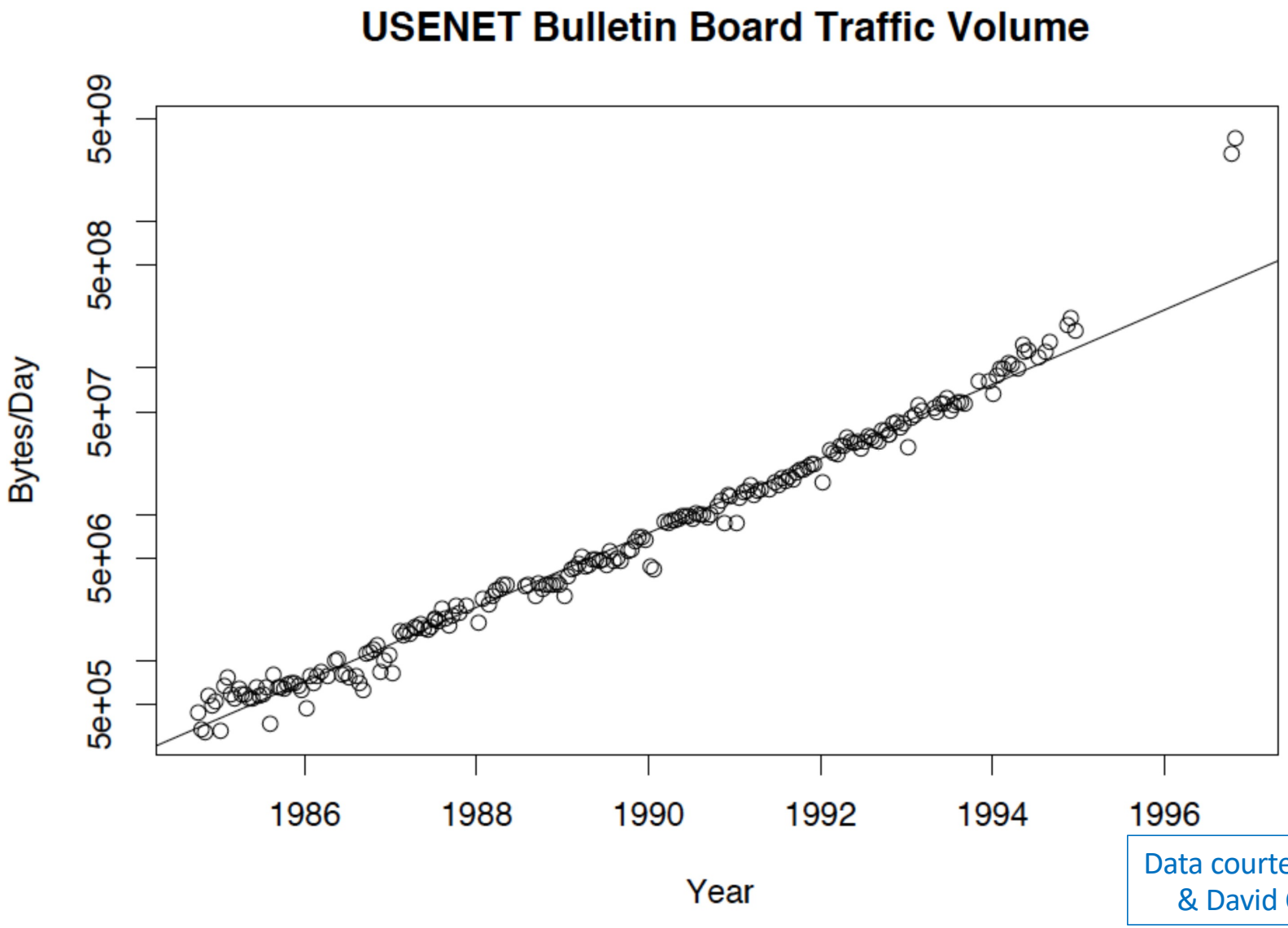
ENTERPRISES, WILLIAM GIBSON & THE ART OF INTERNET MEASUREMENT

But instead:

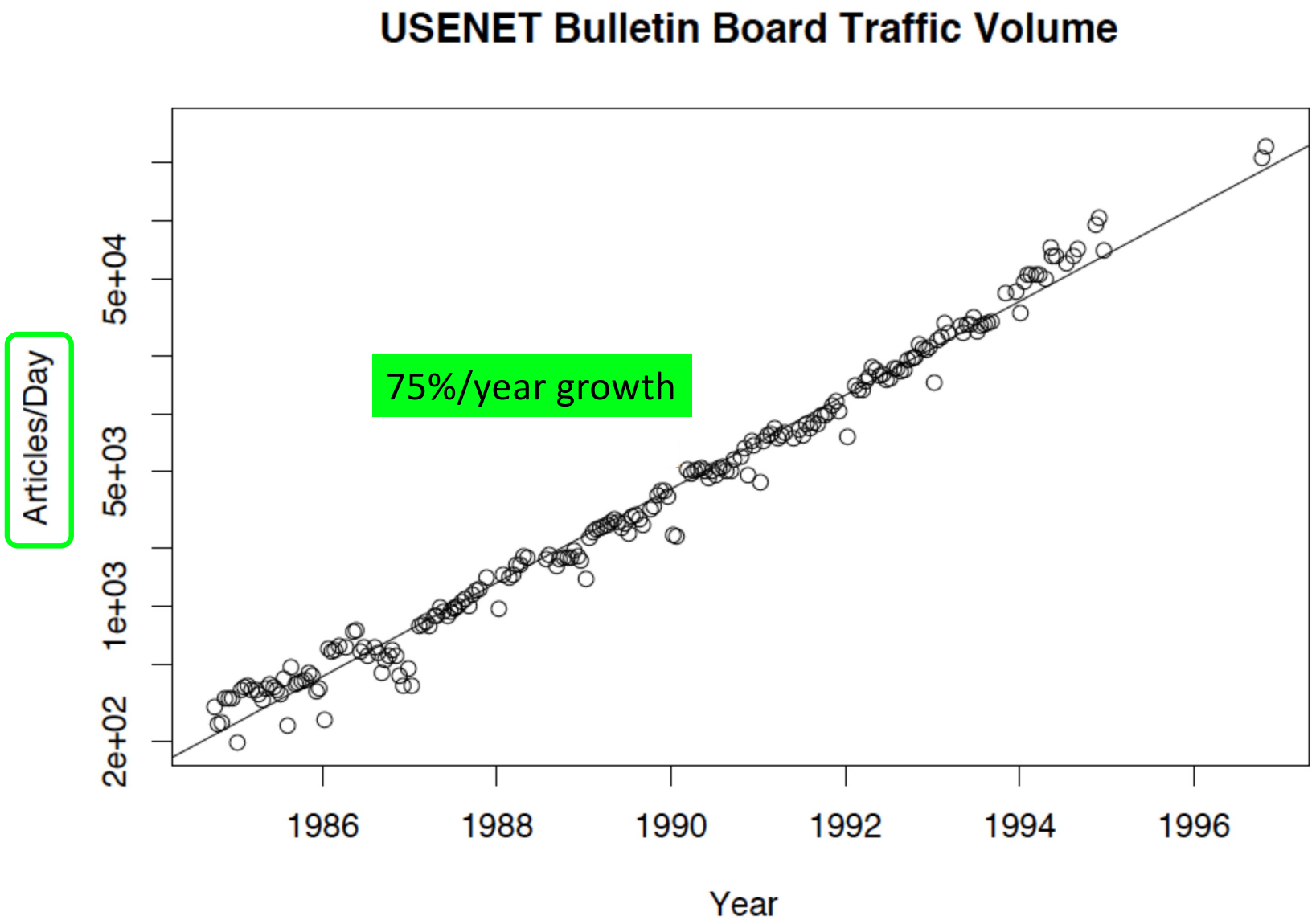
How (some) Internet measurements can inspire excitement, puzzlement, wonder

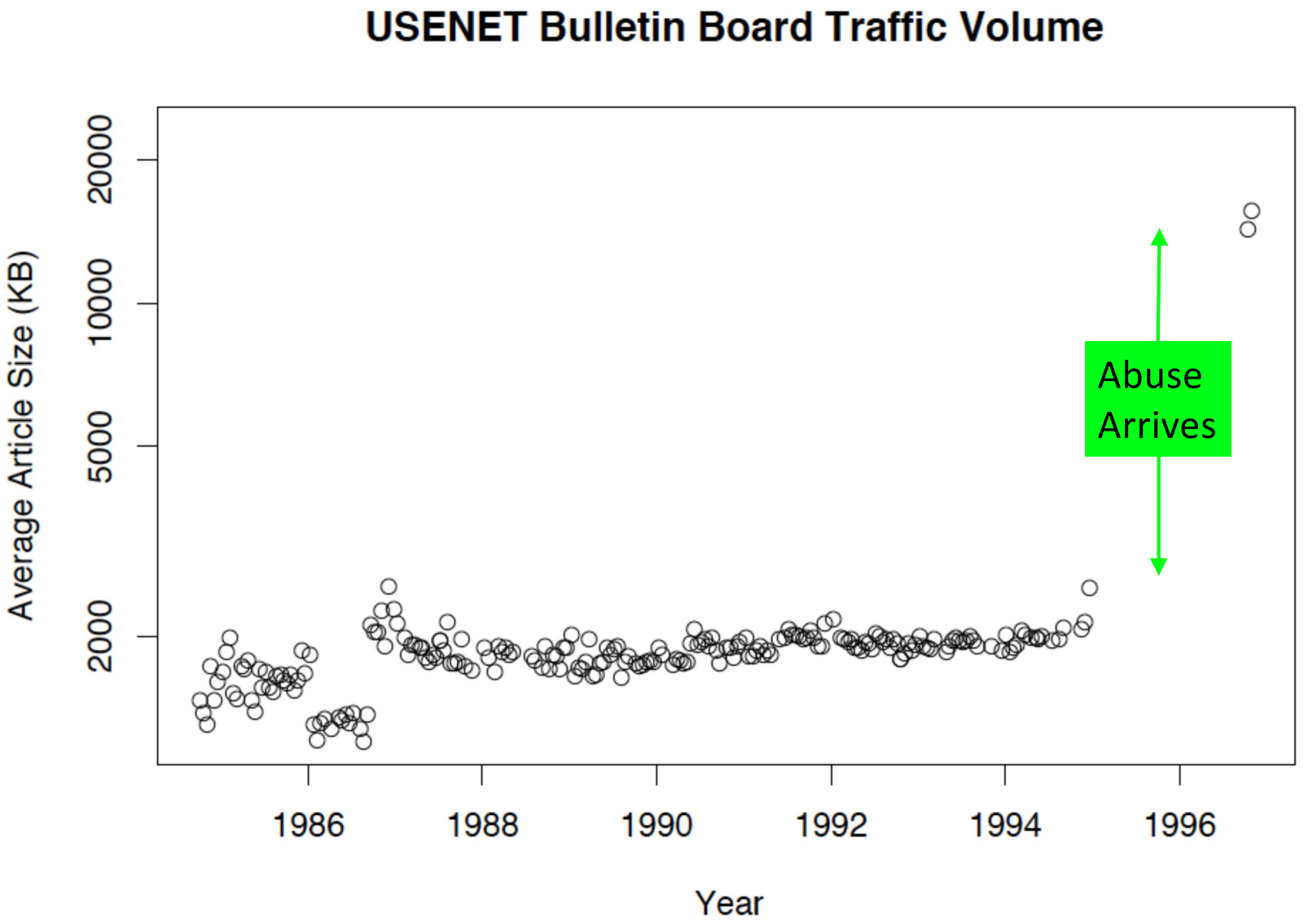

USENET Bulletin Board Traffic Volume
Bytes/Day
0.0e+00
1.0e+08
2.0e+08
1986
1988
1990
1992
1994
year
Data courtesy Rick Adams


USENET Bulletin Board Traffic Volume
Bytes/Day
5e+06
5e+07
5e+08
1986
1988
1990
1992
1994
Year
80%/year growth
?
Data courtesy Rick Adams

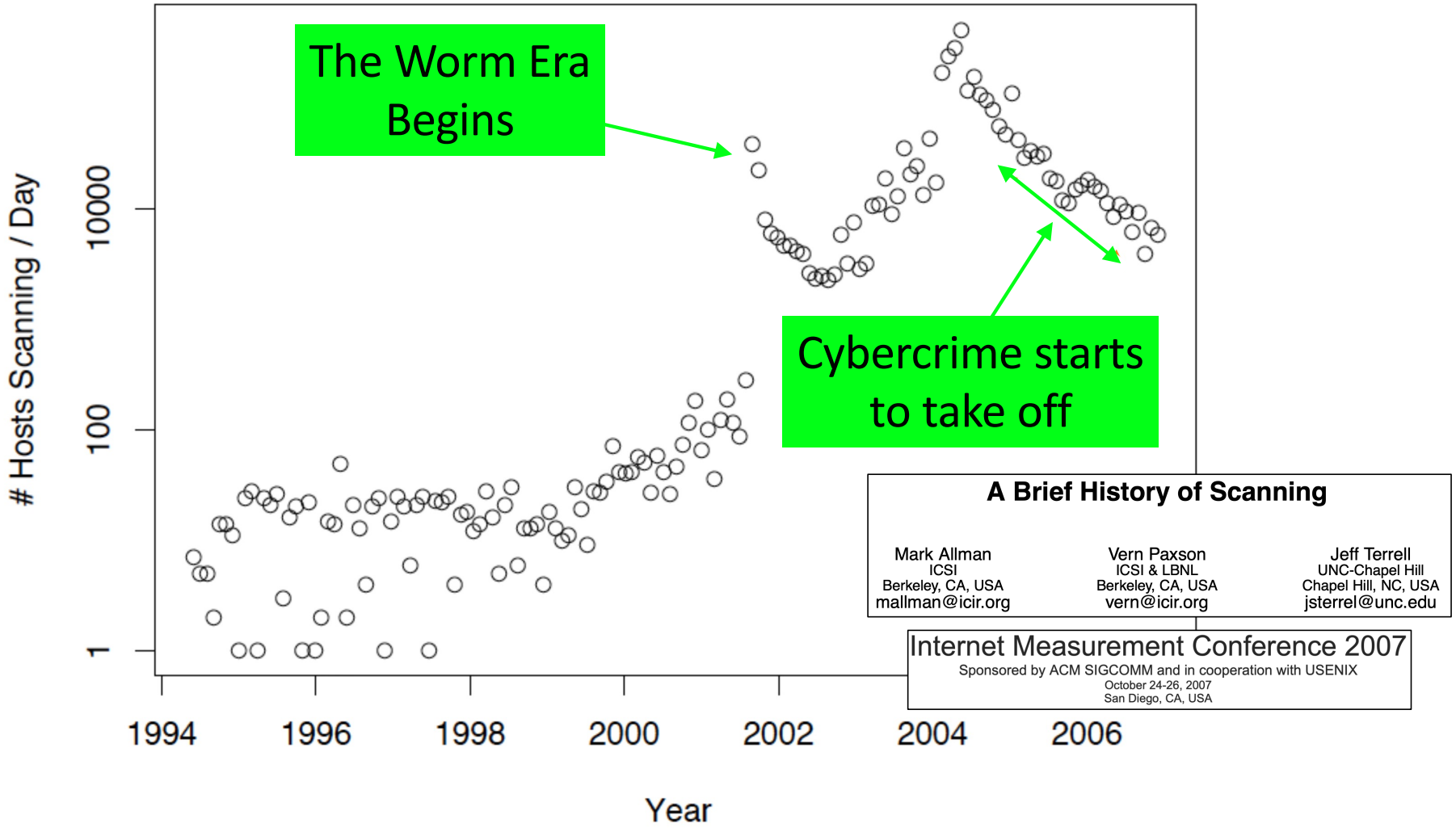


Data courtesy Rick Adams & David C. Lawrence


USENET Bulletin Board Traffic Volume
Articles/Day
2e+02
1e+03
5e+03
5e+04
75%/year growth
1986
1988
1990
1992
1994
1996
Year


USENET Bulletin Board Traffic Volume
Average Article Size (KB)
2000
5000
10000
20000
1986
1988
1990
1992
1994
1996
Year
Abuse
Arrives

Scan Activity Seen At LBL



A Brief History of Scanning

Mark Allman
ICSI
Berkeley, CA, USA
mallman@icir.org

Vern Paxson
ICSI & LBNL
Berkeley, CA, USA
vern@icir.org

Jeff Terrell
UNC-Chapel Hill
Chapel Hill, NC, USA
jsterrel@unc.edu

Internet Measurement Conference 2007
Sponsored by ACM SIGCOMM and in cooperation with USENIX
October 24-26, 2007
San Diego, CA, USA

A Brief History of Scanning

Mark Allman
ICSI
Berkeley, CA, USA
mallman@icir.org

Vern Paxson
ICSI & LBNL
Berkeley, CA, USA
vern@icir.org

Jeff Terrell
UNC-Chapel Hill
Chapel Hill, NC, USA
jsterrel@unc.edu

Internet Measurement Conference 2007
Sponsored by ACM SIGCOMM and in cooperation with USENIX
October 24-26, 2007
San Diego, CA, USA

ABSTRACT

Incessant scanning of hosts by attackers looking for vulnerable servers has become a fact of Internet life. In this paper we present an initial study of the scanning activity observed at one site over the past 12.5 years. We study the onset of scanning in the late 1990s and its evolution in terms of characteristics such as the number of scanners, targets and probing patterns. While our study is preliminary in many ways, it provides the first longitudinal examination of a now ubiquitous Internet phenomenon.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General; C.2.3 [**Computer-Communication Networks**]: Network Operations; C.2.6 [**Computer-Communication Networks**]: Internetworking

General Terms

Measurement,Security

Keywords

Scanning,Longitudinal,Malicious Activity

1. INTRODUCTION

Many forms of Internet activity have proven highly challenging to characterize due to the network's great diversity. Because sound characterization often requires a very *broad* perspective in order to capture the full range of variation manifested, such measurement studies face deep methodological challenges for how to *(i)* acquire sufficient breadth of [illegible] and *(ii)* [illegible]

selected IP addresses [8, 9, 7]. Bailey and colleagues present a system based on a wide range of distributed "black hole" network blocks coupled with probe responders [2], for which they show that the distributed perspective can be vital for capturing the range of variations that scanning activity manifests [3].

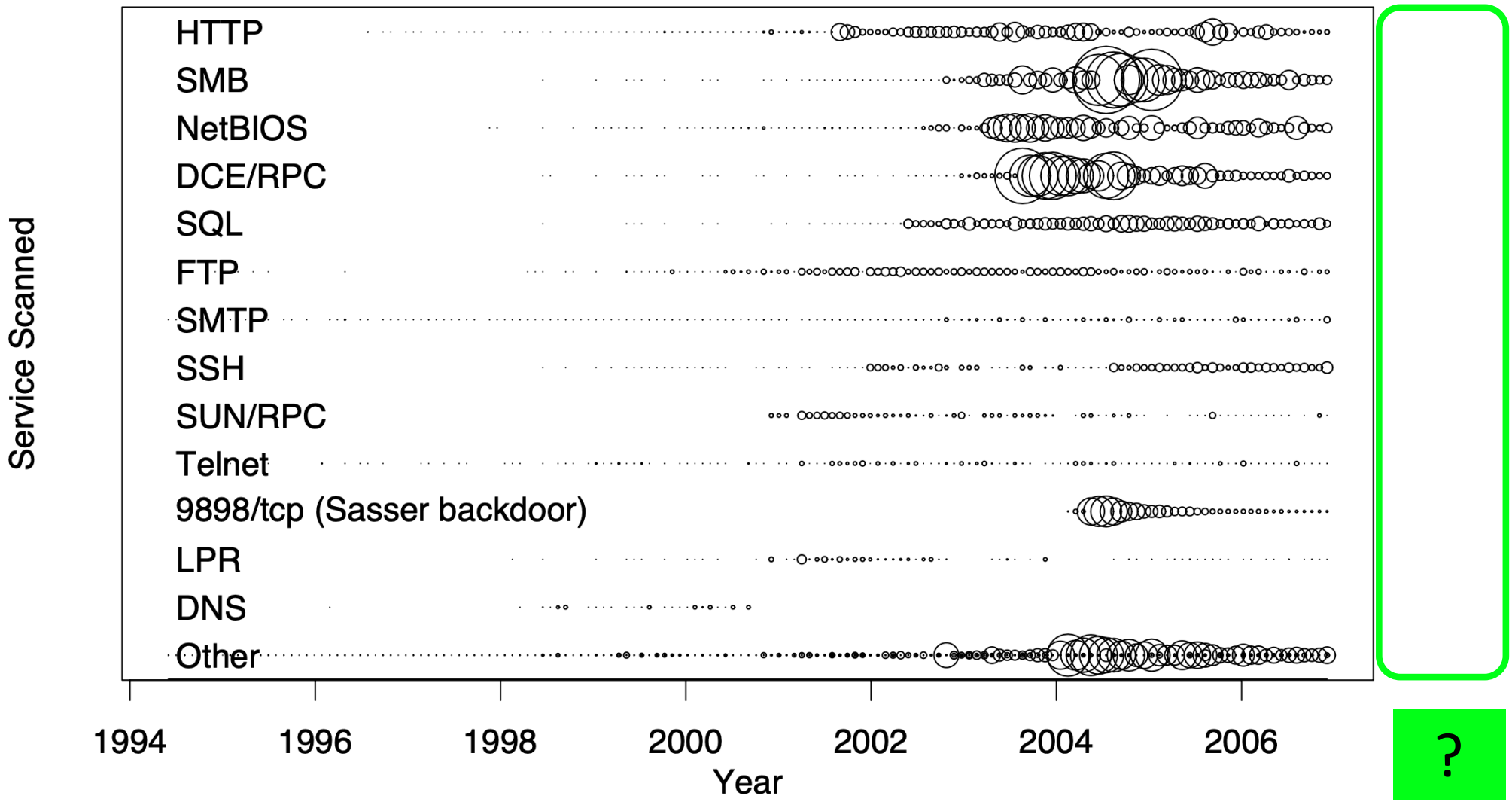
Following a different approach, Yegneswaran and colleagues studied a collection of network scanning logs from the global "DShield" repository [12, 1]. The data spanned a month-long period from 2001 (including the Code Red 2 outbreak) and a 3-month period from 2002, totaling 207 million scans sent to 1.4 million destination addresses. The study examines scanning prevalence, rates, types, and address clustering, offering a unique global perspective.

In this paper we also examine the phenomenon of scanning, but from a perspective global in *time* rather than *space*. That is, the data upon which we base our study begins in 1994—during which the measured site experienced virtually no scanning activity—through 2006, long past the point at which scanning became a ubiquitous phenomenon [10]. While only from a single site, the breadth of the data is extensive: the subset (1/30th) we use for this paper spans 628 million scans sent by 2.4 million distinct IP addresses.

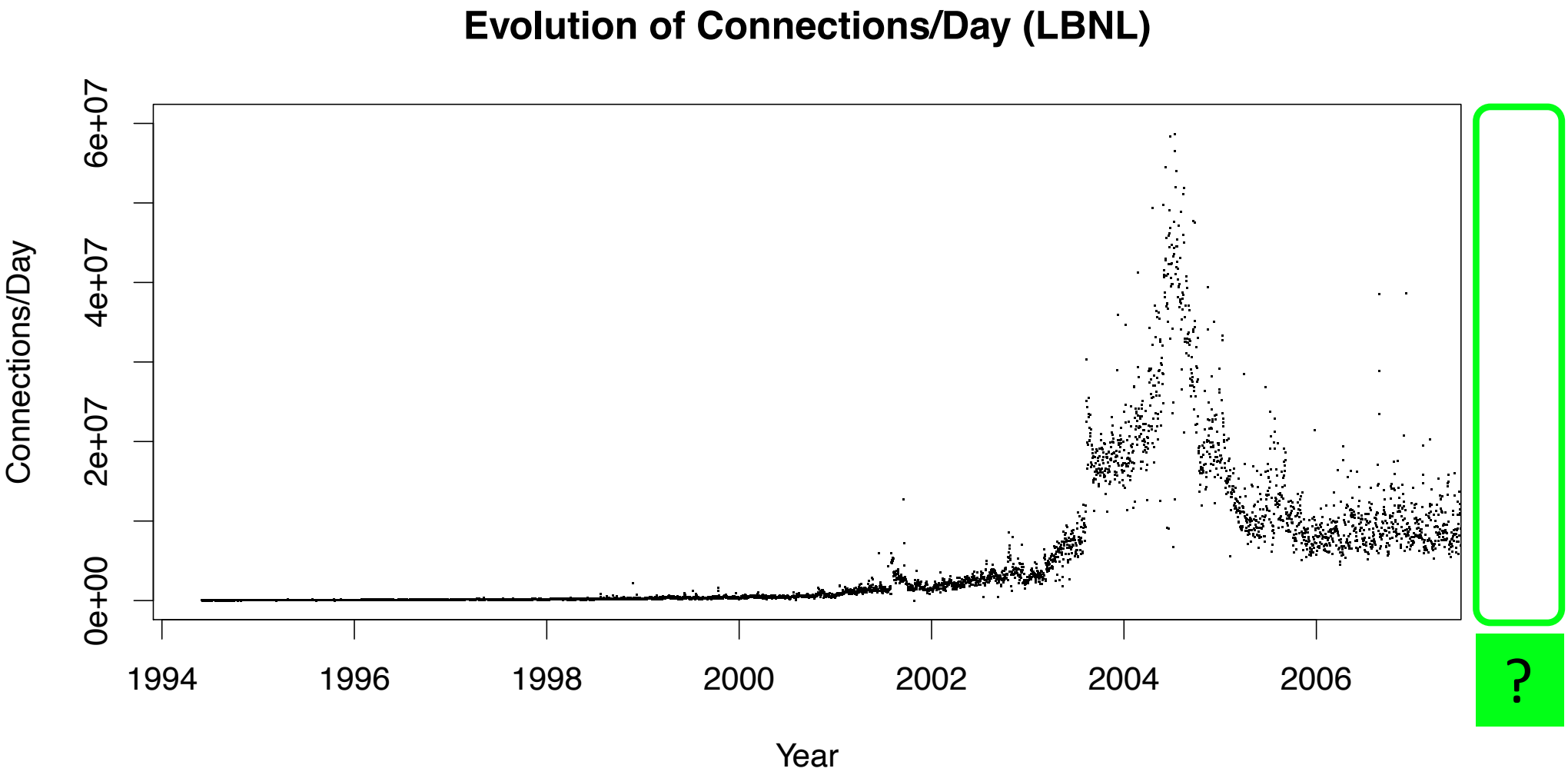
We emphasize that this paper reflects *preliminary* work, with many important questions (e.g., alignment of our findings with those framed in [12]) deferred for a more extensive study we are pursuing. However, we find that even our initial "scratching the surface" of the data reveals a number of interesting results.

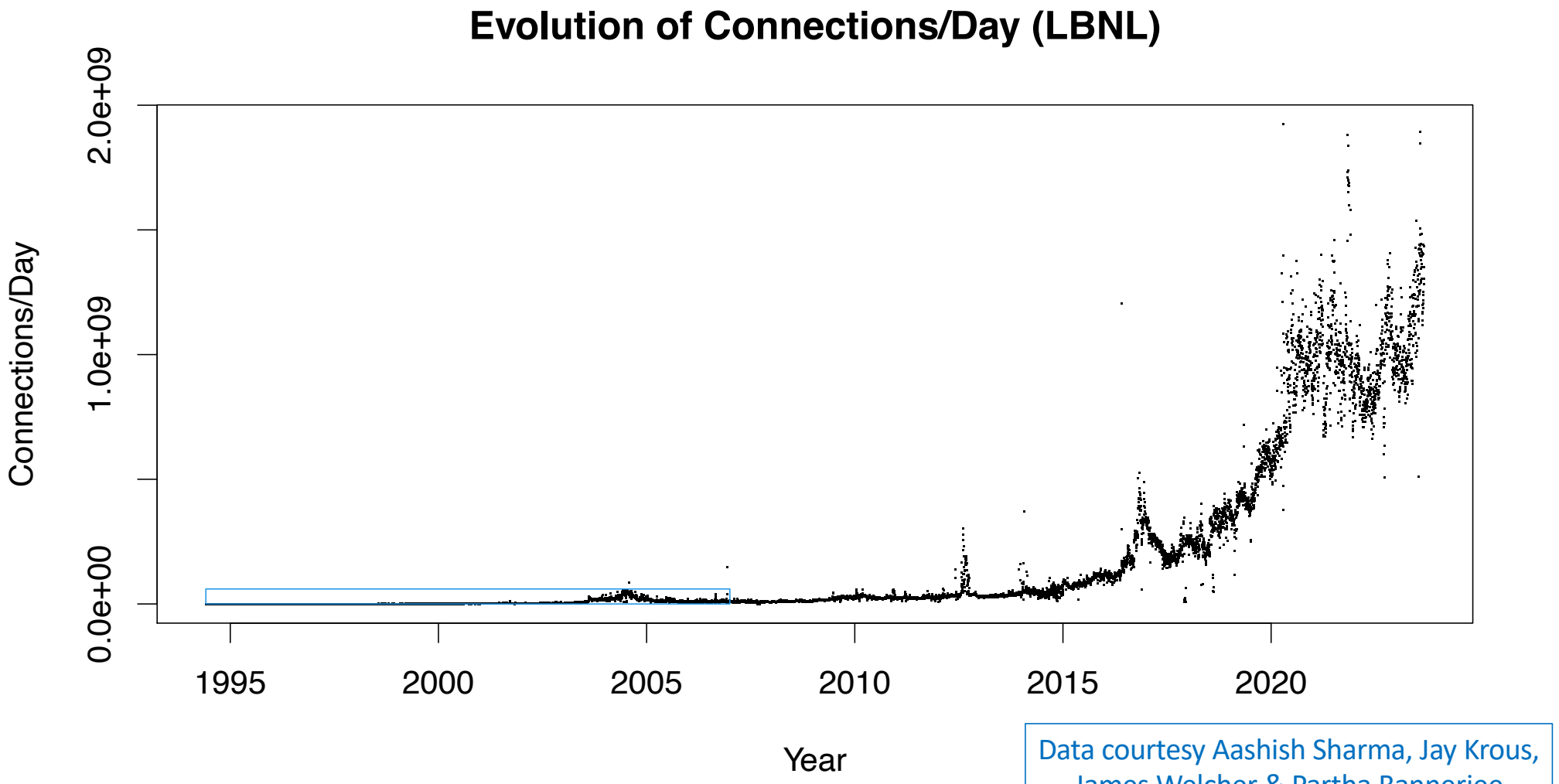
2. DATA

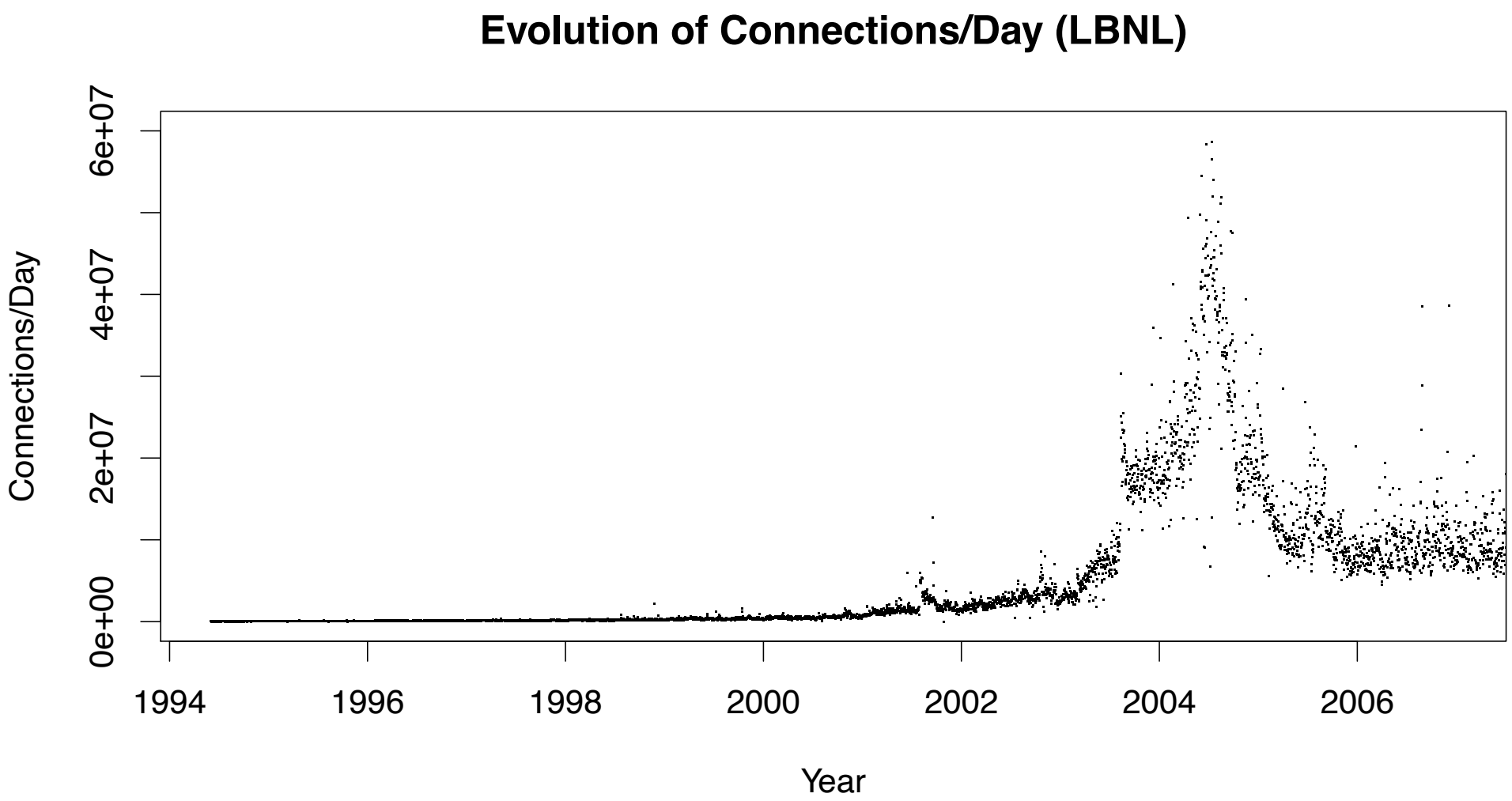
The data presented in this paper comes from 12.5 years of logs of network traffic continuously collected at the border of the Lawrence Berkeley National Laboratory (LBNL) in Berkeley, CA, USA.

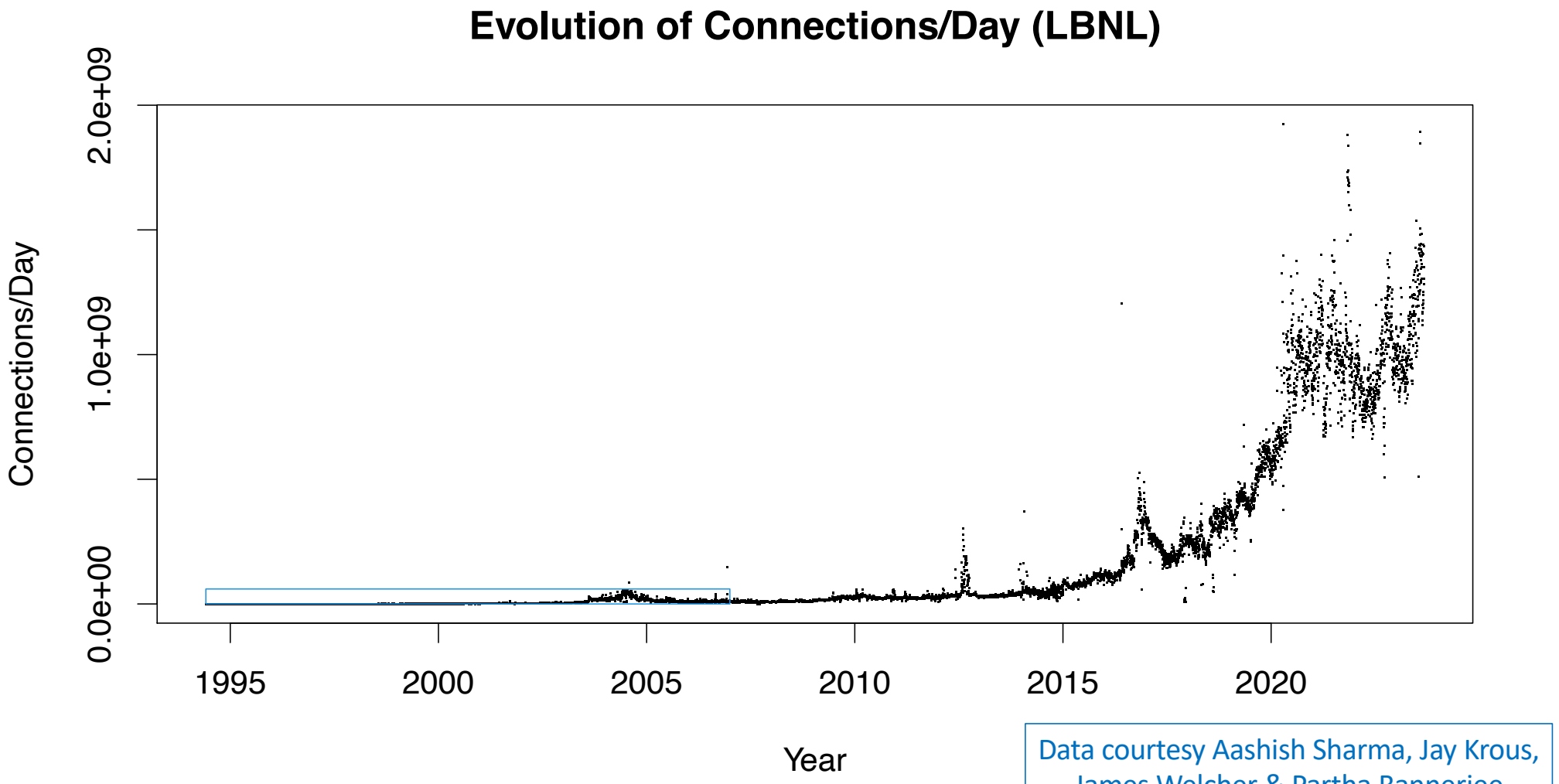

Service Scanned
HTTP
SMB
NetBIOS
DCE/RPC
SQL
FTP
SMTP
SSH
SUN/RPC
Telnet
9898/tcp (Sasser backdoor)
LPR
DNS
Other
1994
1996
1998
2000
2002
2004
2006
Year
?

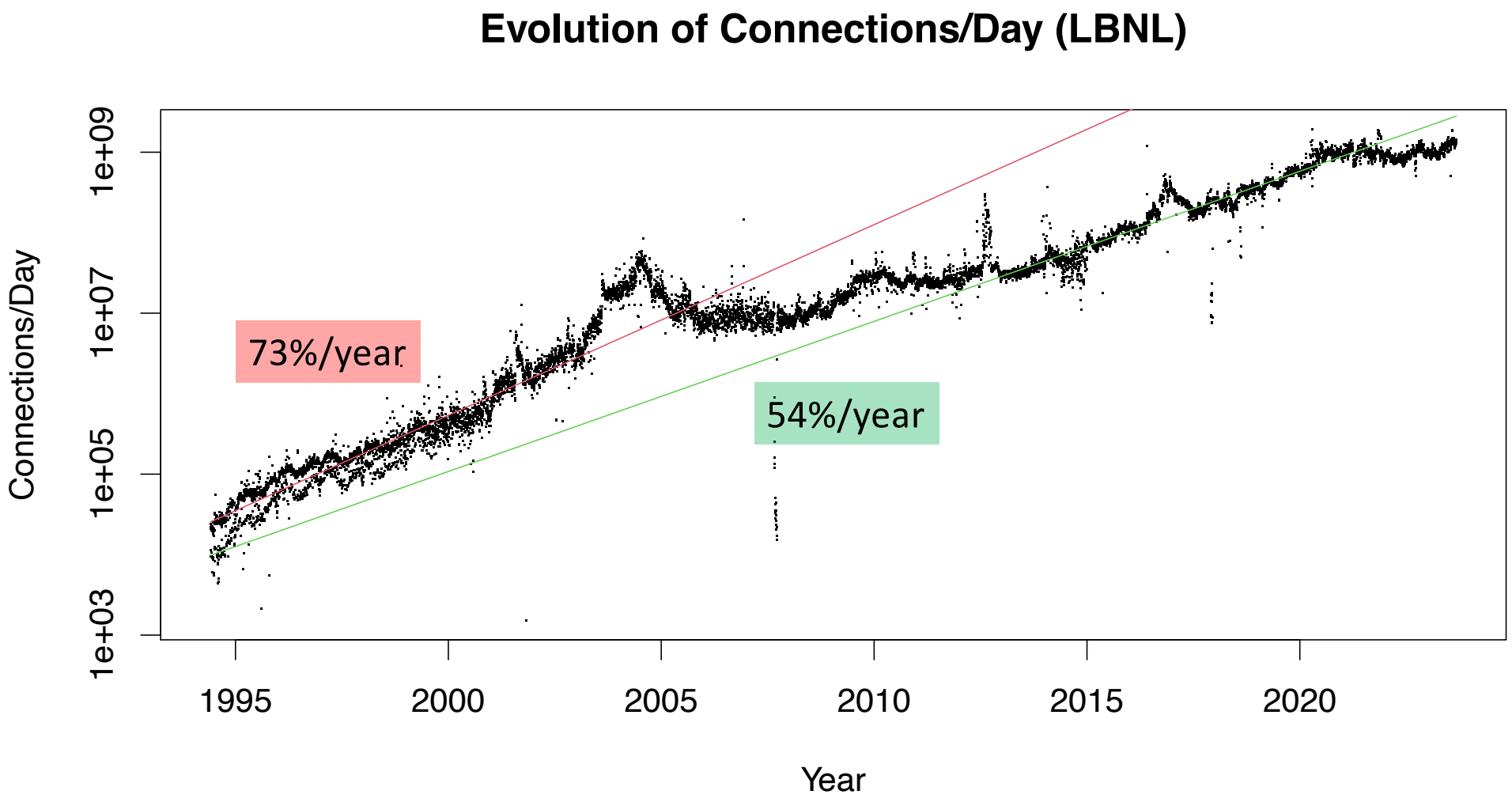
What Has Happened in the Last 16 Years?

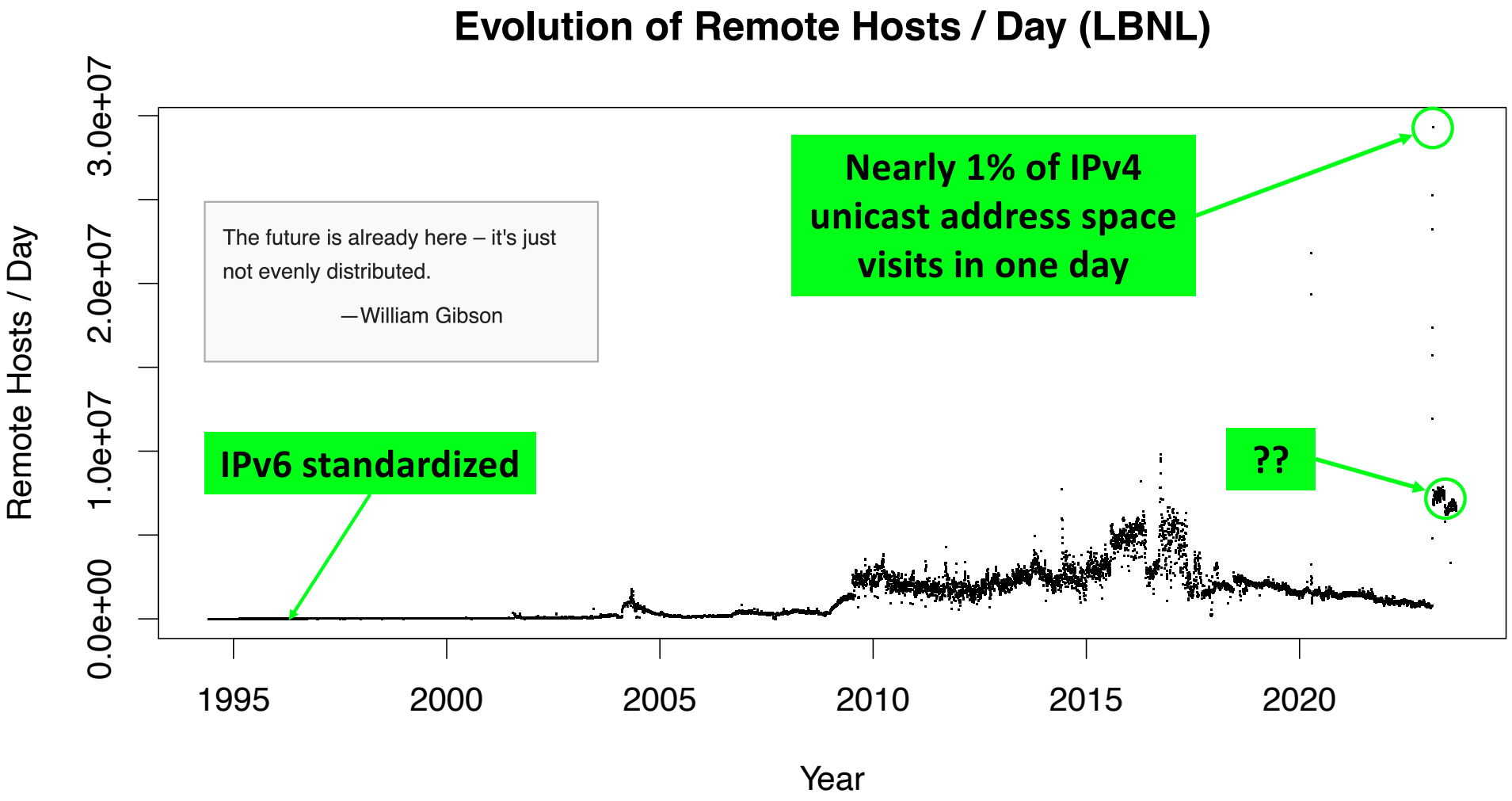

Evolution of Connections/Day (LBNL)
Connections/Day
0e+00
2e+07
4e+07
6e+07
1994
1996
1998
2000
2002
2004
2006
Year
?

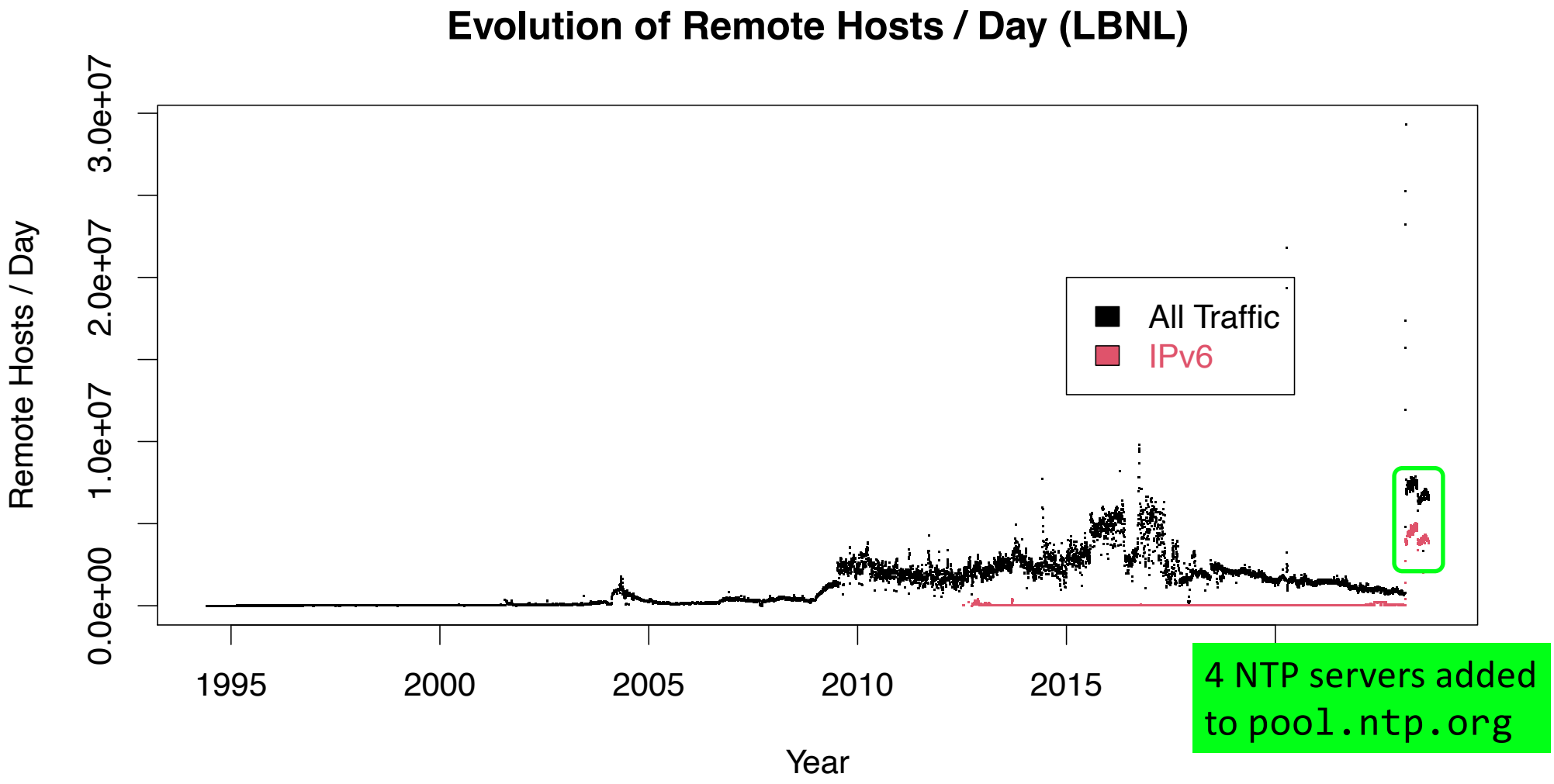

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
Data courtesy Aashish Sharma, Jay Krous,
James Welcher & Partha Bannerjee

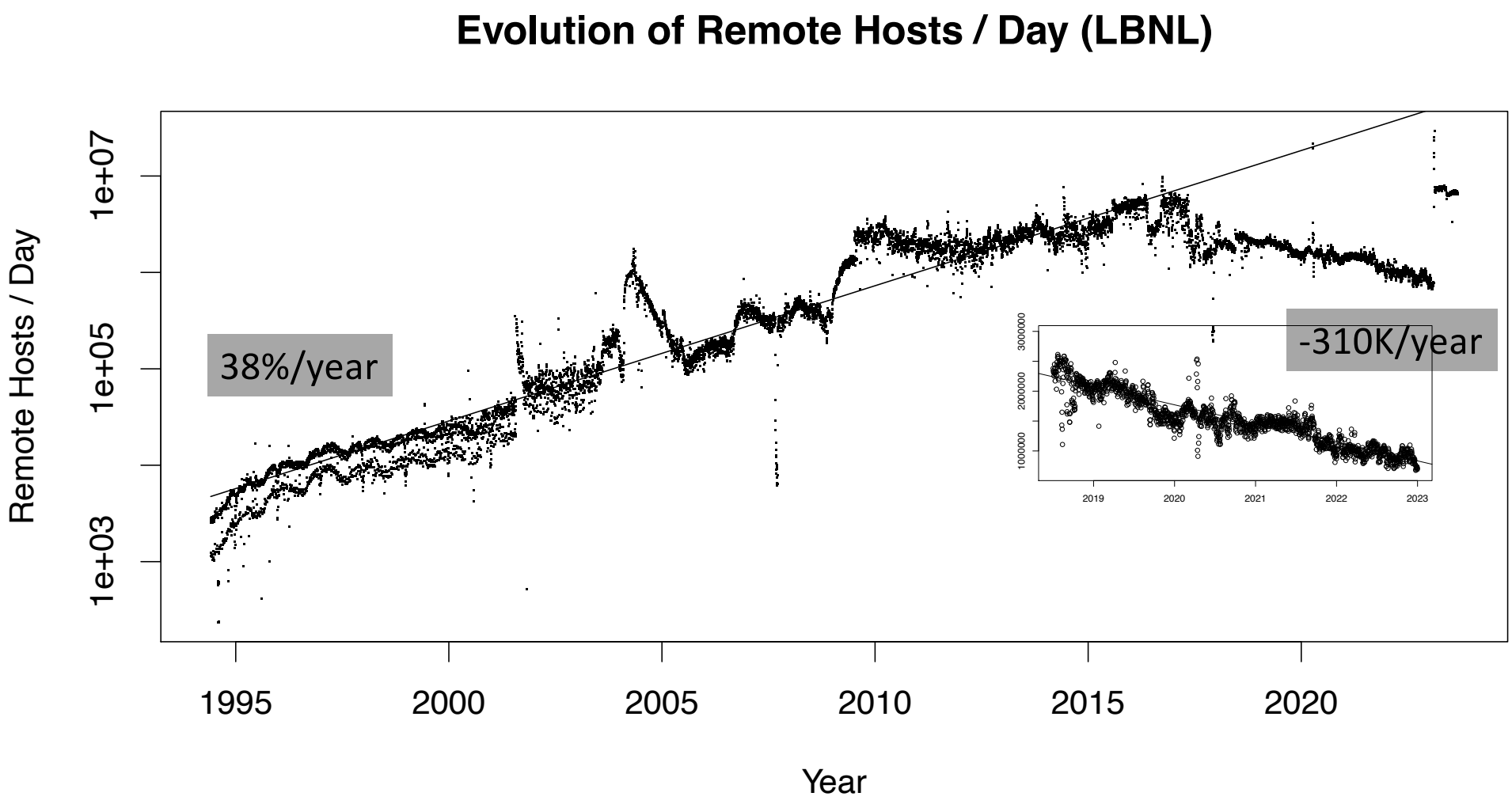

Evolution of Connections/Day (LBNL)
Connections/Day
0e+00
2e+07
4e+07
6e+07
1994
1996
1998
2000
2002
2004
2006
Year

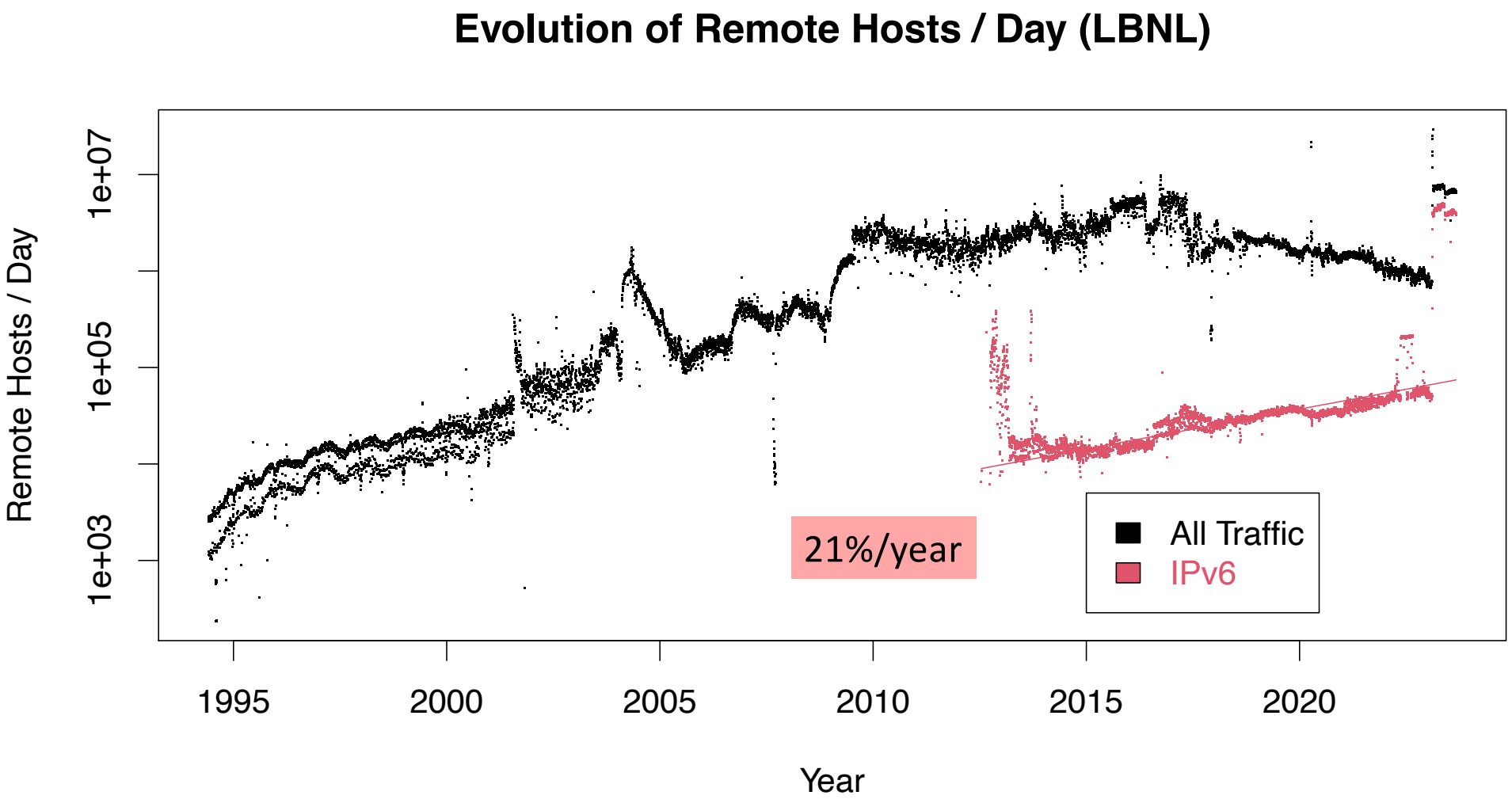

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
Data courtesy Aashish Sharma, Jay Krous,
James Welcher & Partha Bannerjee


Evolution of Connections/Day (LBNL)
Connections/Day
1e+03
1e+05
1e+07
1e+09
1995
2000
2005
2010
2015
2020
Year
73%/year
54%/year


Evolution of Remote Hosts / Day (LBNL)
Remote Hosts / Day
0.0e+00
1.0e+07
2.0e+07
3.0e+07
1995
2000
2005
2010
2015
2020
Year
The future is already here – it's just not evenly distributed.
—William Gibson
Nearly 1% of IPv4 unicast address space visits in one day
IPv6 standardized
??


Evolution of Remote Hosts / Day (LBNL)
Remote Hosts / Day
0.0e+00
1.0e+07
2.0e+07
3.0e+07
1995
2000
2005
2010
2015
Year
All Traffic
IPv6
4 NTP servers added
to pool.ntp.org


Evolution of Remote Hosts / Day (LBNL)
Remote Hosts / Day
1e+03
1e+05
1e+07
38%/year
-310K/year
3000000
2000000
1000000
2019
2020
2021
2022
2023
1995
2000
2005
2010
2015
2020
Year


Evolution of Remote Hosts / Day (LBNL)
Remote Hosts / Day
1e+03
1e+05
1e+07
1995
2000
2005
2010
2015
2020
Year
21%/year
All Traffic
IPv6

Site	IPv6
LBNL	?

Proportion of *connections* as of late 2023

Site	IPv6
LBNL	62%

!

Proportion of *connections* as of late 2023

Site	IPv6
LBNL	62%/2%

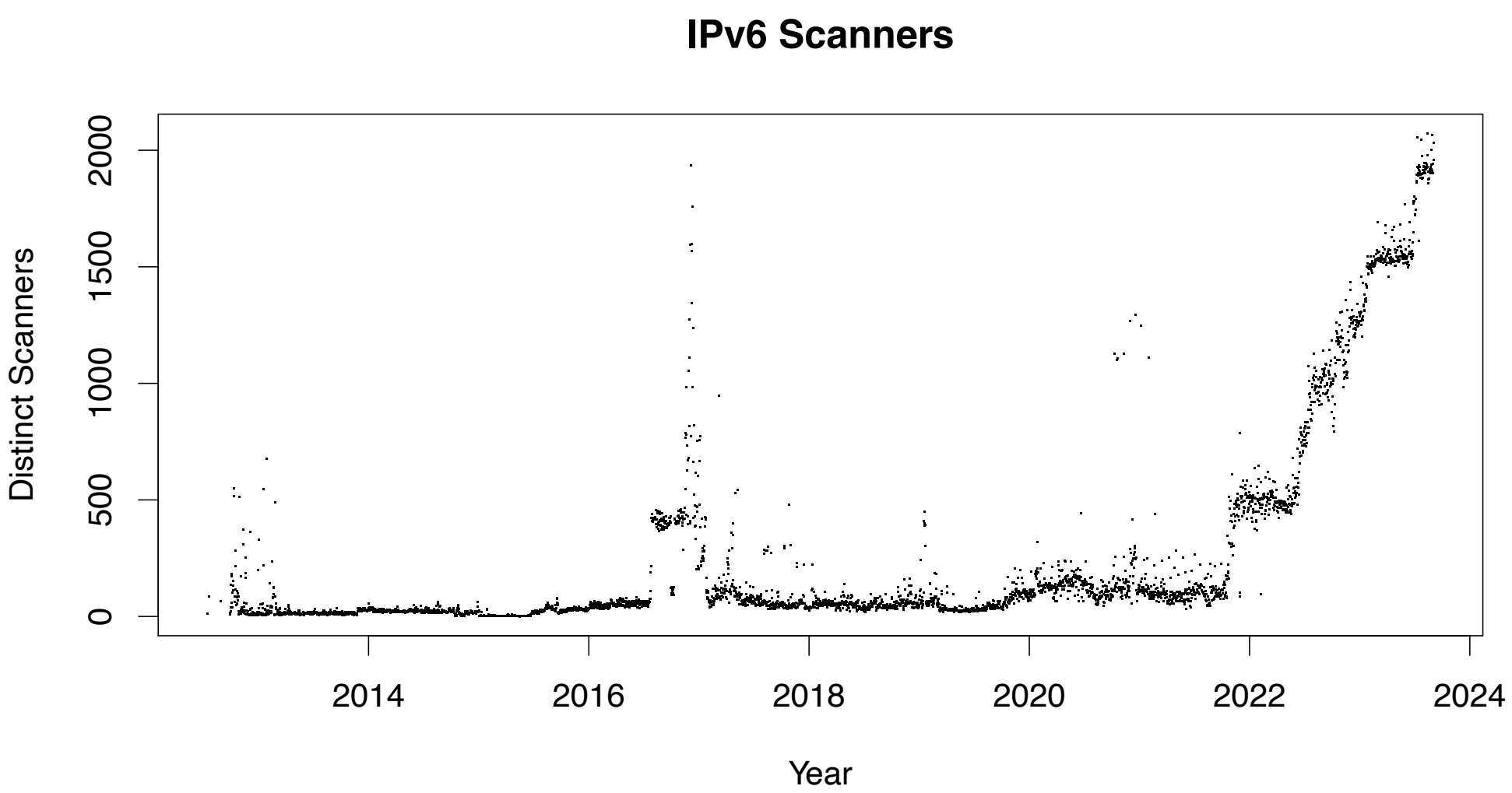
w/o NTP & DNS

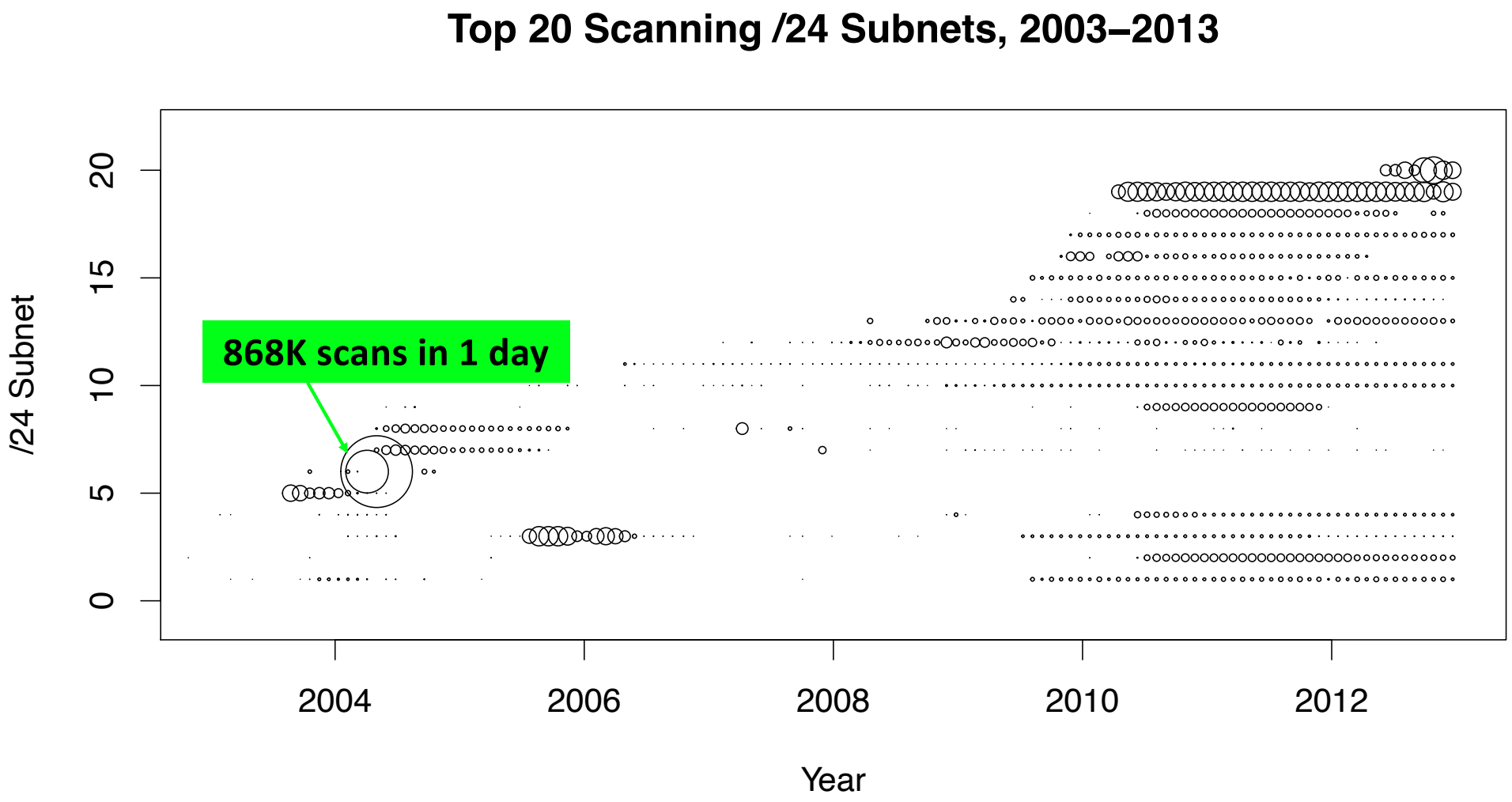
Proportion of *connections* as of late 2023

Site	IPv6
LBNL	62%/2%/0.14%

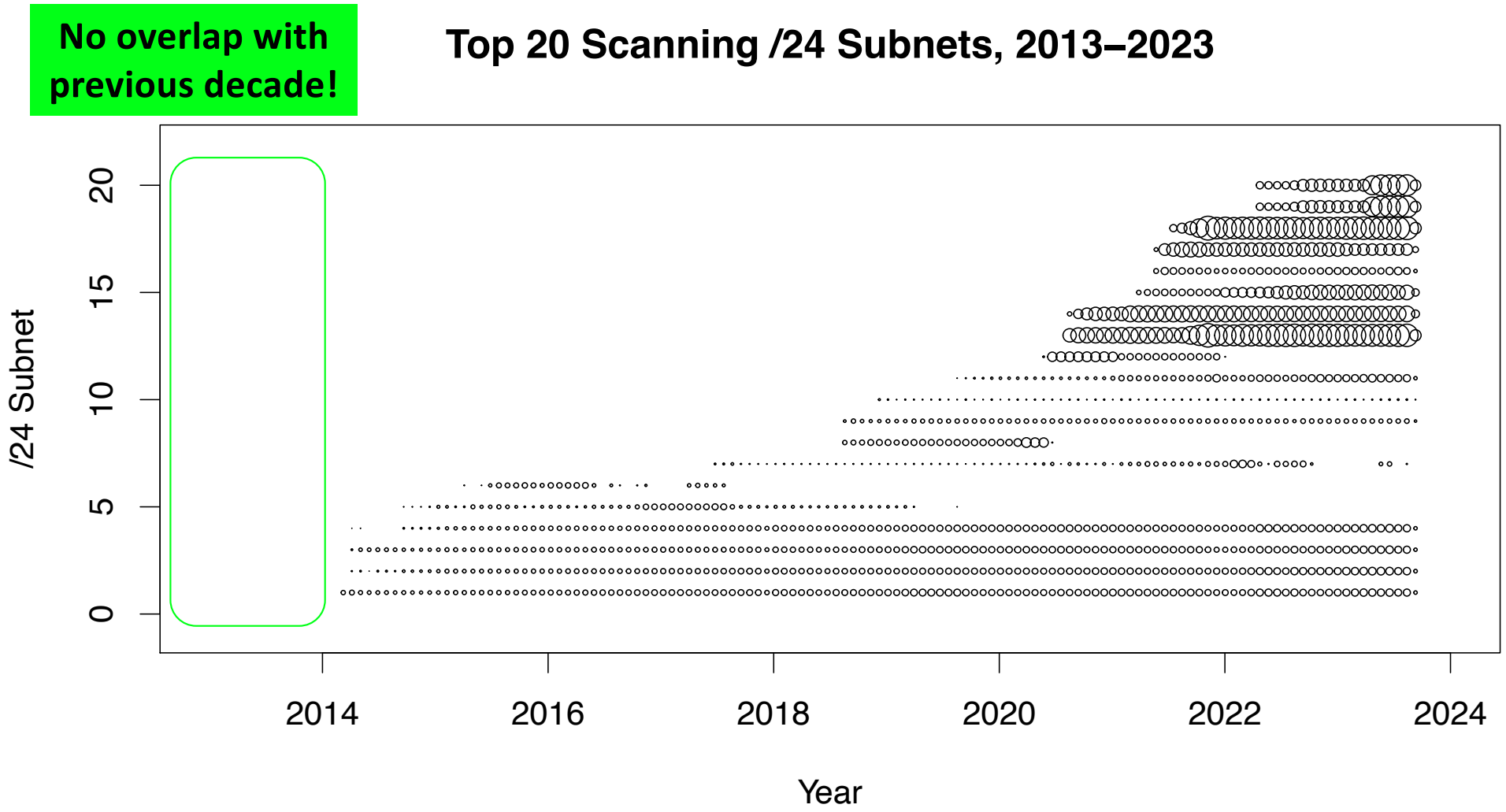
w/o NTP,
DNS & QUIC

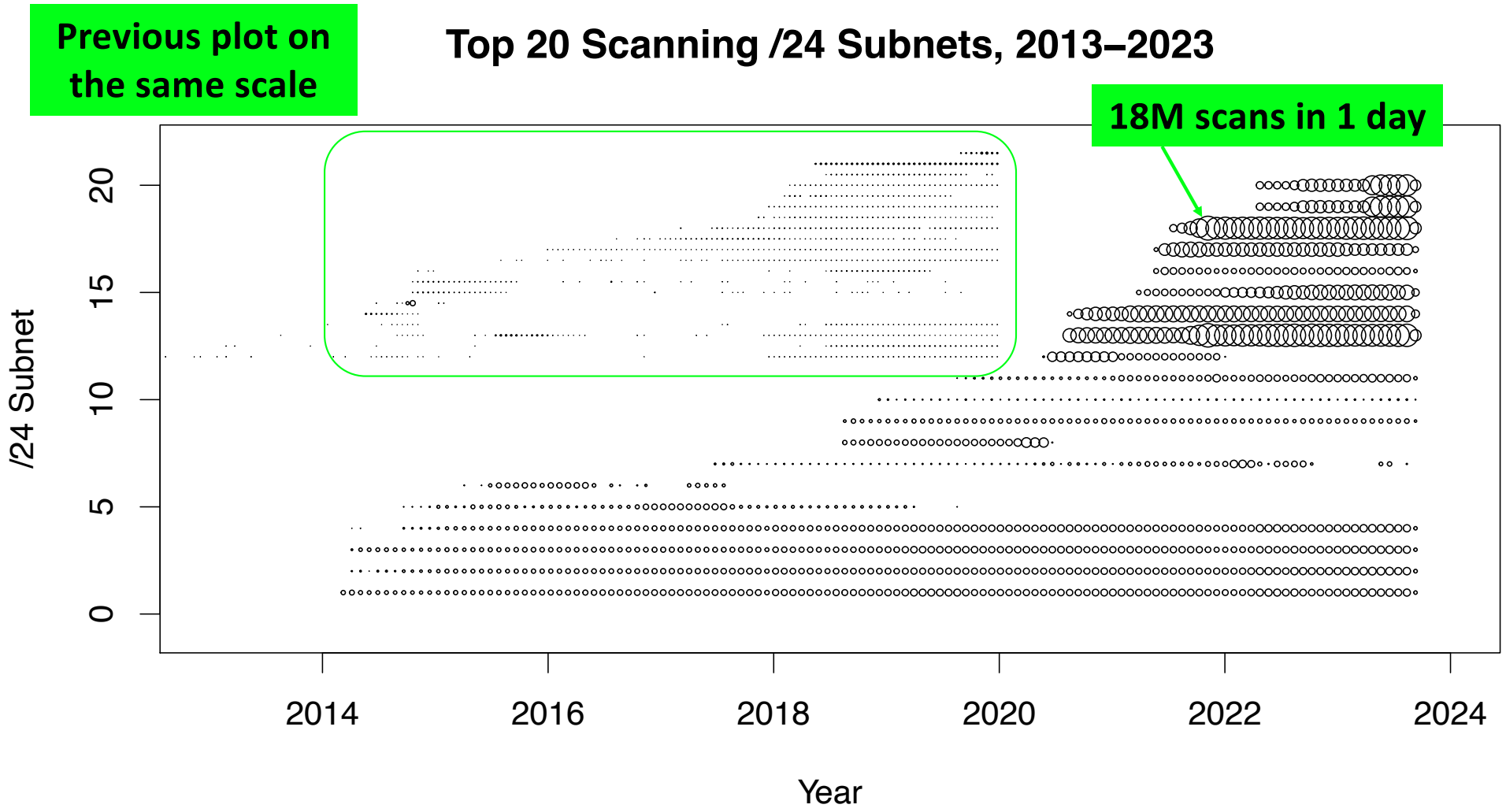
Proportion of *connections* as of late 2023

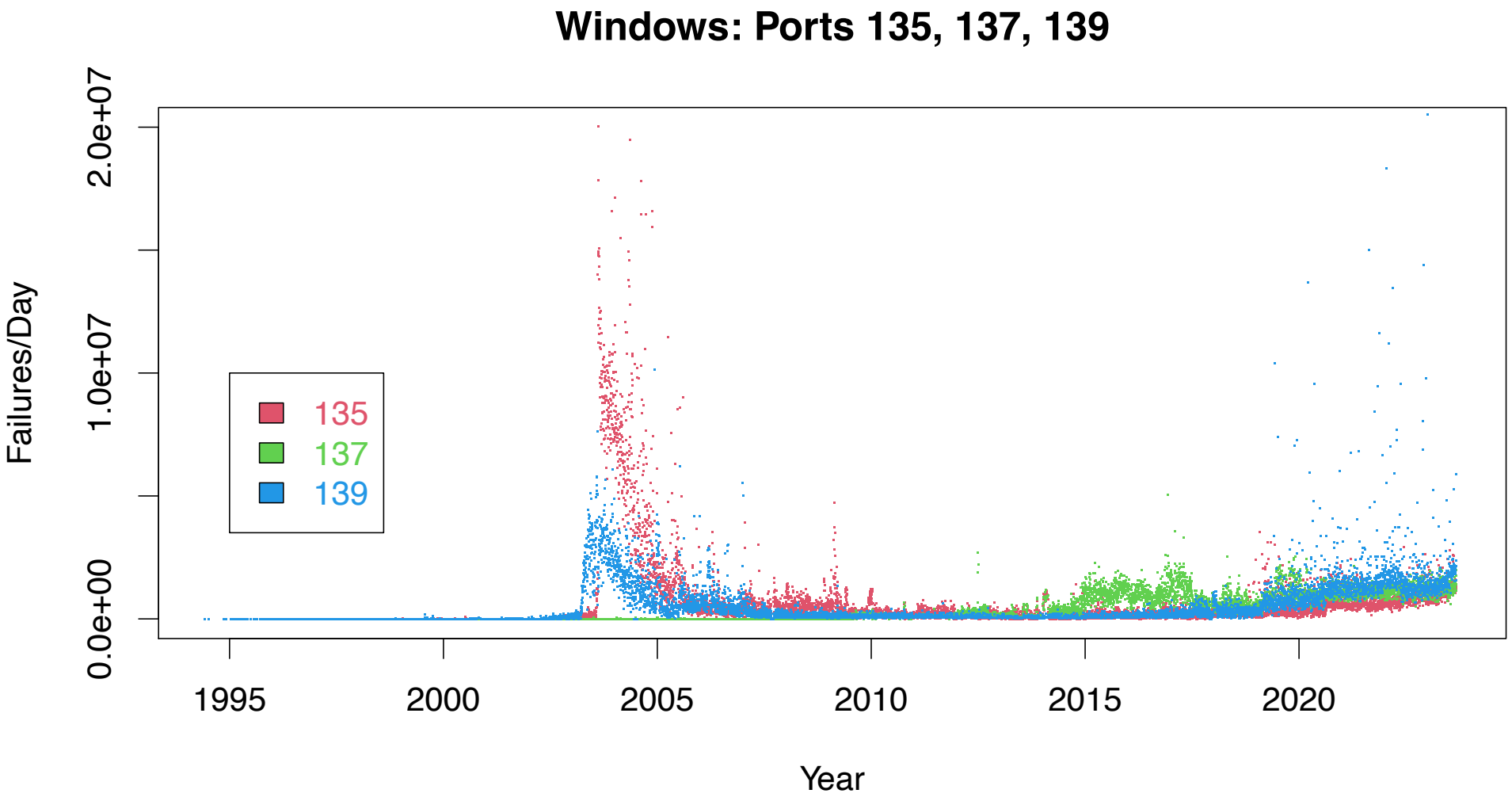

IPv6 Scanners
Distinct Scanners
0
500
1000
1500
2000
2014
2016
2018
2020
2022
2024
Year

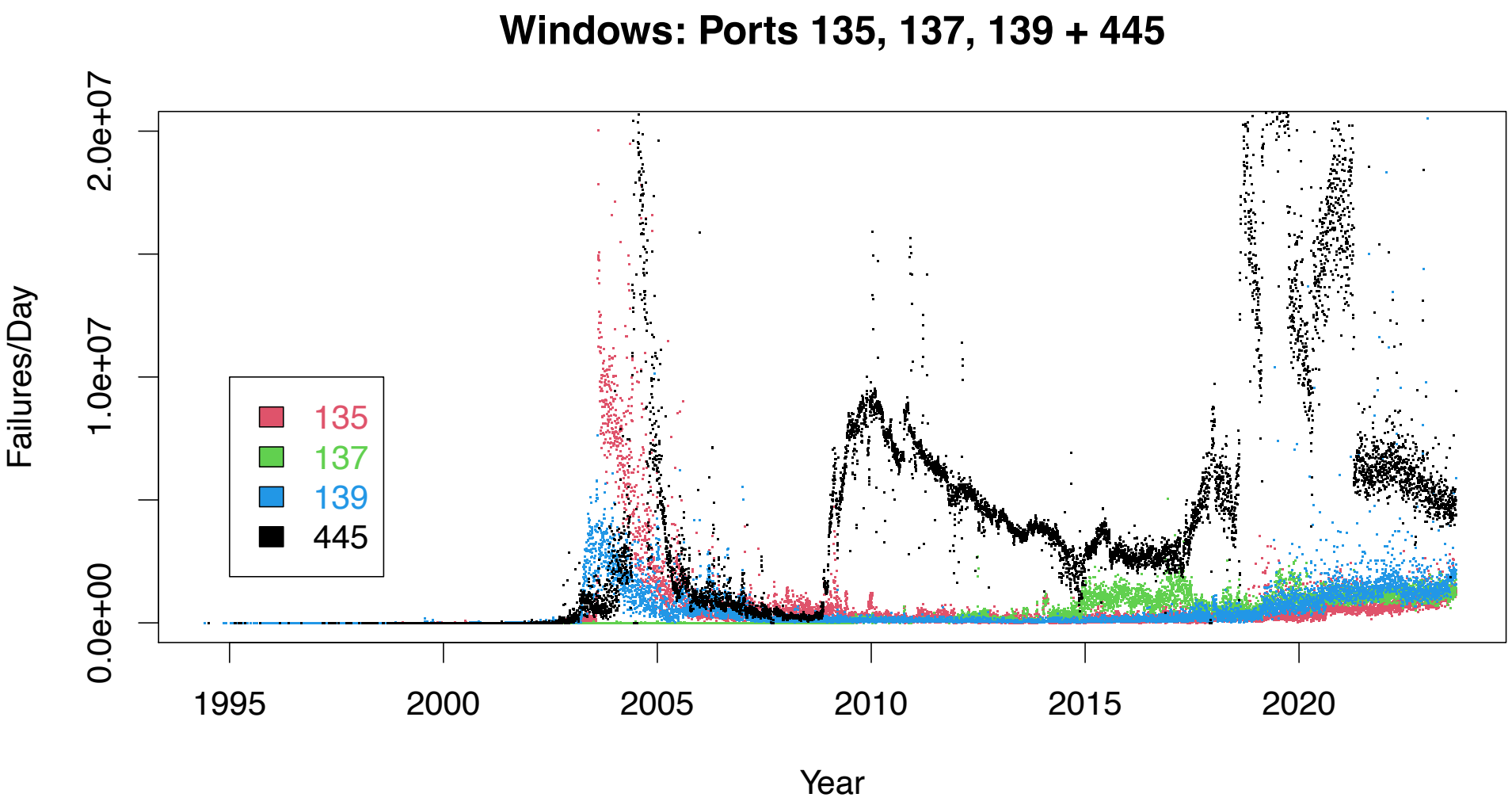

Top 20 Scanning /24 Subnets, 2003–2013
868K scans in 1 day
/24 Subnet
0
5
10
15
20
2004
2006
2008
2010
2012
Year

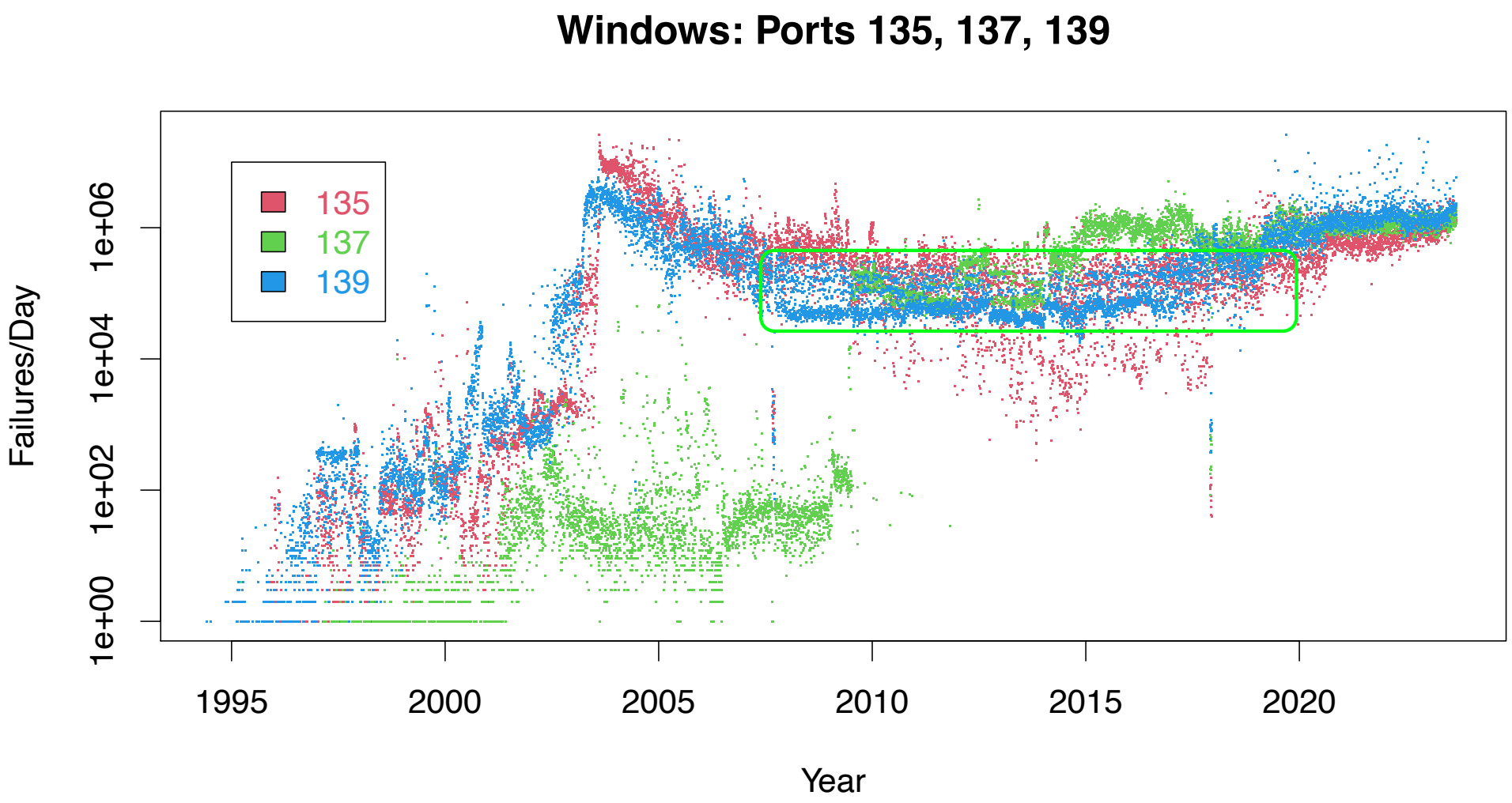
No overlap with previous decade!

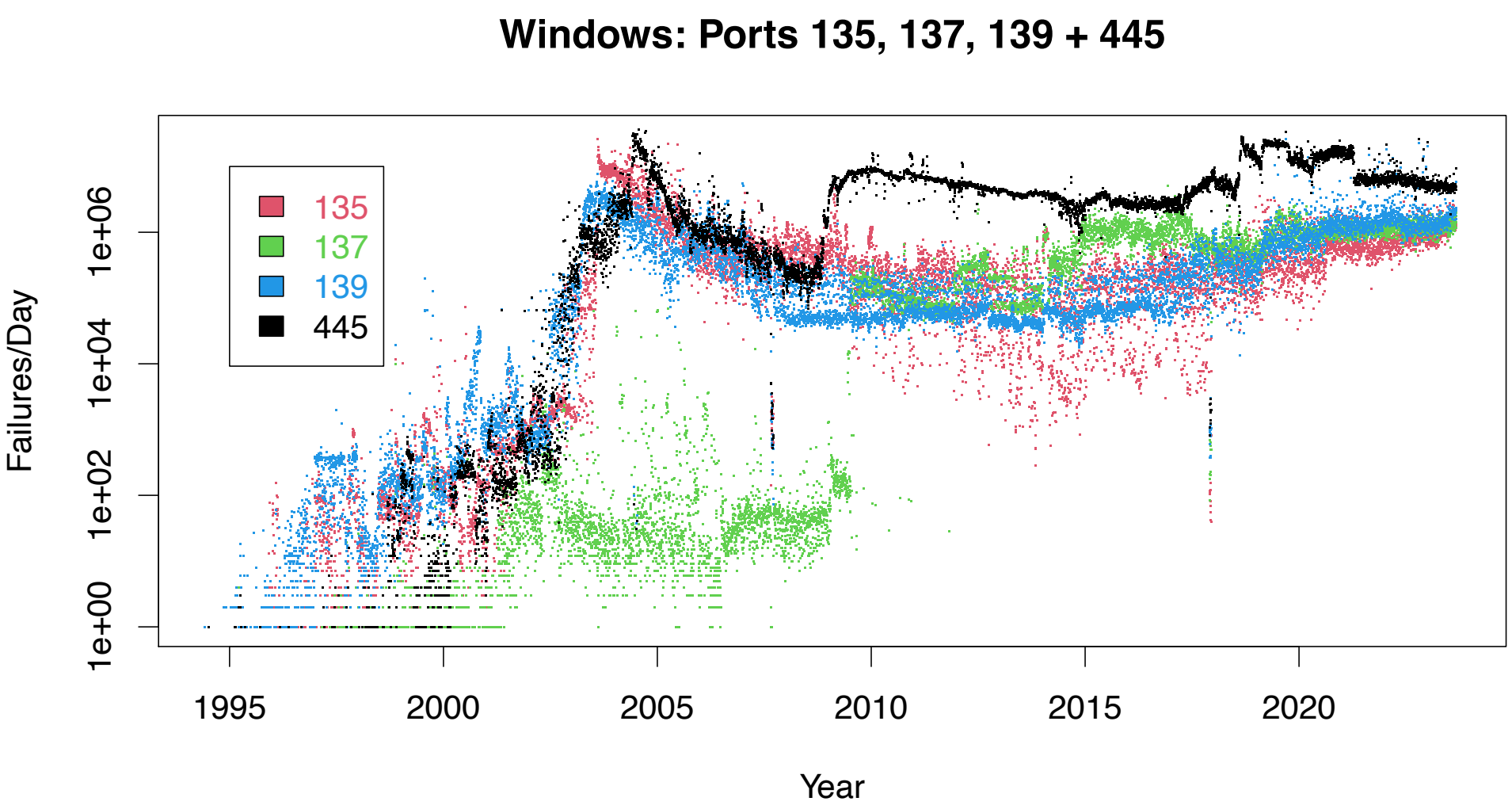


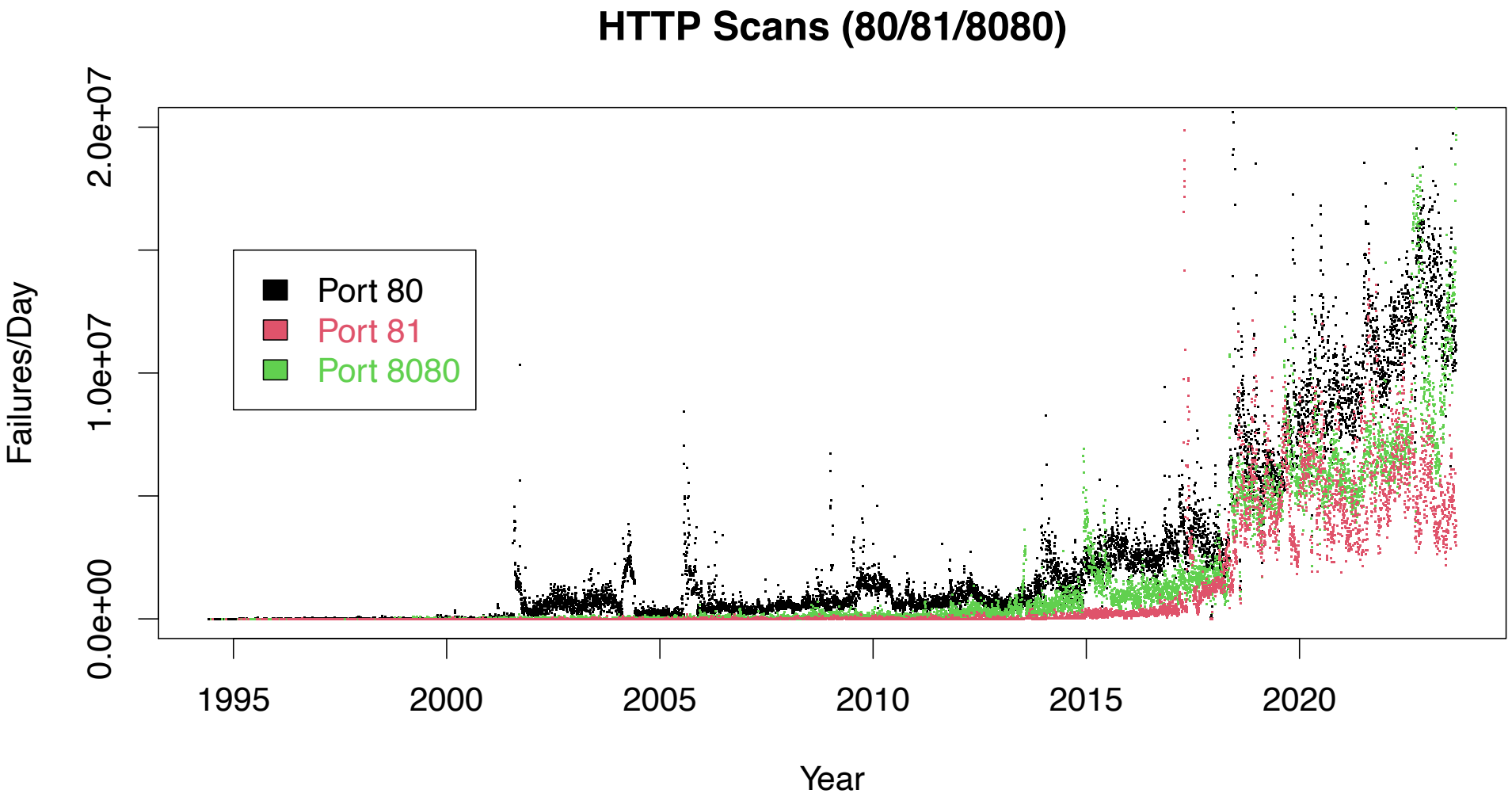

Previous plot on the same scale
Top 20 Scanning /24 Subnets, 2013–2023
18M scans in 1 day
/24 Subnet
0
5
10
15
20
2014
2016
2018
2020
2022
2024
Year

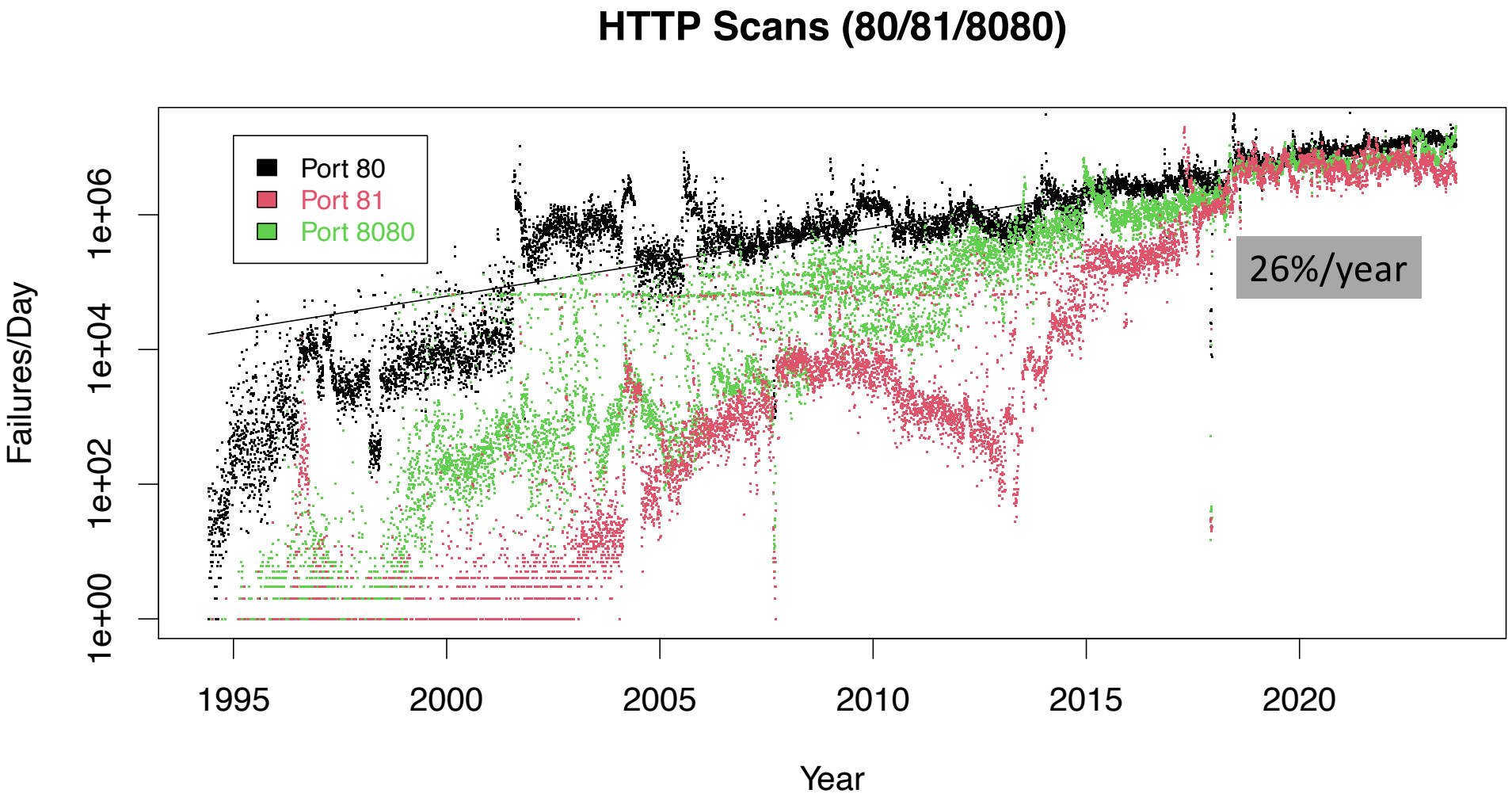

Windows: Ports 135, 137, 139
Failures/Day
0.0e+00
1.0e+07
2.0e+07
135
137
139
1995
2000
2005
2010
2015
2020
Year

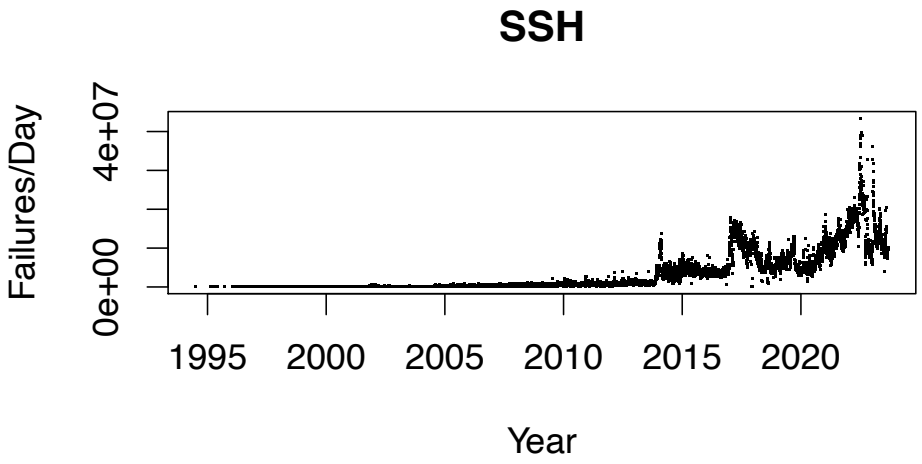

Windows: Ports 135, 137, 139 + 445
Failures/Day
0.0e+00
1.0e+07
2.0e+07
135
137
139
445
1995
2000
2005
2010
2015
2020
Year

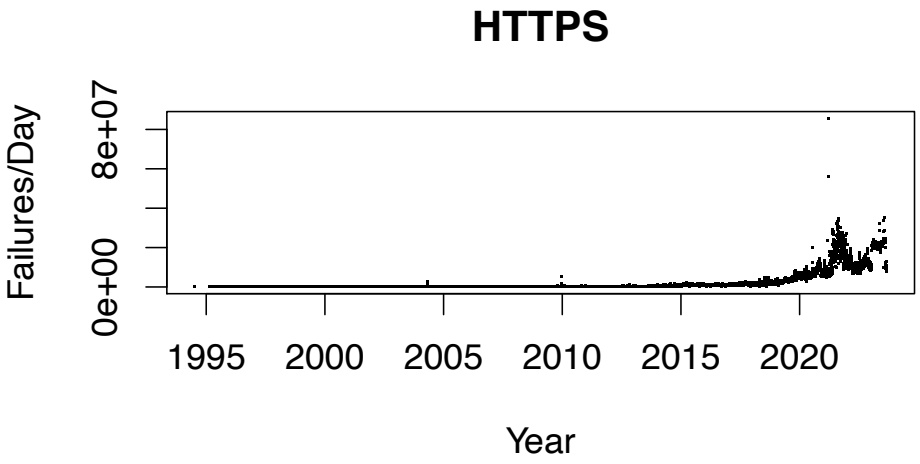

Windows: Ports 135, 137, 139
135
137
139
Failures/Day
1e+00
1e+02
1e+04
1e+06
1995
2000
2005
2010
2015
2020
Year

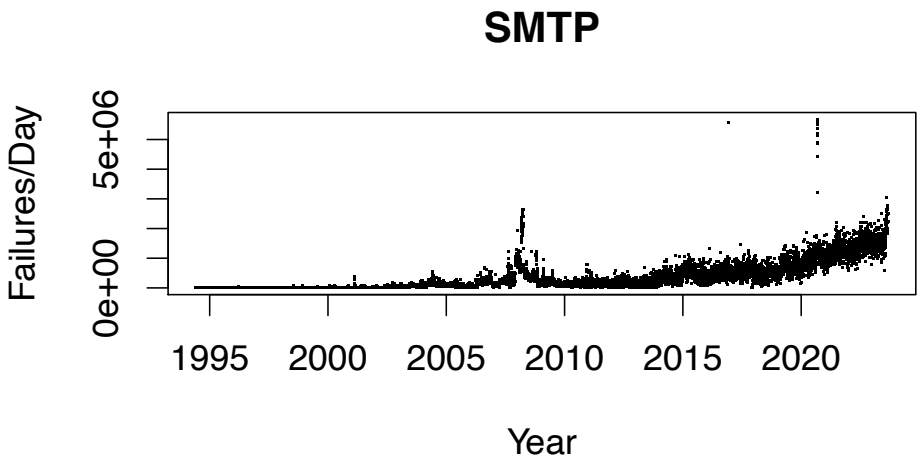

Windows: Ports 135, 137, 139 + 445
135
137
139
445
Failures/Day
1e+00
1e+02
1e+04
1e+06
1995
2000
2005
2010
2015
2020
Year

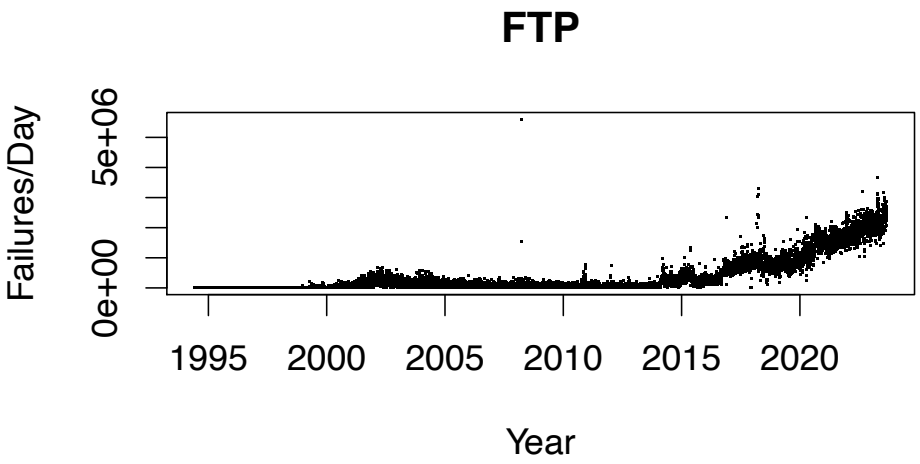

HTTP Scans (80/81/8080)
Failures/Day
0.0e+00
1.0e+07
2.0e+07
Port 80
Port 81
Port 8080
1995
2000
2005
2010
2015
2020
Year

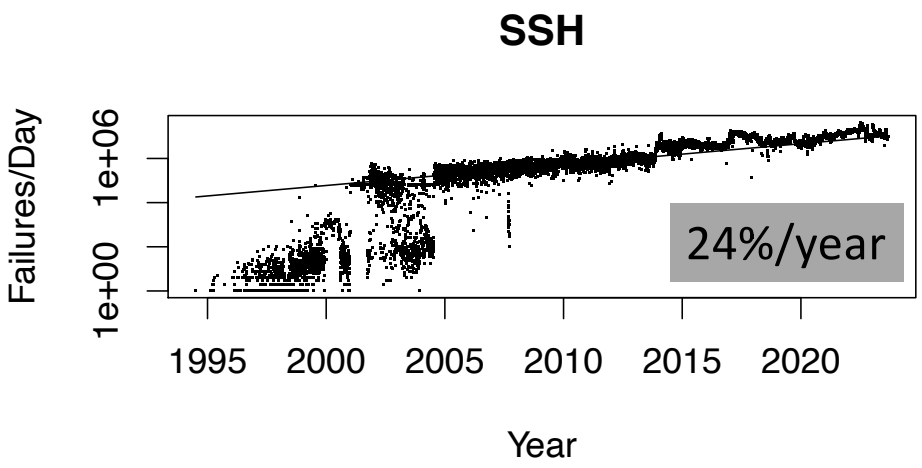

HTTP Scans (80/81/8080)
Port 80
Port 81
Port 8080
26%/year
Failures/Day
1e+00
1e+02
1e+04
1e+06
1995
2000
2005
2010
2015
2020
Year

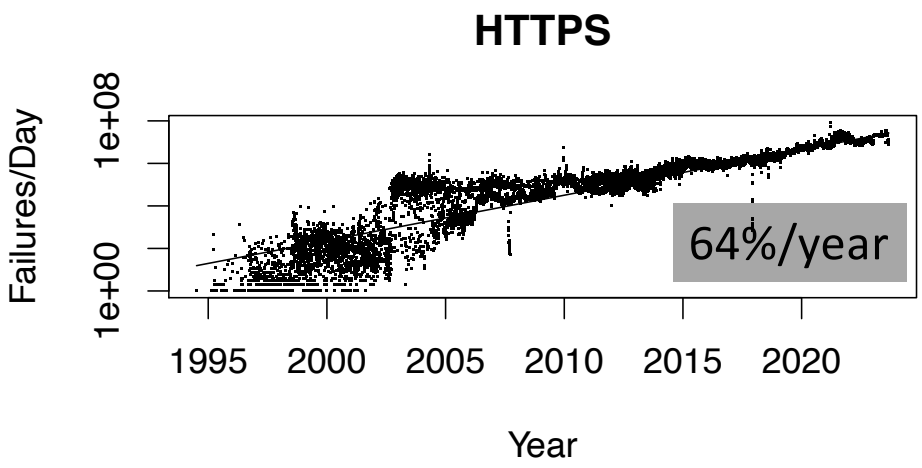

SSH
Failures/Day
0e+00
4e+07
1995
2000
2005
2010
2015
2020
Year

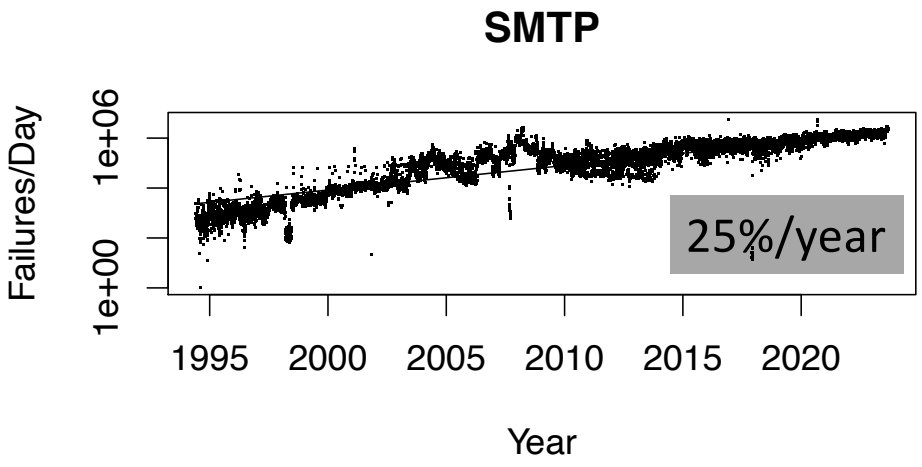

HTTPS
Failures/Day
0e+00
8e+07
1995
2000
2005
2010
2015
2020
Year

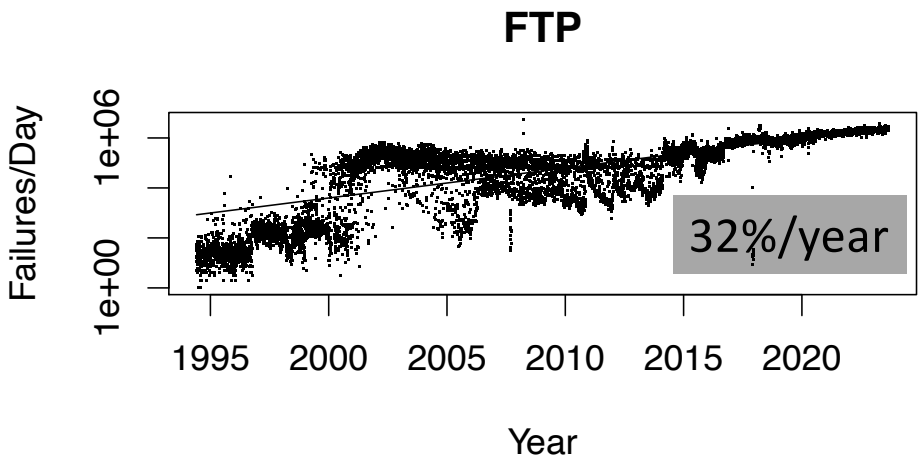

SMTP
Failures/Day
0e+00
5e+06
1995
2000
2005
2010
2015
2020
Year

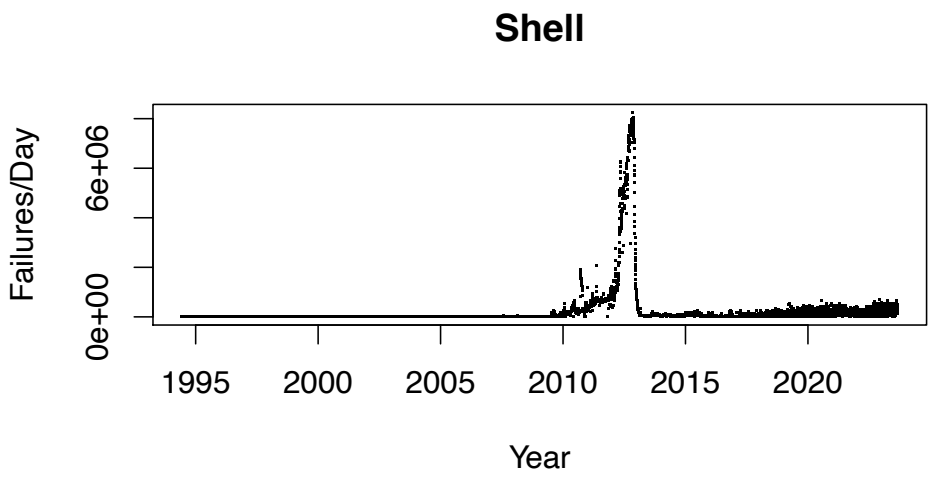

FTP
Failures/Day
0e+00
5e+06
1995
2000
2005
2010
2015
2020
Year

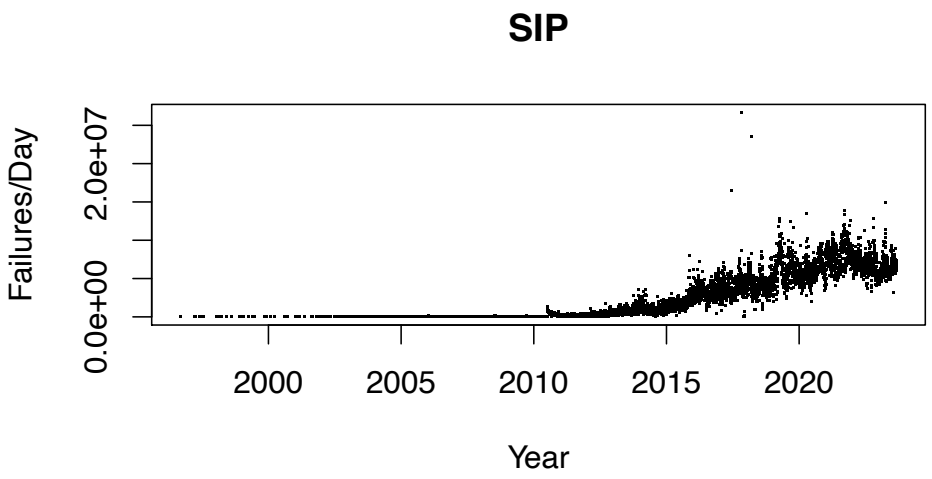

SSH
Failures/Day
1e+00
1e+06
24%/year
1995
2000
2005
2010
2015
2020
Year

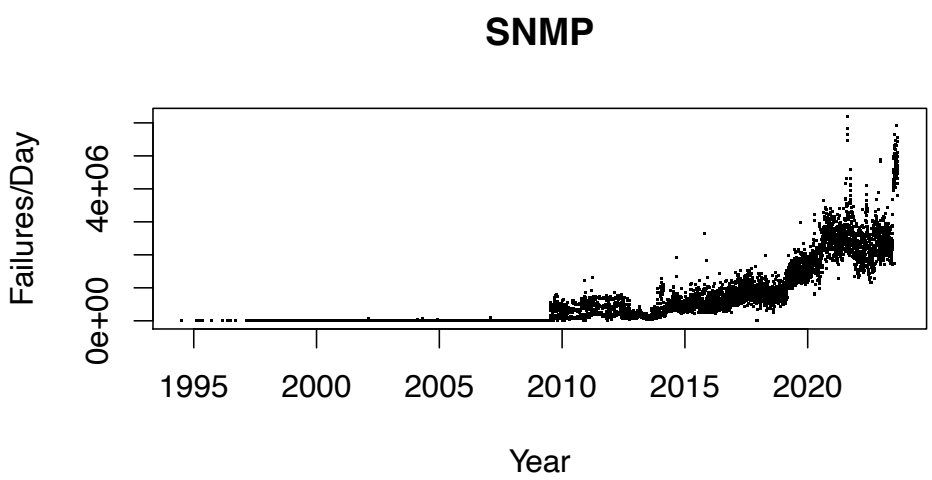

HTTPS
Failures/Day
1e+00
1e+08
64%/year
1995
2000
2005
2010
2015
2020
Year

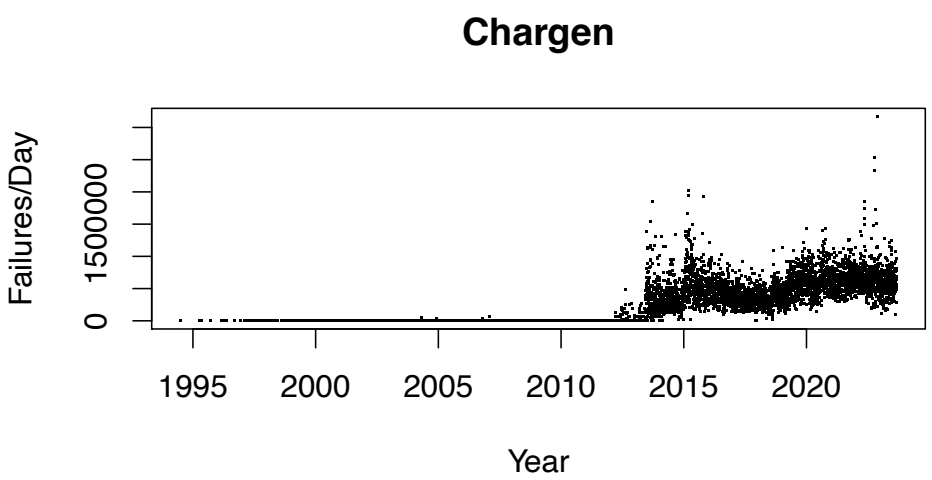

SMTP
Failures/Day
1e+00
1e+06
25%/year
1995
2000
2005
2010
2015
2020
Year

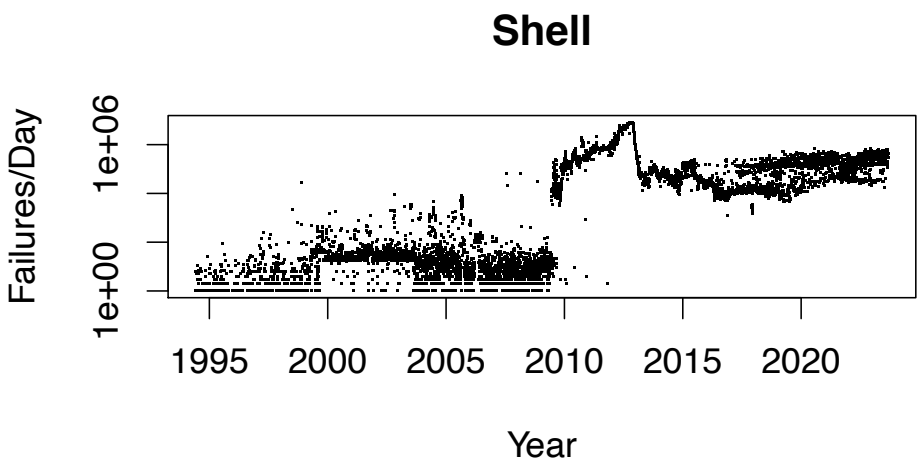

FTP
Failures/Day
1e+00
1e+06
32%/year
1995
2000
2005
2010
2015
2020
Year

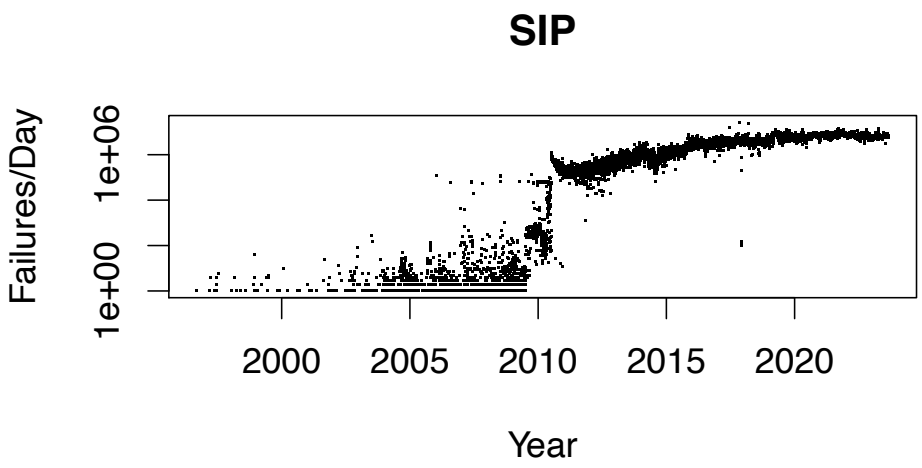

Shell
Failures/Day
0e+00
6e+06
1995
2000
2005
2010
2015
2020
Year

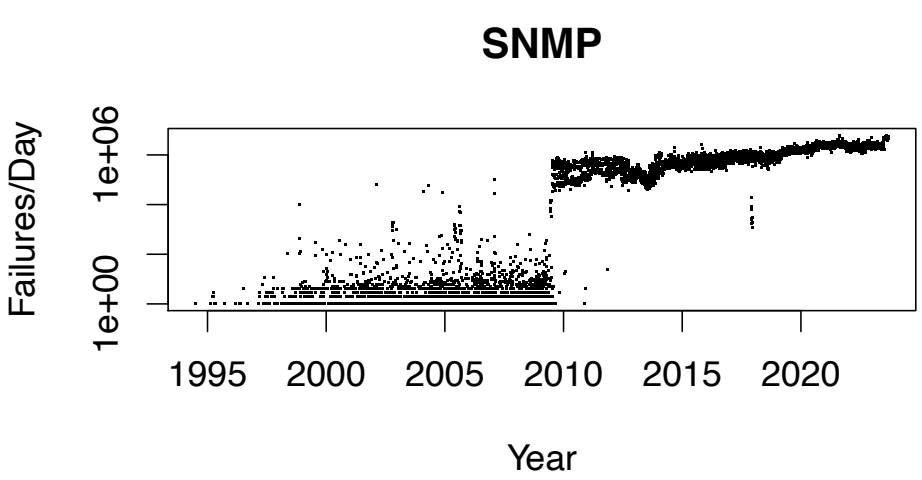

SIP
Failures/Day
0.0e+00
2.0e+07
2000
2005
2010
2015
2020
Year

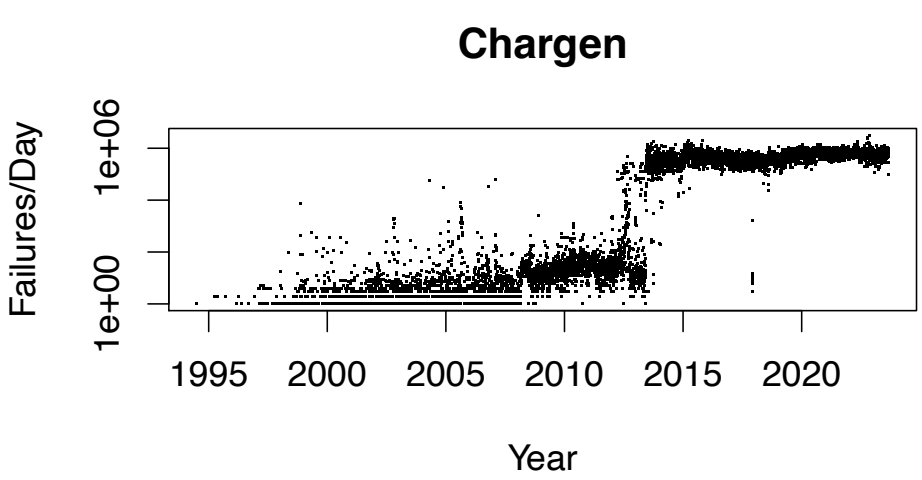

SNMP
Failures/Day
0e+00
4e+06
1995
2000
2005
2010
2015
2020
Year

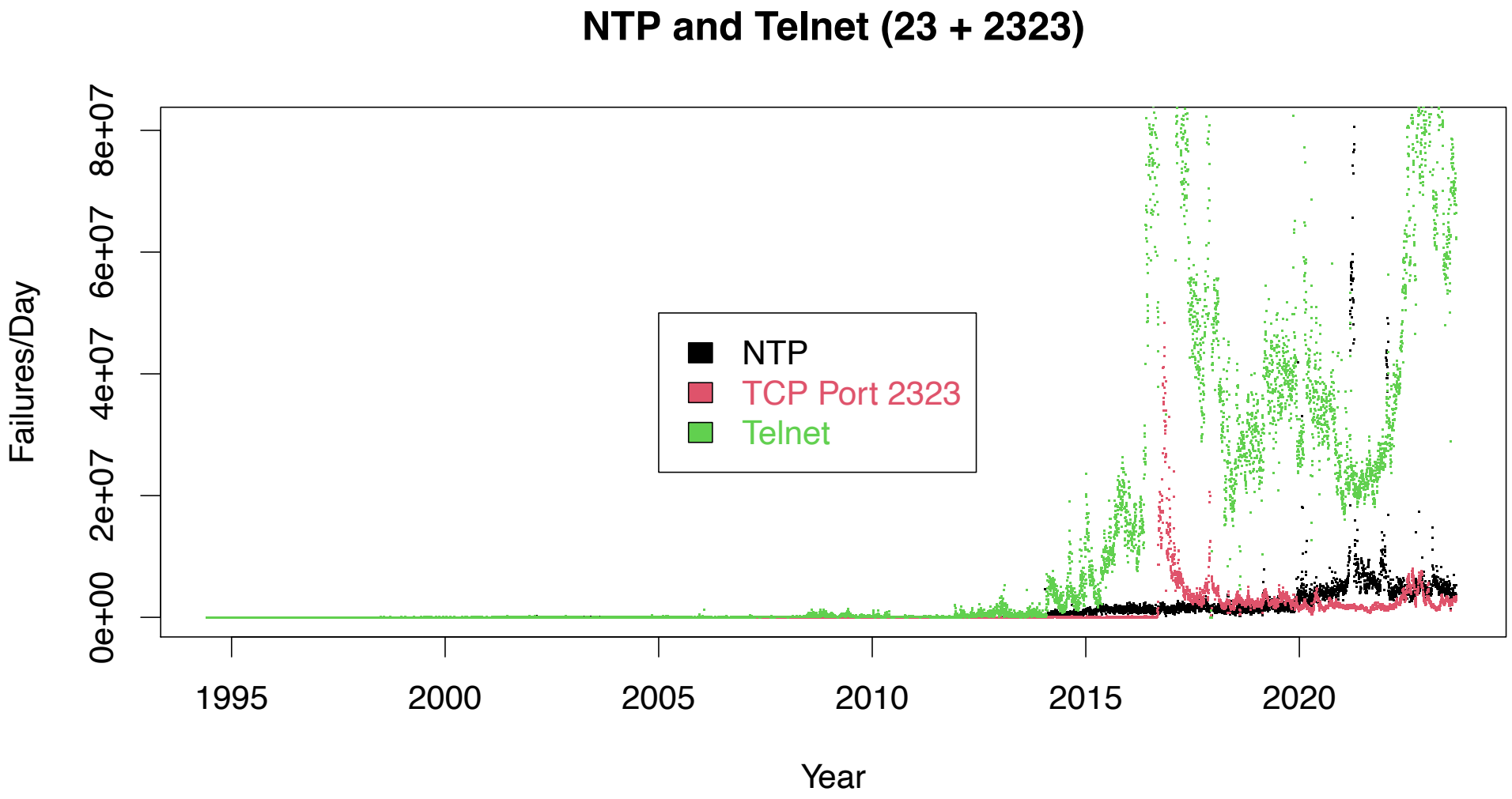

Chargen
Failures/Day
0
1500000
1995
2000
2005
2010
2015
2020
Year

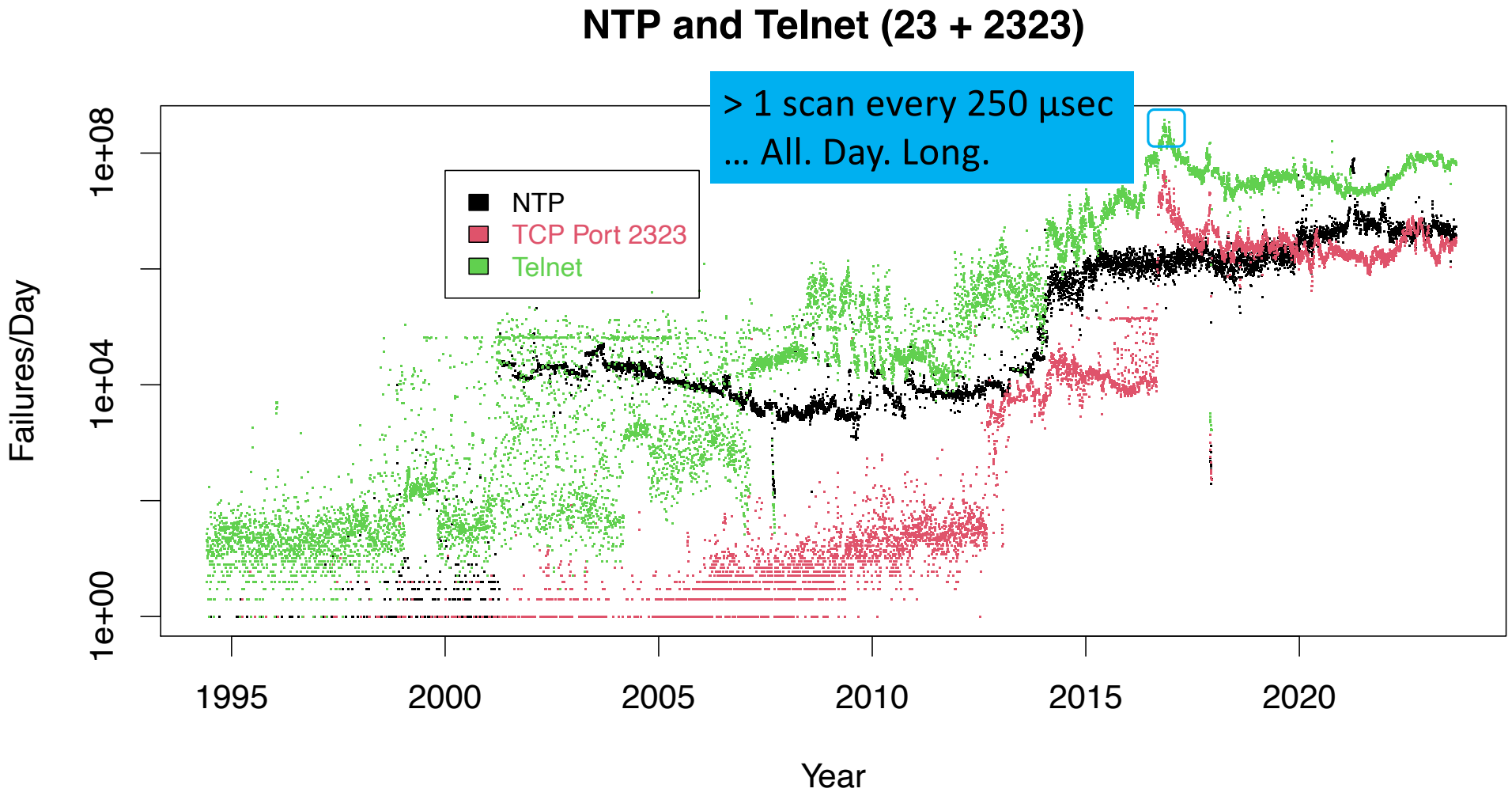

Shell
Failures/Day
1e+00
1e+06
1995
2000
2005
2010
2015
2020
Year

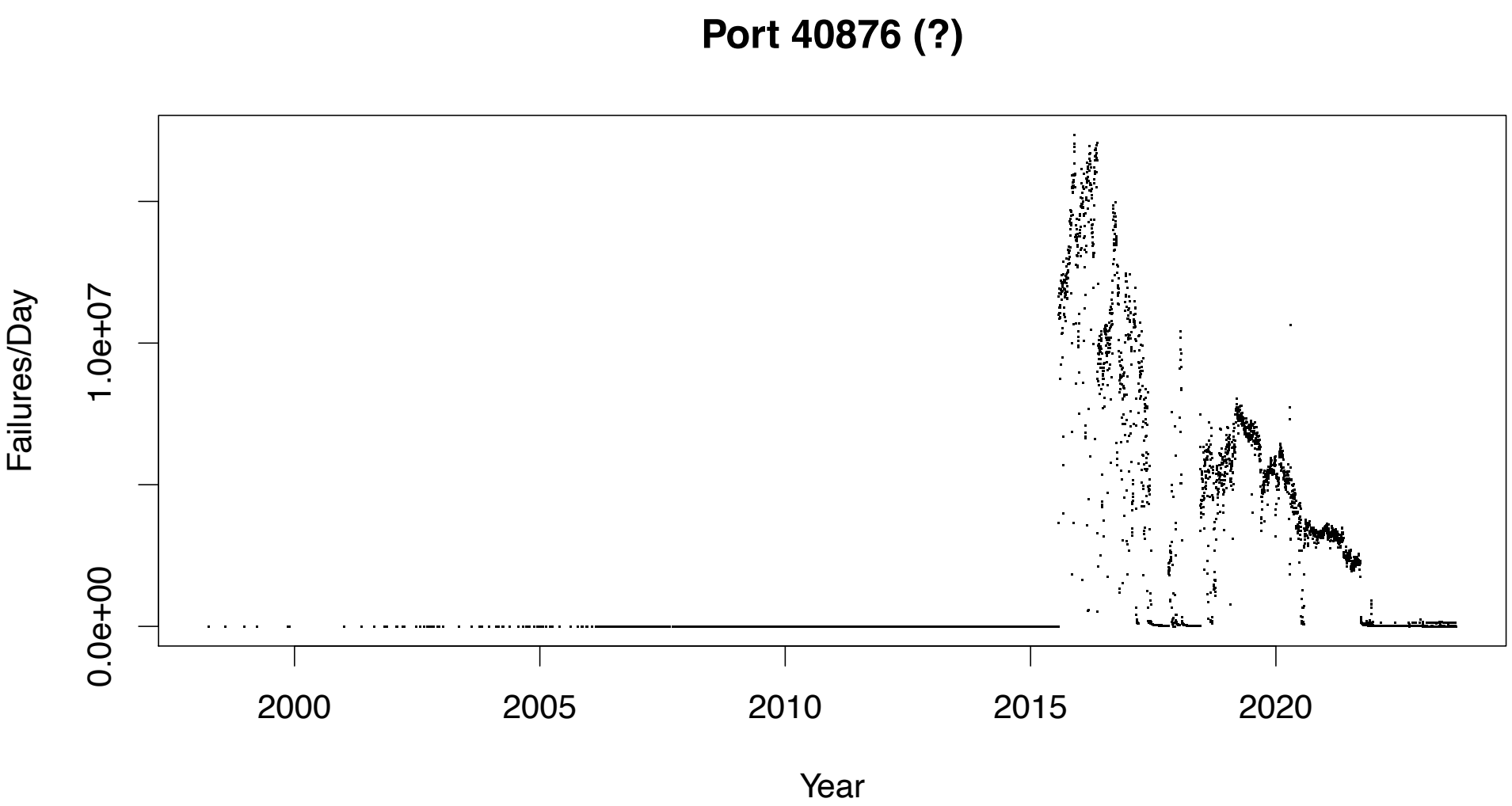

SIP
Failures/Day
1e+00
1e+06
2000
2005
2010
2015
2020
Year

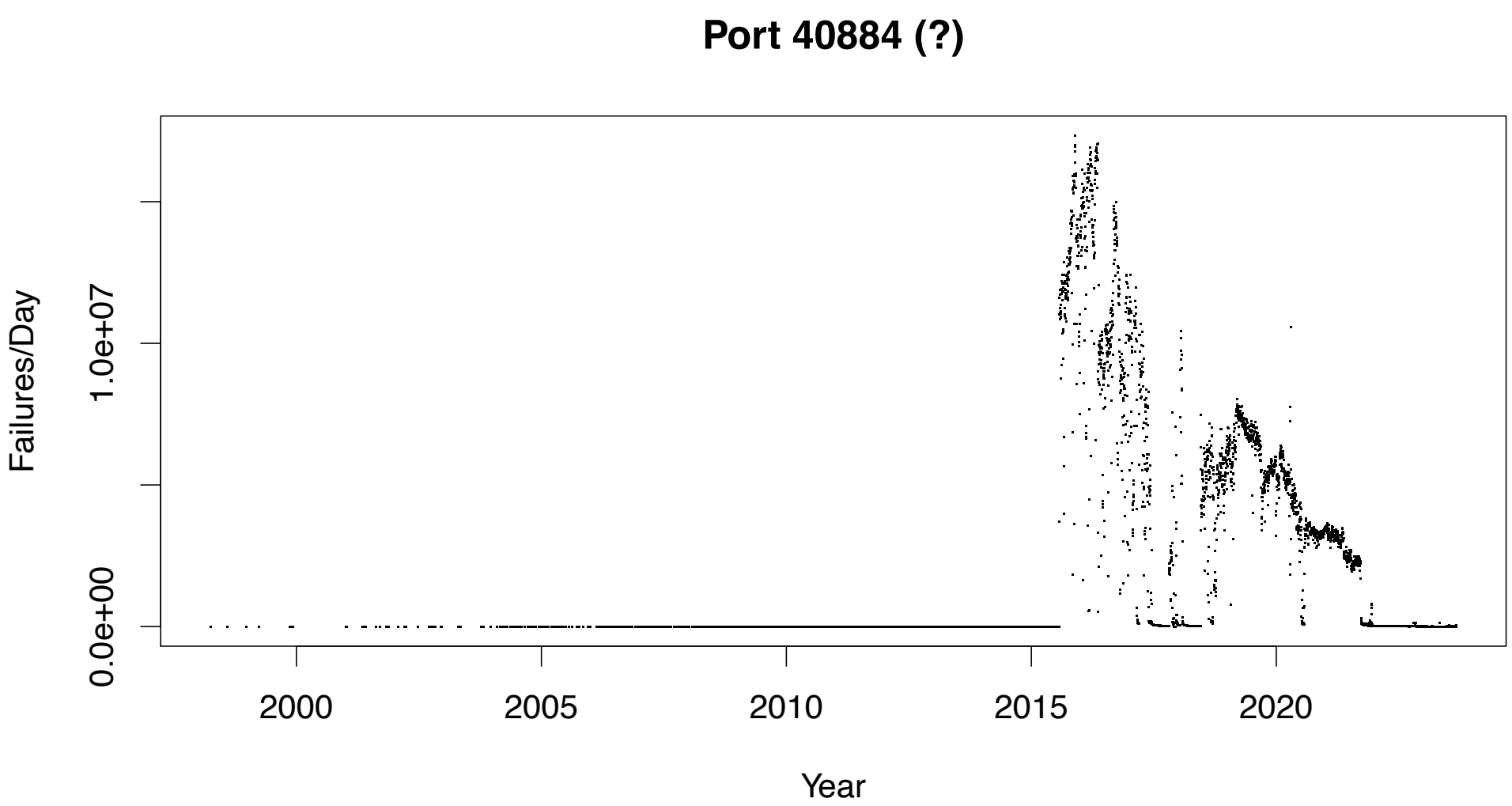

SNMP
Failures/Day
1e+00
1e+06
1995
2000
2005
2010
2015
2020
Year

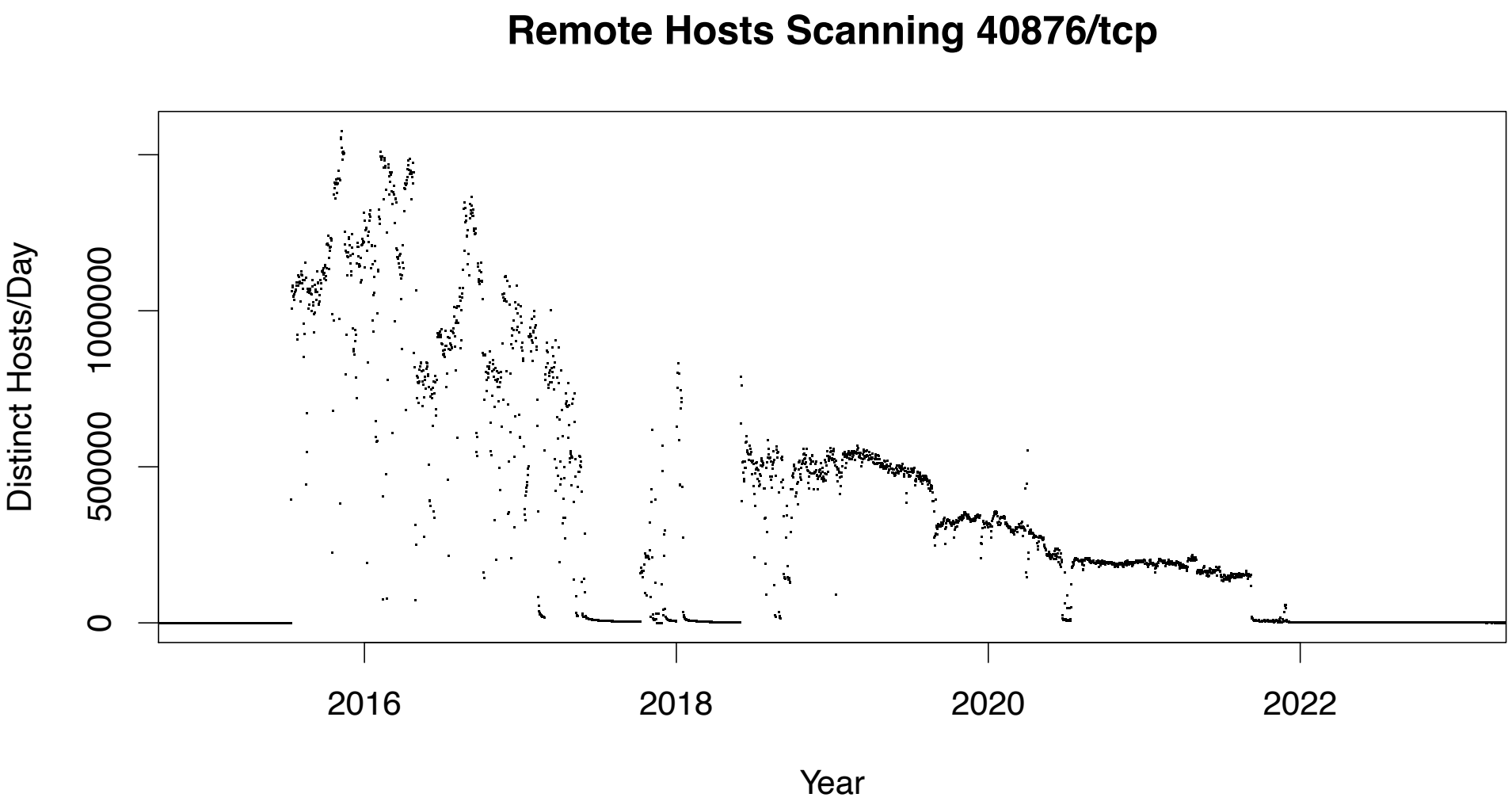

Chargen
Failures/Day
1e+00
1e+06
1995
2000
2005
2010
2015
2020
Year

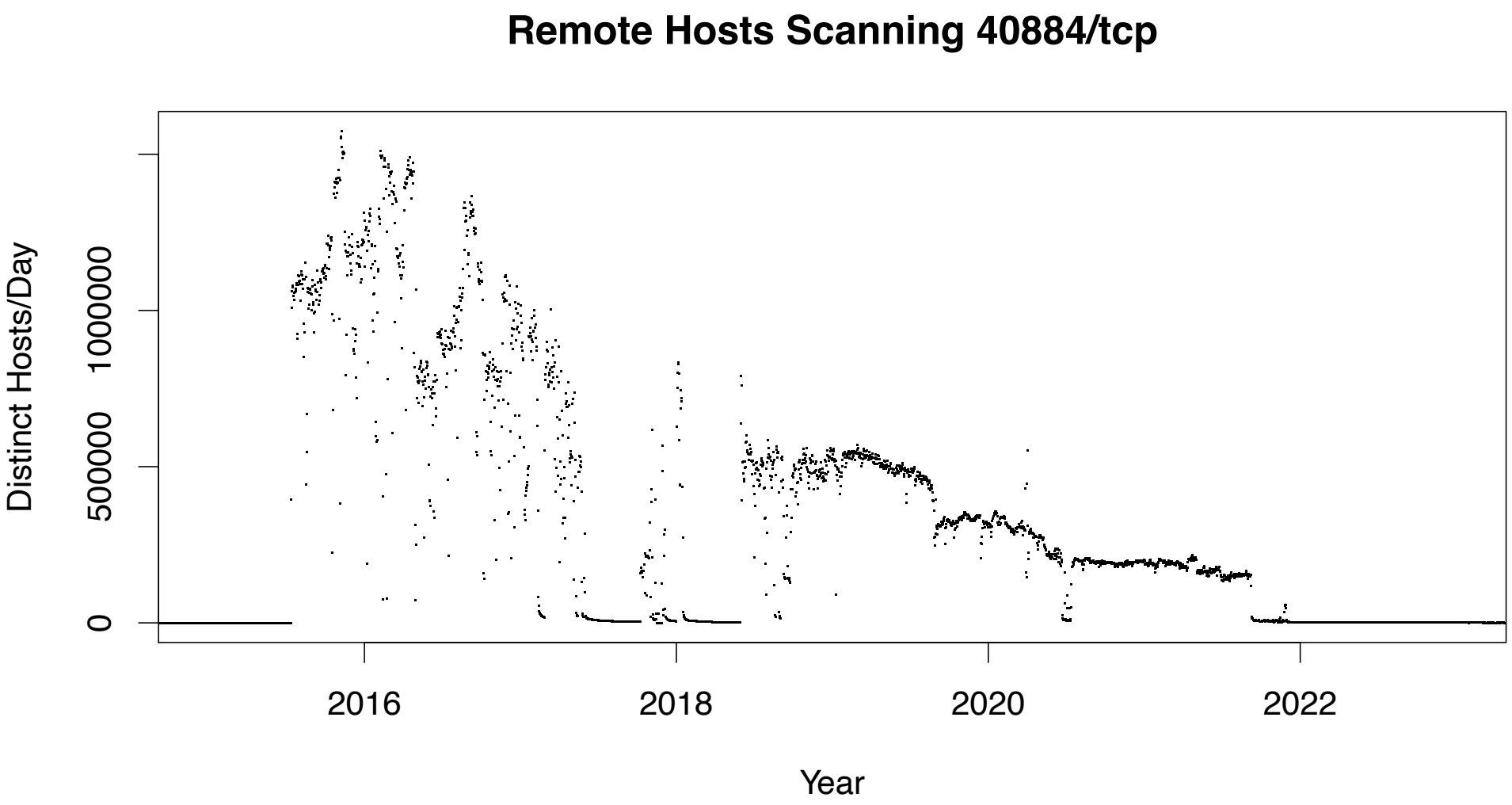

NTP and Telnet (23 + 2323)
Failures/Day
0e+00
2e+07
4e+07
6e+07
8e+07
1995
2000
2005
2010
2015
2020
Year
NTP
TCP Port 2323
Telnet

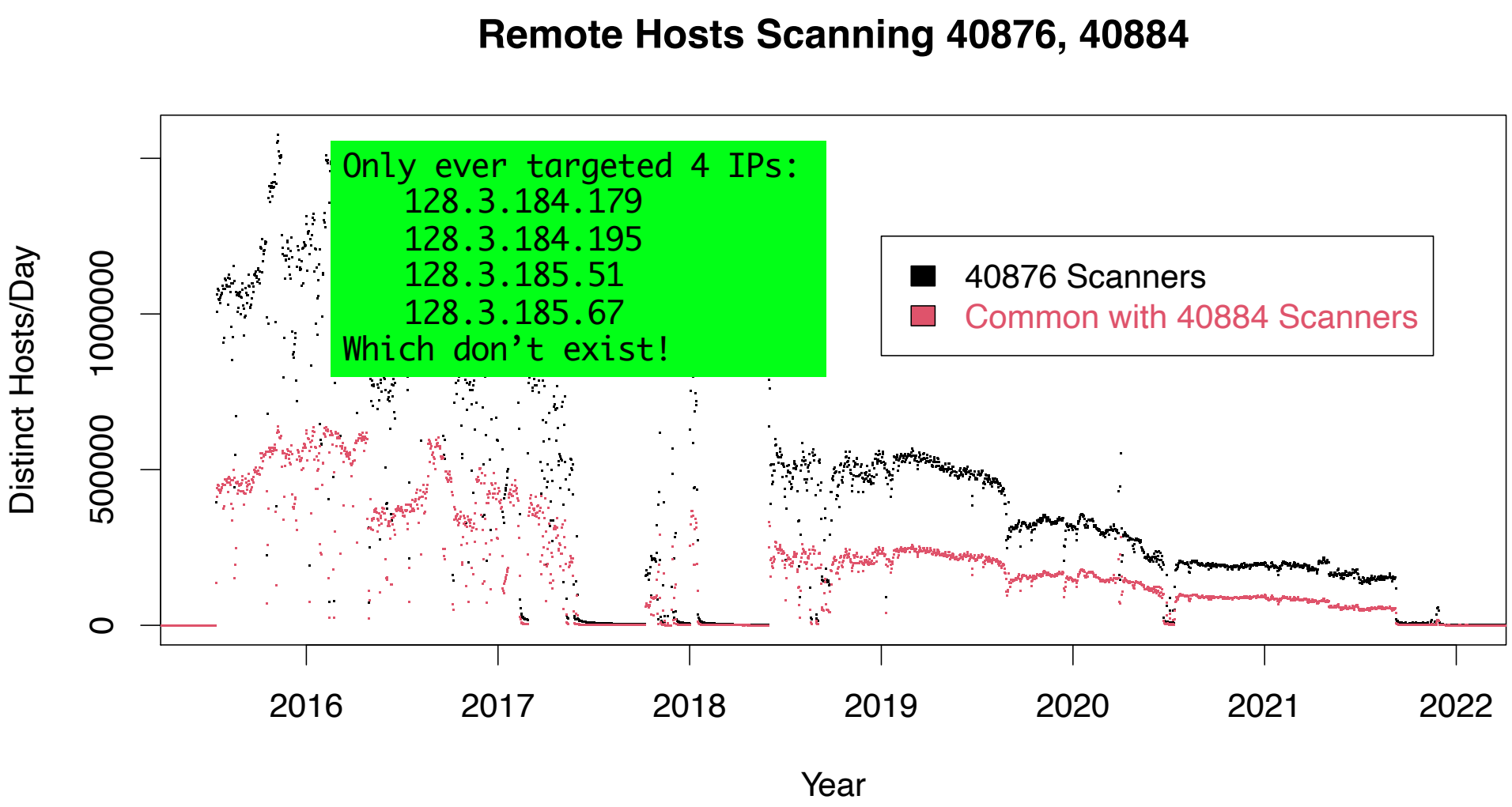

NTP and Telnet (23 + 2323)
> 1 scan every 250 µsec
… All. Day. Long.
NTP
TCP Port 2323
Telnet
Failures/Day
1e+00
1e+04
1e+08
1995
2000
2005
2010
2015
2020
Year


Port 40876 (?)
Failures/Day
0.0e+00
1.0e+07
2000
2005
2010
2015
2020
Year

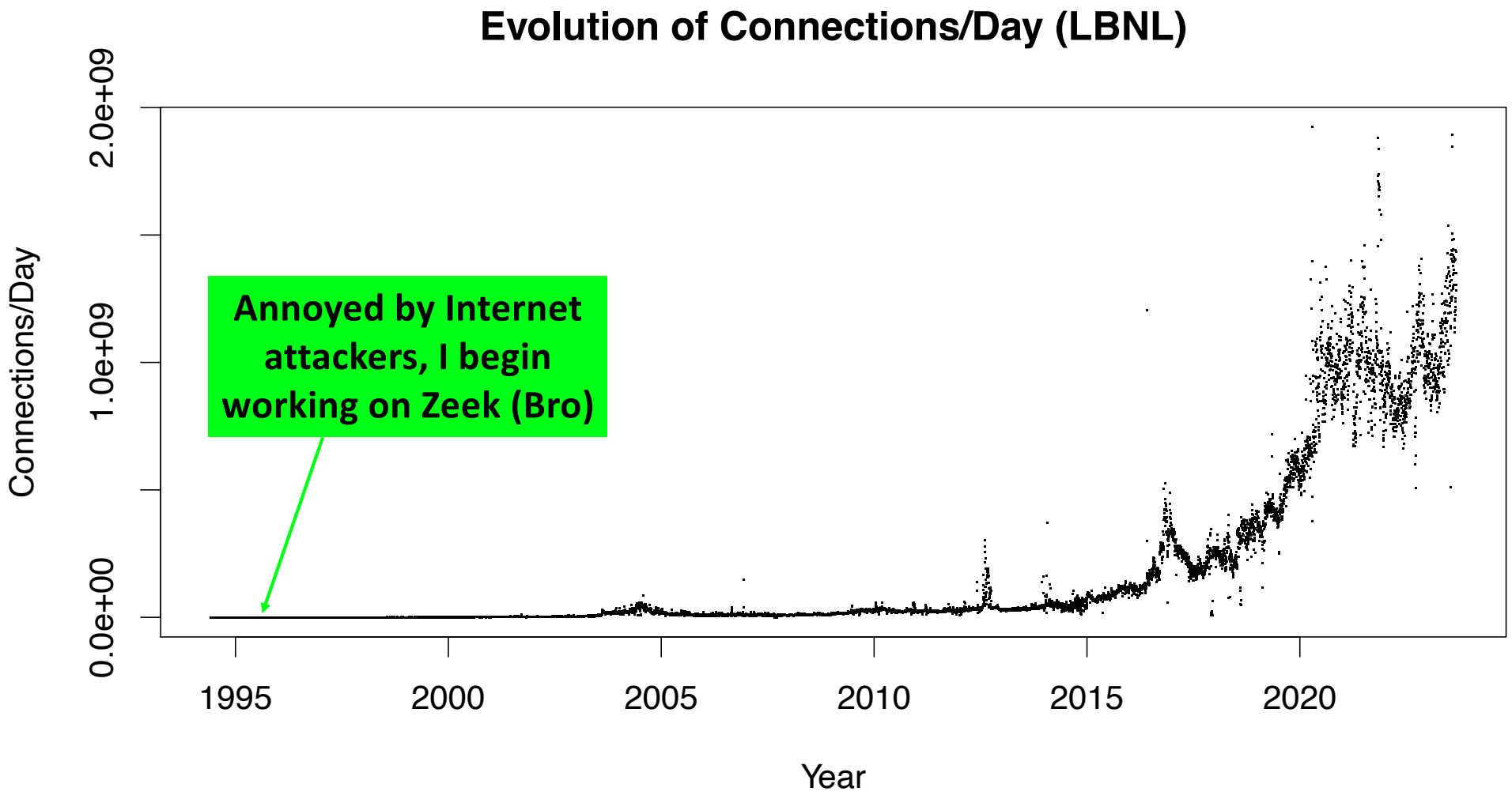

Port 40884 (?)
Failures/Day
0.0e+00
1.0e+07
2000
2005
2010
2015
2020
Year


Remote Hosts Scanning 40876/tcp
Distinct Hosts/Day
0
500000
1000000
2016
2018
2020
2022
Year


Remote Hosts Scanning 40884/tcp
Distinct Hosts/Day
0
500000
1000000
2016
2018
2020
2022
Year


Remote Hosts Scanning 40876, 40884
Only ever targeted 4 IPs:
128.3.184.179
128.3.184.195
128.3.185.51
128.3.185.67
Which don't exist!
40876 Scanners
Common with 40884 Scanners
Distinct Hosts/Day
0
500000
1000000
2016
2017
2018
2019
2020
2021
2022
Year

Enterprises:
Undertakings by individuals/groups


Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
Annoyed by Internet attackers, I begin working on Zeek (Bro)
1995
2000
2005
2010
2015
2020
Year

Bro: A System for Detecting Network Intruders in Real-Time

Authors:

Vern Paxson, Lawrence Berkeley National Laboratory

Abstract:

We describe Bro, a stand-alone system for detecting network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits. We give an overview of the system's design, which emphasizes high-speed (FDDI-rate) monitoring, real-time notification, clear separation between mechanism and policy, and extensibility. To achieve these ends, Bro is divided into an "event engine" that reduces a kernel-filtered network traffic stream into a series of higher-level events, and a "policy script interpreter" that interprets event handlers written in a specialized language used to express a site's security policy. Event handlers can update state information, synthesize new events, record information to disk, and generate real-time notifications via *syslog*. We also discuss a number of attacks that attempt to subvert passive monitoring systems and defenses against these, and give particulars of how Bro analyzes the four applications integrated into it so far: Finger, FTP, Portmapper and Telnet. The system is publicly available in source code form.

7th USENIX Security Symposium, 1998

JANUARY 26–29, 1998 • SAN ANTONIO, TX, USA

USENIX

~~Bro~~: A System for ~~Detecting Network Intruders in Real-Time~~

Zeek High-Performance Generating Detailed Network Meta-Data (that's open source)

Authors:
Vern Paxson, Lawrence Berkeley National Laboratory

Abstract:
We describe Bro, a stand-alone system for detecting network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits. We give an overview of the system's design, which emphasizes high-speed (FDDI-rate) monitoring, real-time notification, clear separation between mechanism and policy, and extensibility. To achieve these ends, Bro is divided into an "event engine" that reduces a kernel-filtered network traffic stream into a series of higher-level events, and a "policy script interpreter" that interprets event handlers written in a specialized language used to express a site's security policy. Event handlers can update state information, synthesize new events, record information to disk, and generate real-time notifications via *syslog*. We also discuss a number of attacks that attempt to subvert passive monitoring systems and defenses against these, and give particulars of how Bro analyzes the four applications integrated into it so far: Finger, FTP, Portmapper and Telnet. The system is publicly available in source code form.

7th USENIX Security Symposium, 1998

JANUARY 26–29, 1998 • SAN ANTONIO, TX, USA

USENIX

ZEEK® LOGS

conn.log | IP, TCP, UDP, ICMP connection details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of first packet
uid	string	Unique identifier of connection
id	record conn_id	Connection's 4-tuple of endpoints
> id.orig_h	addr	IP address of system initiating connection
> id.orig_p	port	Port from which the connection is initiated
> id.resp_h	addr	IP address of system responding to connection request
> id.resp_p	port	Port on which connection response is sent
proto	enum	Transport layer protocol of connection
service	string	Application protocol ID sent over connection
duration	interval	How long connection lasted
orig_bytes	count	Number of payload bytes originator sent
resp_bytes	count	Number of payload bytes responder sent
conn_state	string	Connection state (see conn.log > conn_state)
local_orig	bool	Value=T if connection originated locally
local_resp	bool	Value=T if connection responded locally
missed_bytes	count	Number of bytes missed (packet loss)
history	string	Connection state history (see conn.log > history)
orig_pkts	count	Number of packets originator sent
orig_ip_bytes	count	Number of originator IP bytes (via IP total_length header field)
resp_pkts	count	Number of packets responder sent
resp_ip_bytes	count	Number of responder IP bytes (via IP total_length header field)
tunnel_parents	table	If tunneled, connection UID value of encapsulating parent(s)
orig_l2_addr	string	Link-layer address of originator
resp_l2_addr	string	Link-layer address of responder
vlan	int	Outer VLAN for connection
inner_vlan	int	Inner VLAN for connection

irc.log | IRC communication details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when command seen
uid & id		Underlying connection info > See conn.log
nick	string	Nickname given for connection
user	string	Username given for connection
command	string	Command given by client
value	string	Value for command given by client
addl	string	Any additional data for command
dcc_file_name	string	DCC filename requested
dcc_file_size	count	DCC transfer size as indicated by sender
dcc_mime_type	string	Sniffed mime type of file
fuid	string	File unique ID

dce_rpc.log | Details on DCE/RPC messages

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened

dhcp.log | DHCP lease activity

FIELD	TYPE	DESCRIPTION
ts	time	Earliest time DHCP message observed
uids	table	Unique identifiers of DHCP connections
client_addr	addr	IP address of client
server_addr	addr	IP address of server handing out lease
client_port	port	Client port at time of server handing out IP
server_port	port	Server port at time of server handing out IP
mac	string	Client's hardware address
host_name	string	Name given by client in Hostname option 12
client_fqdn	string	FQDN given by client in Client FQDN option 81
domain	string	Domain given by server in option 15
requested_addr	addr	IP address requested by client
assigned_addr	addr	IP address assigned by server
lease_time	interval	IP address lease interval
client_message	string	Message with DHCP_DECLINE so client can tell server why address was rejected
server_message	string	Message with DHCP_NAK to let client know why request was rejected
msg_types	vector	DHCP message types seen by transaction
duration	interval	Duration of DHCP session
client_chaddr	string	Hardware address reported by the client
msg_orig	vector	Address originated from msg_types field
client_software	string	Software reported by client in vendor_class
server_software	string	Software reported by server in vendor_class
circuit_id	string	DHCP relay agents that terminate circuits
agent_remote_id	string	Globally unique ID added by relay agents to identify remote host end of circuit
subscriber_id	string	Value independent of physical network connection that provides customer DHCP configuration regardless of physical location

kerberos.log | Kerberos authentication

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > See conn.log
request_type	string	Authentication Service (AS) or Ticket Granting Service (TGS)
client	string	Client
service	string	Service
success	bool	Request result
error_msg	string	Error message
from	time	Ticket valid from
till	time	Ticket valid until
cipher	string	Ticket encryption type
forwardable	bool	Forwardable ticket requested
renewable	bool	Renewable ticket requested
client_cert_subject	string	Subject of client certificate, if any
client_cert_fuid	string	File unique ID of client cert, if any
server_cert_subject	string	Subject of server certificate, if any
server_cert_fuid	string	File unique ID of server cert, if any
auth_ticket	string	Ticket hash authorizing request/transaction
new_ticket	string	Ticket hash returned by KDC

dns.log | DNS query/response details

FIELD	TYPE	DESCRIPTION
ts	time	Earliest timestamp of DNS protocol message
uid & id		Underlying connection info > See conn.log
proto	enum	Transport layer protocol of connection
trans_id	count	16-bit identifier assigned by program that generated DNS query
rtt	interval	Round trip time for query and response
query	string	Domain name subject of DNS query
qclass	count	QCLASS value specifying query class
qclass_name	string	Descriptive name query class
qtype	count	QTYPE value specifying query type
qtype_name	string	Descriptive name for query type
rcode	count	Response code value in DNS response
rcode_name	string	Descriptive name of response code value
AA	bool	Authoritative Answer bit: responding name server is authority for domain name
TC	bool	Truncation bit: message was truncated
RD	bool	Recursion Desired bit: client wants recursive service for query
RA	bool	Recursion Available bit: name server supports recursive queries
Z	count	Reserved field, usually zero in queries and responses
answers	vector	Set of resource descriptions in query answer
TTLs	vector	Caching intervals of RRs in answers field
rejected	bool	DNS query was rejected by server
auth	table	Authoritative responses for query
addl	table	Additional responses for query

dpd.log | Dynamic protocol detection failures

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when protocol analysis failed
uid & id		Underlying connection info > See conn.log
proto	enum	Transport protocol for violation
analyzer	string	Analyzer that generated violation
failure_reason	string	Textual reason for analysis failure
packet_segment	string	Payload chunk that most likely resulted in protocol violation

modbus.log | PLC requests (ICS)

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of the PLC request
uid & id		Underlying connection info > See conn.log

mysql.log | MySQL

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > See conn.log
cmd	string	Command that was issued
arg	string	Argument issued to command
success	bool	Server replied command succeeded
rows	count	Number of affected rows, if any
response	string	Server message, if any

files.log | File analysis results

FIELD	TYPE	DESCRIPTION
ts	time	Time when file first seen
fuid	string	Identifier associated with single file
uid & id		Underlying connection info > See conn.log
source	string	Identification of file data source
depth	count	Value to represent depth of file in relation to source
analyzers	table	Set of analysis types done during file analysis
mime_type	string	Mime type, as determined by Zeek's signatures
filename	string	Filename, if available from file source
duration	interval	Duration file was analyzed for
local_orig	bool	Indicates if data originated from local network
is_orig	bool	If file sent by connection originator or responder
seen_bytes	count	Number of bytes provided to file analysis engine
total_bytes	count	Total number of bytes that should comprise full file
missing_bytes	count	Number of bytes in file stream missed
overflow_bytes	count	Number of bytes in file stream not delivered to stream file analyzers
timedout	bool	If file analysis timed out at least once
parent_fuid	string	Container file ID was extracted from
md5	string	MD5 digest of file contents
sha1	string	SHA1 digest of file contents
sha256	string	SHA256 digest of file contents
extracted	string	Local filename of extracted file
extracted_cutoff	bool	Set to true if file being extracted was cut off so whole file was not logged
extracted_size	count	Number of bytes extracted to disk
entropy	double	Information density of file contents

pe.log | Portable executable

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
id	string	File id of this portable executable file
machine	string	Target machine file was compiled for
compile_ts	time	Time file was created
os	string	Required operating system
subsystem	string	Subsystem required to run this file
is_exe	bool	Is file an executable, or just an object file?
is_64bit	bool	Is file a 64-bit executable?
uses_aslr	bool	Does file support Address Space Layout Randomization?
uses_dep	bool	Does file support Data Execution Prevention?
uses_code_integrity	bool	Does file enforce code integrity checks?
uses_seh	bool	Does file use structured exception handing?
has_import_table	bool	Does file have import table?
has_export_table	bool	Does file have export table?
has_cert_table	bool	Does file have attribute certificate table?
has_debug_data	bool	Does file have debug table?
section_names	vector of string	Names of sections, in order

ftp.log | FTP request/reply details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when command sent
uid & id		Underlying connection info > See conn.log
user	string	Username for current FTP session
password	string	Password for current FTP session
command	string	Command given by client
arg	string	Argument for command, if given
mime_type	string	Sniffed mime type of file
file_size	count	Size of file
reply_code	count	Reply code from server in response to command
reply_msg	string	Reply code from server in response to command
data_channel	record FTP::Expected Data Channel	Expected FTP data channel
fuid	string	File unique ID

http.log | HTTP request/reply details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when request happened
uid & id		Underlying connection info > See conn.log
trans_depth	count	Pipelined depth into connection
method	string	Verb used in HTTP request (GET, POST, etc.)
host	string	Value of HOST header
uri	string	URI used in request
referrer	string	Value of referer header
version	string	Value of version portion of request
user_agent	string	Value of User-Agent header from client
origin	string	Value of Origin header from client
request_body_len	count	Uncompressed data size from client
response_body_len	count	Uncompressed data size from server
status_code	count	Status code returned by server
status_msg	string	Status message returned by server
info_code	count	Last seen 1xx info reply code from server
info_msg	string	Last seen 1xx info reply message from server
tags	table	Indicators of various attributes discovered
username	string	Username if basic-auth performed for request
password	string	Password if basic-auth performed for request
proxied	table	All headers indicative of proxied request
orig_fuids	vector	Ordered vector of file unique IDs
orig_filenames	vector	Ordered vector of filenames from client
orig_mime_types	vector	Ordered vector of mime types
resp_fuids	vector	Ordered vector of file unique IDs
resp_filenames	vector	Ordered vector of filenames from server
resp_mime_types	vector	Ordered vector of mime types

corelight

ZEEK® LOGS

radius.log | RADIUS authentication attempts

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > See conn.log
username	string	Username, if present
mac	string	MAC address, if present
framed_addr	addr	Address given to network access server, if present
tunnel_client	string	Address (IPv4, IPv6, or FQDN) of initiator end of tunnel, if present
connect_info	string	Connect info, if present
reply_msg	string	Reply message from server challenge
result	string	Successful or failed authentication
ttl	interval	Duration between first request and either Access-Accept message or an error

sip.log | SIP analysis

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when request happened
uid & id		Underlying connection info > See conn.log
trans_depth	count	Pipelined depth into request/response transaction
method	string	Verb used in SIP request (INVITE, etc)
uri	string	URI used in request
date	string	Contents of Date: header from client
request_from	string	Contents of request From: header[1]
request_to	string	Contents of To: header
response_from	string	Contents of response From: header[1]
response_to	string	Contents of response To: header
reply_to	string	Contents of Reply-To: header
call_id	string	Contents of Call-ID: header from client
seq	string	Contents of CSeq: header from client
subject	string	Contents of Subject: header from client
request_path	vector	Client message transmission path, extracted from headers
response_path	vector	Server message transmission path, extracted from headers
user_agent	string	Contents of User-Agent: header from client
status_code	count	Status code returned by server
status_msg	string	Status message returned by server
warning	string	Contents of Warning: header
request_body_len	count	Contents of Content-Length: header from client
response_body_len	count	Contents of Content-Length: header from server
content_type	string	Contents of Content-Type: header from server

tunnel.log | Details of encapsulating tunnels

FIELD	TYPE	DESCRIPTION
ts	time	Time at which tunnel activity occurred

smb_mapping.log | SMB mappings

FIELD	TYPE	DESCRIPTION
ts	time	Time when tree was mapped
uid & id		Underlying connection info > See conn.log
path	string	Name of tree path
service	string	Type of resource of tree (disk share, printer share, named pipe, etc)
native_file_system	string	File system of tree
share_type	string	If this is SMB2, share type will be included

smtp.log | SMTP transactions

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when message was first seen
uid & id		Underlying connection info > See conn.log
trans_depth	count	Transaction depth if there are multiple msgs
helo	string	Contents of Helo header
mailfrom	string	Email addresses found in From header
rcptto	table	Email addresses found in Rcpt header
date	string	Contents of Date header
from	string	Contents of From header
to	table	Contents of To header
cc	table	Contents of CC header
reply_to	string	Contents of ReplyTo header
msg_id	string	Contents of MsgID header
in_reply_to	string	Contents of In-Reply-To header
subject	string	Contents of Subject header
x_originating_ip	addr	Contents of X-Originating-IP header
first_received	string	Contents of first Received header
second_received	string	Contents of second Received header
last_reply	string	Last message server sent to client
path	vector	Message transmission path, from headers
user_agent	string	Value of User-Agent header from client
tls	bool	Indicates connection switched to using TLS
fuids	vector	File unique IDs attached to message
is_webmail	bool	If message sent via webmail

snmp.log | SNMP messages

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of first packet of SNMP session
uid & id		Underlying connection info > See conn.log
duration	interval	Amount of time between first packet belonging to SNMP session and latest seen
version	string	Version of SNMP being used
community	string	Community string of first SNMP packet associated with session
get_requests	count	Number of variable bindings in GetRequest/GetNextRequest PDUs seen for session
get_bulk_requests	count	Number of variable bindings in GetBulkRequest PDUs seen for session
get_responses	count	Number of variable bindings in GetResponse/Response PDUs seen for session
set_requests	count	Number of variable bindings in SetRequest PDUs seen for session
display_string	string	System description of SNMP responder endpoint
up_since	time	Time at which SNMP responder endpoint claims it's been up since

socks.log | SOCKS proxy requests

FIELD	TYPE	DESCRIPTION
ts	time	Time when proxy connection detected
uid & id		Underlying connection info > See conn.log
version	count	Protocol version of SOCKS
user	string	Username used to request a login to proxy
password	string	Password used to request a login to proxy
status	string	Server status for attempt at using proxy
request	record SOCKS::Address	Client requested SOCKS address
request_p	port	Client requested port
bound	record SOCKS::Address	Server bound address
bound_p	port	Server bound port

software.log | Software observed on network

FIELD	TYPE	DESCRIPTION
ts	time	Time at which software was detected
host	addr	IP address detected running the software
host_p	port	Port on which software is running
software_type	enum	Type of software detected (e.g., HTTP::SERVER)
name	string	Name of software (e.g., Apache)
version	record Software::Version	Software version
unparsed_version	string	Full, unparsed version string found
url	string	Root URL where software was discovered

ssh.log | SSH handshakes

FIELD	TYPE	DESCRIPTION
ts	time	Time when SSH connection began
uid & id		Underlying connection info > See conn.log
version	count	SSH major version (1 or 2)
auth_success	bool	Authentication result (T=success, F=failure, unset=unknown)
auth_attempts	count	Number of authentication attempts observed
direction	enum	Direction of connection
client	string	Client's version string
server	string	Server's version string
cipher_alg	string	Encryption algorithm in use
mac_alg	string	Signing (MAC) algorithm in use
compression_alg	string	Compression algorithm in use
kex_alg	string	Key exchange algorithm in use
host_key_alg	string	Server host key's algorithm
host_key	string	Server's key fingerprint
remote_location	record geo_location	Add geographic data related to remote host of connection

dnp3.log | Distributed Network Protocol (ICS)

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of the DNP3 request
uid & id		Underlying connection info > See conn.log

ssl.log | SSL handshakes

FIELD	TYPE	DESCRIPTION
ts	time	Time when SSL connection first detected
uid & id		Underlying connection info > See conn.log
version	string	SSL/TLS version server chose
cipher	string	SSL/TLS cipher suite server chose
curve	string	Elliptic curve server chose when using ECDH/ECDHE
server_name	string	Value of Server Name Indicator SSL/TLS extension
resumed	bool	Flag that indicates session was resumed
last_alert	string	Last alert seen during connection
next_protocol	string	Next protocol server chose using application layer next protocol extension, if present
established	bool	Flags if SSL session successfully established
ssl_history	string	SSL history showing which types of packets were received in which order. Client-side letters are capitalized, server-side lower-case
cert_chain_fps	vector	All fingerprints for the certificates offered by the server
client_cert_chain_fps	vector	All fingerprints for the certificates offered by the client
subject	string	Subject of X.509 cert offered by server
issuer	string	Subject of signer of X.509 server cert
client_subject	string	Subject of X.509 cert offered by client
client_issuer	string	Subject of signer of client cert
sni_matches_cert	bool	Set to true if the hostname sent in the SNI matches the certificate, false if they do not. Unset if the client did not send an SNI
request_client_certificate_authorities	vector	List of client certificate CAs accepted by the server
server_version	count	Numeric version of the server in the server hello
client_version	count	Numeric version of the client in the client hello
client_ciphers	vector	Ciphers that were offered by the client for the connection
ssl_client_exts	vector	SSL client extensions
ssl_server_exts	vector	SSL server extensions
ticket_lifetime_hint	count	Suggested ticket lifetime sent in the session ticket handshake by the server
dh_param_size	count	The diffie helman parameter size, when using DH
point_formats	vector	Supported elliptic curve point formats
client_curves	vector	The curves supported by the client
orig_alpn	vector	Application layer protocol negotiation extension sent by the client
client_supported_versions	vector	TLS 1.3 supported versions
server_supported_version	count	TLS 1.3 supported version
psk_key_exchange_modes	vector	TLS 1.3 Pre-shared key exchange modes
client_key_share_groups	vector	Key share groups from client hello
server_key_share_group	count	Selected key share group from server hello

syslog.log | Syslog messages

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when syslog message was seen
uid & id		Underlying connection info > See conn.log
proto	enum	Protocol over which message was seen
facility	string	Syslog facility for message
severity	string	Syslog severity for message
message	string	Plain text message

smb_files.log | Details on SMB files

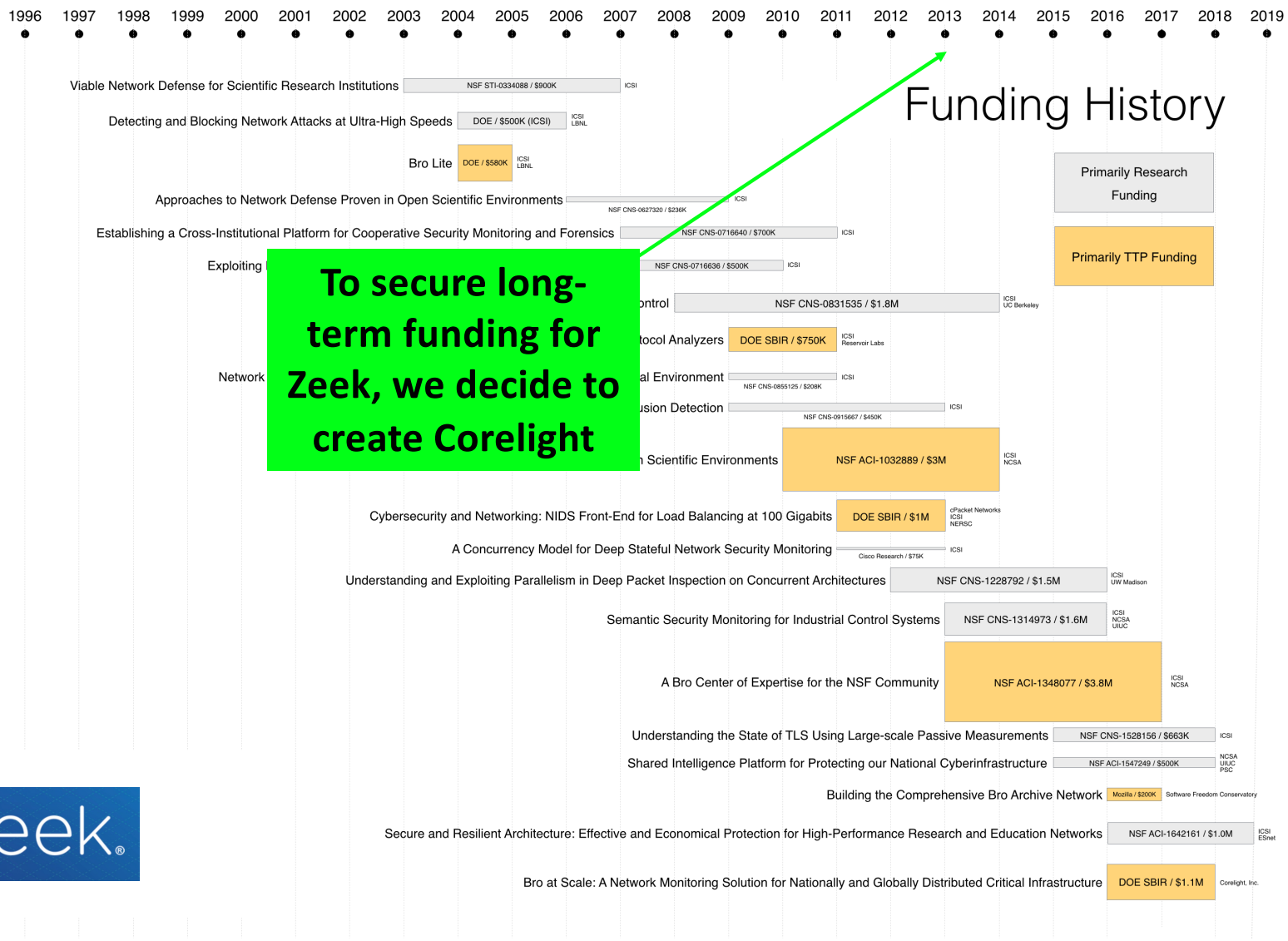
FIELD	TYPE	DESCRIPTION
ts	time	Time when file was first discovered
uid & id		Underlying connection info > See conn.log
fuid	string	Unique ID of file
action	enum	Action this log record represents
path	string	Path pulled from tree that file was transferred to or from
name	string	Filename if one was seen
size	count	Total size of file
prev_name	string	If rename action was seen, this will be file's previous name
times	record SMB::MAC-Times	Last time file was modified

weird.log | Unexpected network/protocol

FIELD	TYPE	DESCRIPTION
ts	time	Time when weird occurred
uid & id		Underlying connection info > See conn.log
name	string	Name of weird that occurred
addl	string	Additional information accompanying weird, if any
notice	bool	If weird was turned into a notice
peer	string	Peer that originated weird

x509.log | X.509 certificate info

FIELD	TYPE	DESCRIPTION
ts	time	Current timestamp
fingerprint	string	Fingerprint of the certificate
certificate	record X509::Certificate	Basic information about certificate
san	record X509::Subject Alternative Name	Subject alternative name extension of certificate
basic_constraints	record X509::Basic Constraints	Basic constraints extension of certificate


1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Funding History
Primarily Research Funding
Primarily TTP Funding
To secure long-term funding for Zeek, we decide to create Corelight
Viable Network Defense for Scientific Research Institutions
NSF STI-0334088 / $900K
ICSI
Detecting and Blocking Network Attacks at Ultra-High Speeds
DOE / $500K (ICSI)
ICSI
LBNL
Bro Lite
DOE / $580K
ICSI
LBNL
Approaches to Network Defense Proven in Open Scientific Environments
NSF CNS-0627320 / $236K
ICSI
Establishing a Cross-Institutional Platform for Cooperative Security Monitoring and Forensics
NSF CNS-0716640 / $700K
ICSI
Exploiting
NSF CNS-0716636 / $500K
ICSI
ontrol
NSF CNS-0831535 / $1.8M
ICSI
UC Berkeley
tocol Analyzers
DOE SBIR / $750K
ICSI
Reservoir Labs
Network
al Environment
NSF CNS-0855125 / $208K
ICSI
usion Detection
NSF CNS-0915667 / $450K
ICSI
Scientific Environments
NSF ACI-1032889 / $3M
ICSI
NCSA
Cybersecurity and Networking: NIDS Front-End for Load Balancing at 100 Gigabits
DOE SBIR / $1M
cPacket Networks
ICSI
NERSC
A Concurrency Model for Deep Stateful Network Security Monitoring
Cisco Research / $75K
ICSI
Understanding and Exploiting Parallelism in Deep Packet Inspection on Concurrent Architectures
NSF CNS-1228792 / $1.5M
ICSI
UW Madison
Semantic Security Monitoring for Industrial Control Systems
NSF CNS-1314973 / $1.6M
ICSI
NCSA
UIUC
A Bro Center of Expertise for the NSF Community
NSF ACI-1348077 / $3.8M
ICSI
NCSA
Understanding the State of TLS Using Large-scale Passive Measurements
NSF CNS-1528156 / $663K
ICSI
Shared Intelligence Platform for Protecting our National Cyberinfrastructure
NSF ACI-1547249 / $500K
NCSA
UIUC
PSC
Building the Comprehensive Bro Archive Network
Mozilla / $200K
Software Freedom Conservatory
Secure and Resilient Architecture: Effective and Economical Protection for High-Performance Research and Education Networks
NSF ACI-1642161 / $1.0M
ICSI
ESnet
Bro at Scale: A Network Monitoring Solution for Nationally and Globally Distributed Critical Infrastructure
DOE SBIR / $1.1M
Corelight, Inc.

zeek®

Enterprises & Replication of Previous Results: Keynote Edition

ACM Internet Measurement Conference

IMC 2022

Inside-out and Backwards: A Retrospective Look at How Measurement Research *Really* Happens

Stefan Savage
University of California, San Diego

Reflecting on all this… in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us

Business plan: a network analysis "app store"

Reflecting on all this… in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us

Paying the bills via professional services ⇒ Aha!

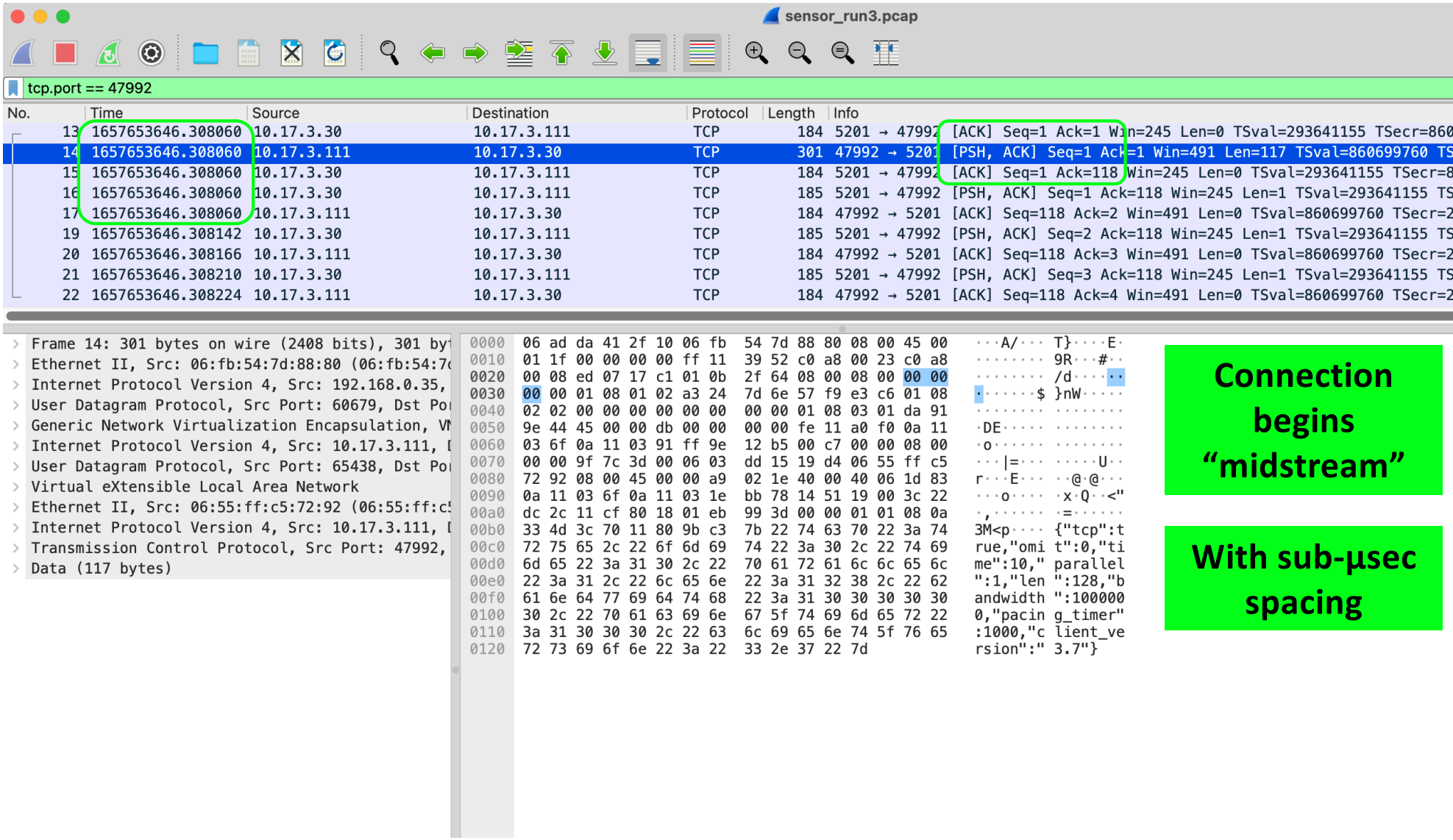
Reflecting on all this… in the context of founding a startup

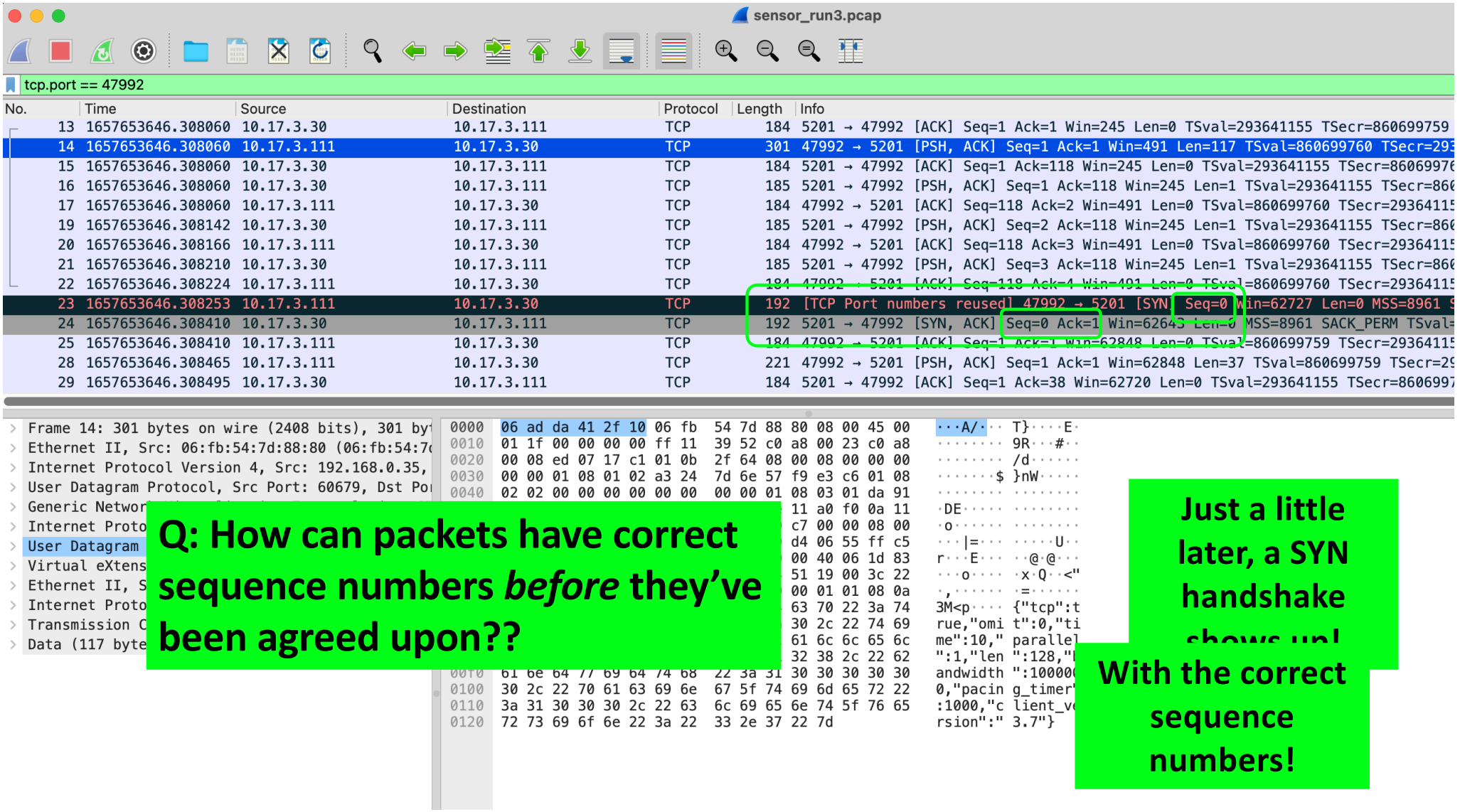
- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us

Nature can be so cruel …

Measurement *Glitch* or …
… Baffling **Reality**?

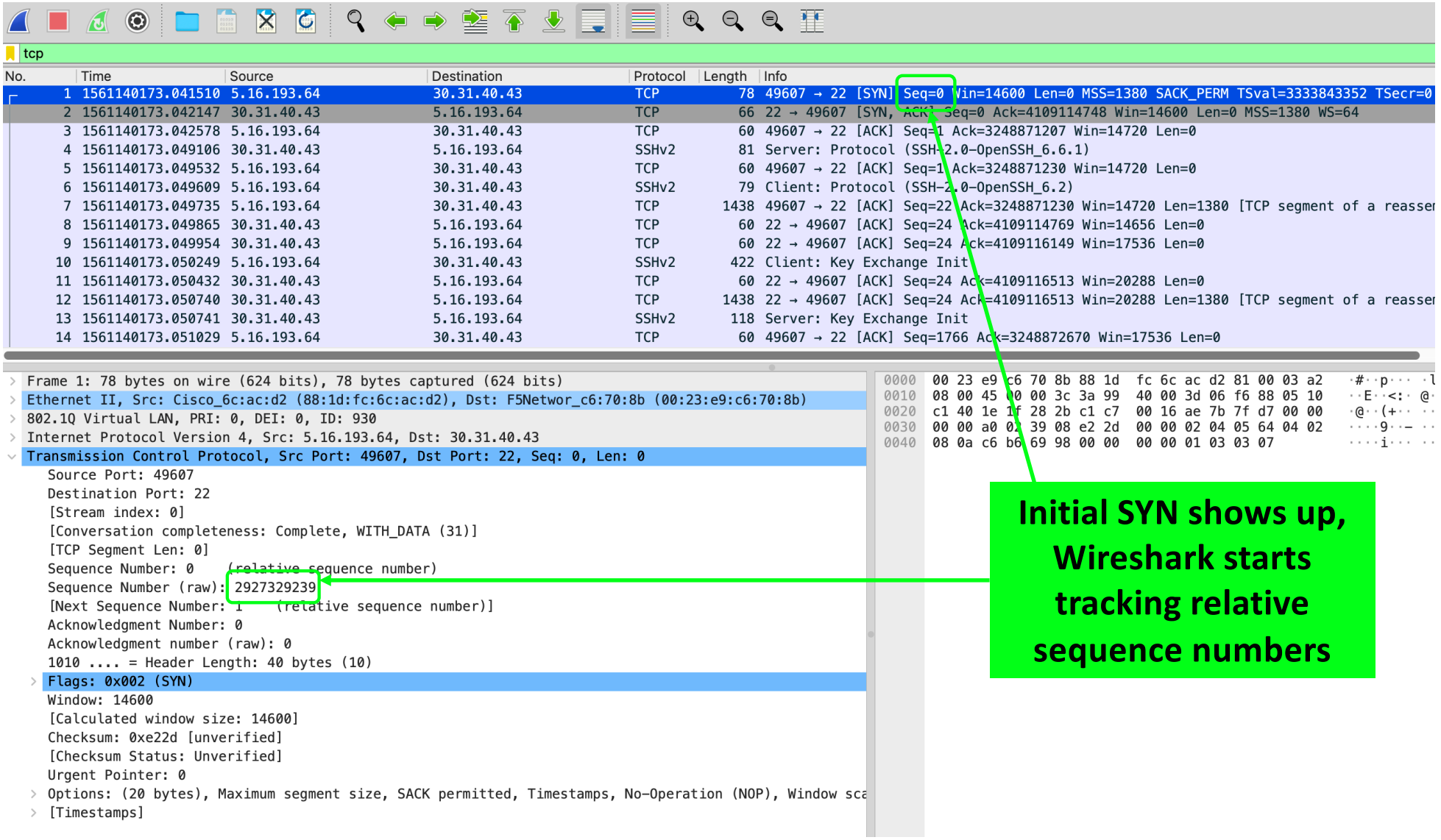
#1

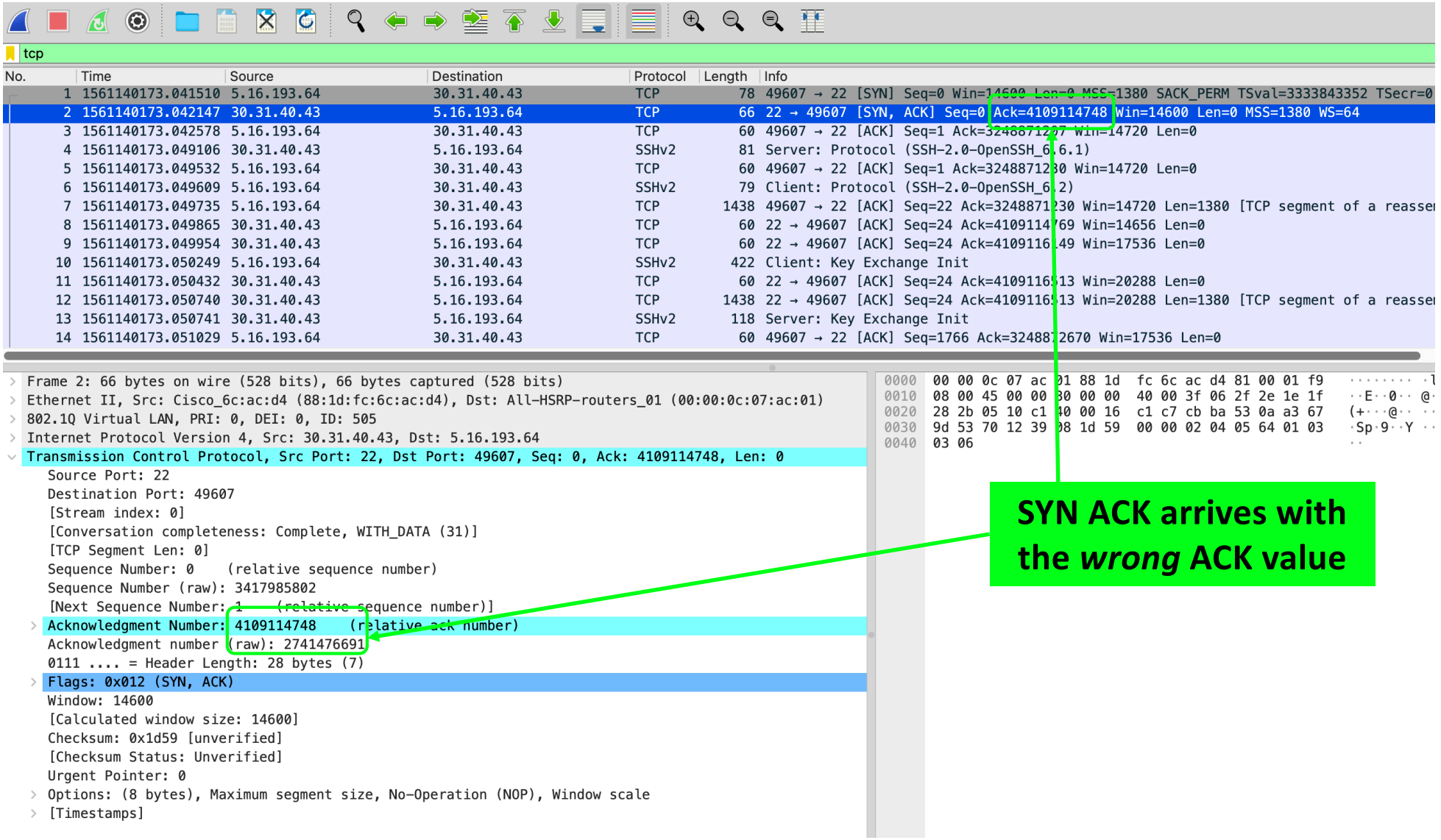

sensor_run3.pcap
tcp.port == 47992
No. Time Source Destination Protocol Length Info
13 1657653646.308060 10.17.3.30 10.17.3.111 TCP 184 5201 → 47992 [ACK] Seq=1 Ack=1 Win=245 Len=0 TSval=293641155 TSecr=860699759
14 1657653646.308060 10.17.3.111 10.17.3.30 TCP 301 47992 → 5201 [PSH, ACK] Seq=1 Ack=1 Win=491 Len=117 TSval=860699760 TSecr=29
15 1657653646.308060 10.17.3.30 10.17.3.111 TCP 184 5201 → 47992 [ACK] Seq=1 Ack=118 Win=245 Len=0 TSval=293641155 TSecr=8606997
16 1657653646.308060 10.17.3.30 10.17.3.111 TCP 185 5201 → 47992 [PSH, ACK] Seq=1 Ack=118 Win=245 Len=1 TSval=293641155 TSecr=86
17 1657653646.308060 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=118 Ack=2 Win=491 Len=0 TSval=860699760 TSecr=2936411
19 1657653646.308142 10.17.3.30 10.17.3.111 TCP 185 5201 → 47992 [PSH, ACK] Seq=2 Ack=118 Win=245 Len=1 TSval=293641155 TSecr=86
20 1657653646.308166 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=118 Ack=3 Win=491 Len=0 TSval=860699760 TSecr=2936411
21 1657653646.308210 10.17.3.30 10.17.3.111 TCP 185 5201 → 47992 [PSH, ACK] Seq=3 Ack=118 Win=245 Len=1 TSval=293641155 TSecr=86
22 1657653646.308224 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=118 Ack=4 Win=491 Len=0 TSval=860699760 TSecr=2936411
Frame 14: 301 bytes on wire (2408 bits), 301 by
Ethernet II, Src: 06:fb:54:7d:88:80 (06:fb:54:7
Internet Protocol Version 4, Src: 192.168.0.35,
User Datagram Protocol, Src Port: 60679, Dst Po
Generic Network Virtualization Encapsulation, V
Internet Protocol Version 4, Src: 10.17.3.111,
User Datagram Protocol, Src Port: 65438, Dst Po
Virtual eXtensible Local Area Network
Ethernet II, Src: 06:55:ff:c5:72:92 (06:55:ff:c
Internet Protocol Version 4, Src: 10.17.3.111,
Transmission Control Protocol, Src Port: 47992,
Data (117 bytes)
0000 06 ad da 41 2f 10 06 fb 54 7d 88 80 08 00 45 00 ···A/··· T}····E·
0010 01 1f 00 00 00 00 ff 11 39 52 c0 a8 00 23 c0 a8 ········ 9R···#··
0020 00 08 ed 07 17 c1 01 0b 2f 64 08 00 08 00 00 00 ········ /d······
0030 00 00 01 08 01 02 a3 24 7d 6e 57 f9 e3 c6 01 08 ······$ }nW·····
0040 02 02 00 00 00 00 00 00 00 00 01 08 03 01 da 91 ········ ········
0050 9e 44 45 00 00 db 00 00 00 00 fe 11 a0 f0 0a 11 ·DE····· ········
0060 03 6f 0a 11 03 91 ff 9e 12 b5 00 c7 00 00 08 00 ·o····· ········
0070 00 00 9f 7c 3d 00 06 03 dd 15 19 d4 06 55 ff c5 ···|=··· ·····U··
0080 72 92 08 00 45 00 00 a9 02 1e 40 00 40 06 1d 83 r···E··· ··@·@···
0090 0a 11 03 6f 0a 11 03 1e bb 78 14 51 19 00 3c 22 ···o···· ·x·Q··<"
00a0 dc 2c 11 cf 80 18 01 eb 99 3d 00 00 01 01 08 0a ·,····· ·=·····
00b0 33 4d 3c 70 11 80 9b c3 7b 22 74 63 70 22 3a 74 3M<p···· {"tcp":t
00c0 72 75 65 2c 22 6f 6d 69 74 22 3a 30 2c 22 74 69 rue,"omi t":0,"ti
00d0 6d 65 22 3a 31 30 2c 22 70 61 72 61 6c 6c 65 6c me":10," parallel
00e0 22 3a 31 2c 22 6c 65 6e 22 3a 31 32 38 2c 22 62 ":1,"len ":128,"b
00f0 61 6e 64 77 69 64 74 68 22 3a 31 30 30 30 30 30 andwidth ":100000
0100 30 2c 22 70 61 63 69 6e 67 5f 74 69 6d 65 72 22 0,"pacin g_timer"
0110 3a 31 30 30 30 2c 22 63 6c 69 65 6e 74 5f 76 65 :1000,"c lient_ve
0120 72 73 69 6f 6e 22 3a 22 33 2e 37 22 7d rsion":" 3.7"}
Connection begins "midstream"
With sub-µsec spacing

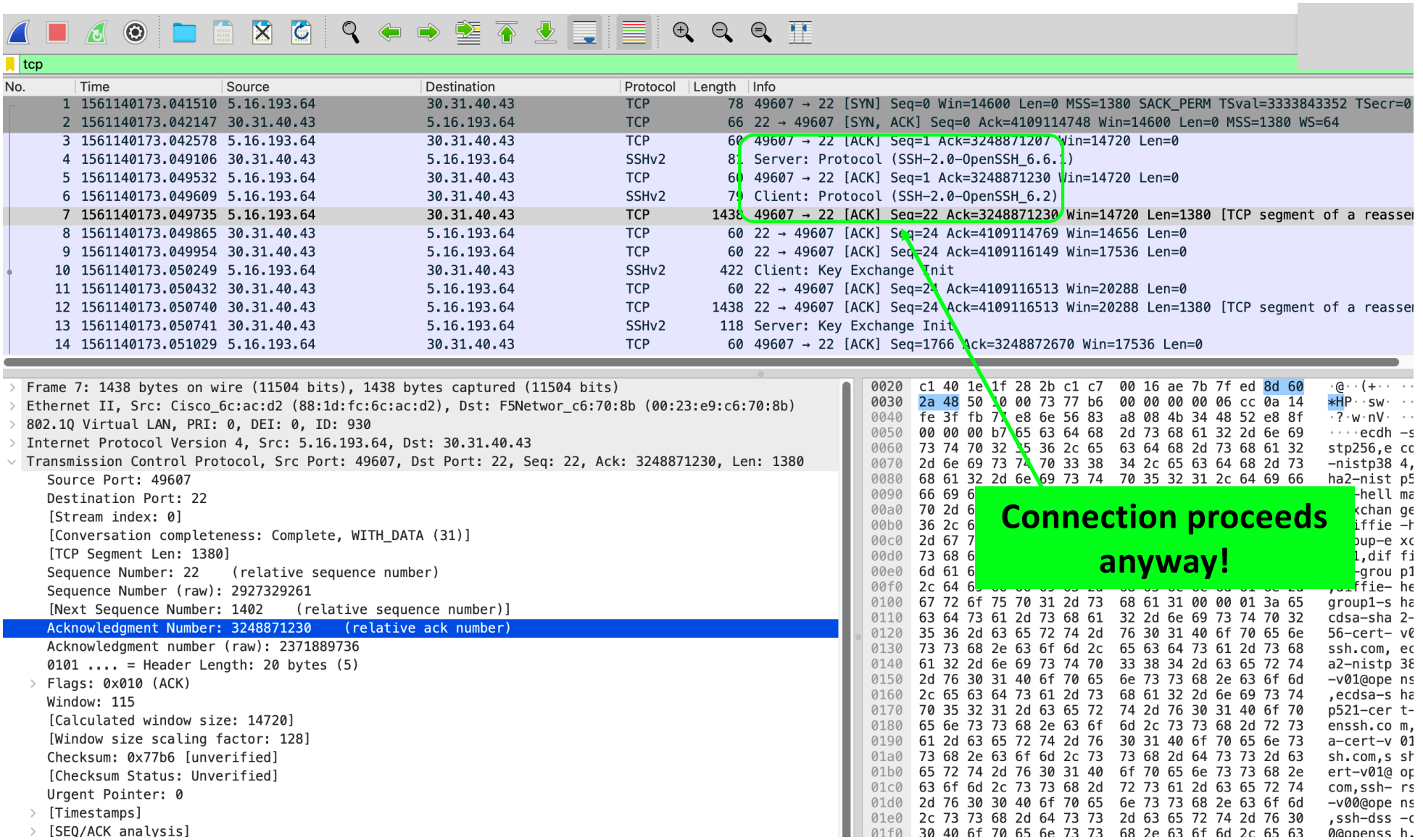

sensor_run3.pcap
tcp.port == 47992
No. Time Source Destination Protocol Length Info
13 1657653646.308060 10.17.3.30 10.17.3.111 TCP 184 5201 → 47992 [ACK] Seq=1 Ack=1 Win=245 Len=0 TSval=293641155 TSecr=860699759
14 1657653646.308060 10.17.3.111 10.17.3.30 TCP 301 47992 → 5201 [PSH, ACK] Seq=1 Ack=1 Win=491 Len=117 TSval=860699760 TSecr=293
15 1657653646.308060 10.17.3.30 10.17.3.111 TCP 184 5201 → 47992 [ACK] Seq=1 Ack=118 Win=245 Len=0 TSval=293641155 TSecr=8606997
16 1657653646.308060 10.17.3.30 10.17.3.111 TCP 185 5201 → 47992 [PSH, ACK] Seq=1 Ack=118 Win=245 Len=1 TSval=293641155 TSecr=860
17 1657653646.308060 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=118 Ack=2 Win=491 Len=0 TSval=860699760 TSecr=2936411
19 1657653646.308142 10.17.3.30 10.17.3.111 TCP 185 5201 → 47992 [PSH, ACK] Seq=2 Ack=118 Win=245 Len=1 TSval=293641155 TSecr=860
20 1657653646.308166 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=118 Ack=3 Win=491 Len=0 TSval=860699760 TSecr=2936411
21 1657653646.308210 10.17.3.30 10.17.3.111 TCP 185 5201 → 47992 [PSH, ACK] Seq=3 Ack=118 Win=245 Len=1 TSval=293641155 TSecr=860
22 1657653646.308224 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=118 Ack=4 Win=491 Len=0 TSval=860699760 TSecr=2936411
23 1657653646.308253 10.17.3.111 10.17.3.30 TCP 192 [TCP Port numbers reused] 47992 → 5201 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 S
24 1657653646.308410 10.17.3.30 10.17.3.111 TCP 192 5201 → 47992 [SYN, ACK] Seq=0 Ack=1 Win=62643 Len=0 MSS=8961 SACK_PERM TSval=
25 1657653646.308410 10.17.3.111 10.17.3.30 TCP 184 47992 → 5201 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=860699759 TSecr=2936411
28 1657653646.308465 10.17.3.111 10.17.3.30 TCP 221 47992 → 5201 [PSH, ACK] Seq=1 Ack=1 Win=62848 Len=37 TSval=860699759 TSecr=29
29 1657653646.308495 10.17.3.30 10.17.3.111 TCP 184 5201 → 47992 [ACK] Seq=1 Ack=38 Win=62720 Len=0 TSval=293641155 TSecr=8606997
Frame 14: 301 bytes on wire (2408 bits), 301 byt
Ethernet II, Src: 06:fb:54:7d:88:80 (06:fb:54:7
Internet Protocol Version 4, Src: 192.168.0.35,
User Datagram Protocol, Src Port: 60679, Dst Po
Generic Networ
Internet Proto
User Datagram
Virtual eXtens
Ethernet II, S
Internet Proto
Transmission C
Data (117 byte
0000 06 ad da 41 2f 10 06 fb 54 7d 88 80 08 00 45 00 ···A/··· T}····E·
0010 01 1f 00 00 00 00 ff 11 39 52 c0 a8 00 23 c0 a8 ········ 9R···#··
0020 00 08 ed 07 17 c1 01 0b 2f 64 08 00 08 00 00 00 ········ /d······
0030 00 00 01 08 01 02 a3 24 7d 6e 57 f9 e3 c6 01 08 ·······$ }nW·····
0040 02 02 00 00 00 00 00 00 00 00 01 08 03 01 da 91 ········ ········
11 a0 f0 0a 11 ·DE····· ········
c7 00 00 08 00 ·o····· ········
d4 06 55 ff c5 ···|=··· ····U··
00 40 06 1d 83 r···E··· ··@·@···
51 19 00 3c 22 ···o···· ·x·Q··<"
00 01 01 08 0a ·,····· ·=·····
63 70 22 3a 74 3M<p···· {"tcp":t
30 2c 22 74 69 rue,"omi t":0,"ti
61 6c 6c 65 6c me":10," parallel
32 38 2c 22 62 ":1,"len ":128,"
61 6e 64 77 69 64 74 68 22 3a 31 30 30 30 30 30 andwidth ":10000
0100 30 2c 22 70 61 63 69 6e 67 5f 74 69 6d 65 72 22 0,"pacin g_timer"
0110 3a 31 30 30 30 2c 22 63 6c 69 65 6e 74 5f 76 65 :1000,"c lient_ve
0120 72 73 69 6f 6e 22 3a 22 33 2e 37 22 7d rsion":" 3.7"}
Q: How can packets have correct sequence numbers before they've been agreed upon??
Just a little later, a SYN handshake shows up!
With the correct sequence numbers!

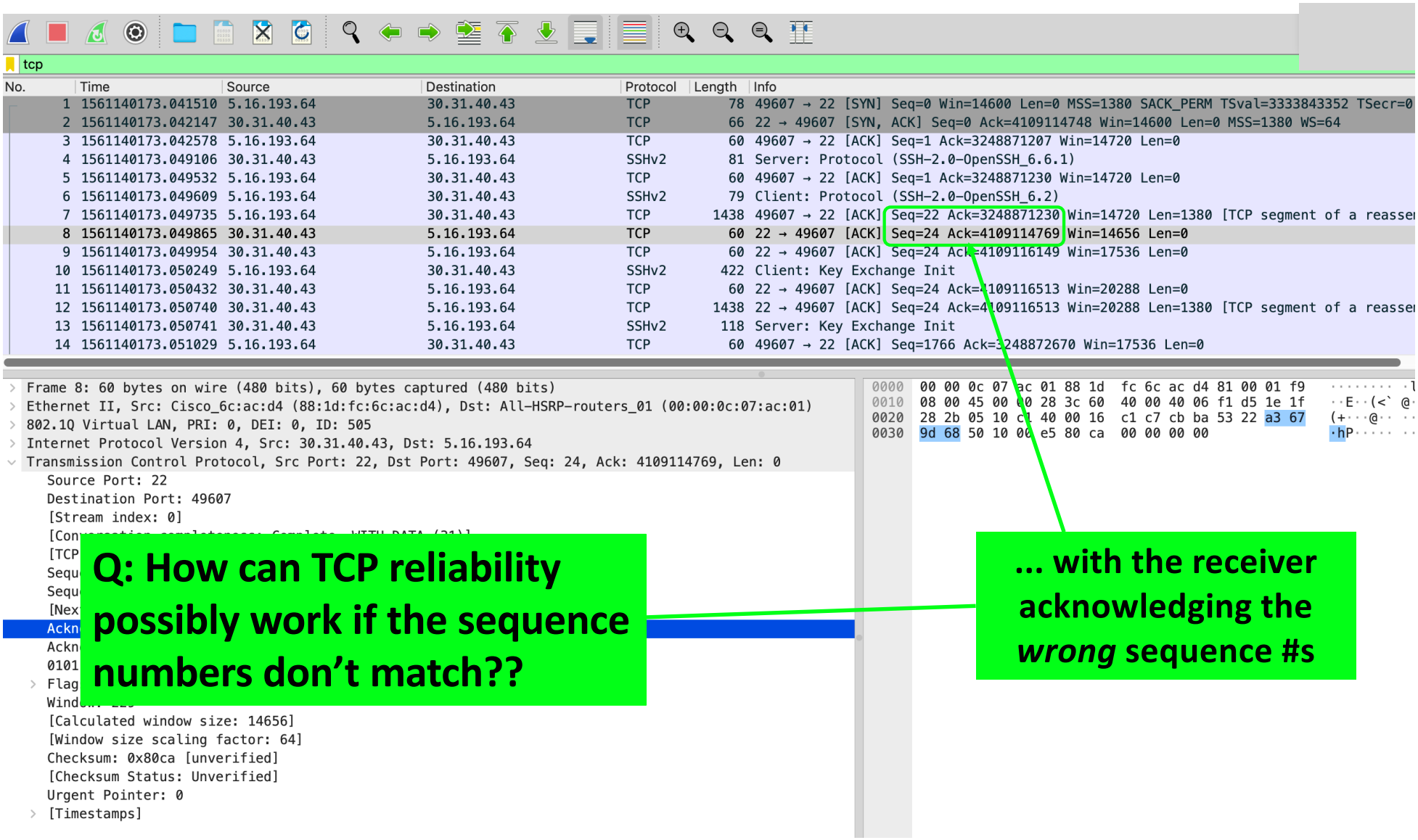
Measurement *Glitch* or …
… Baffling **Reality**?

#2

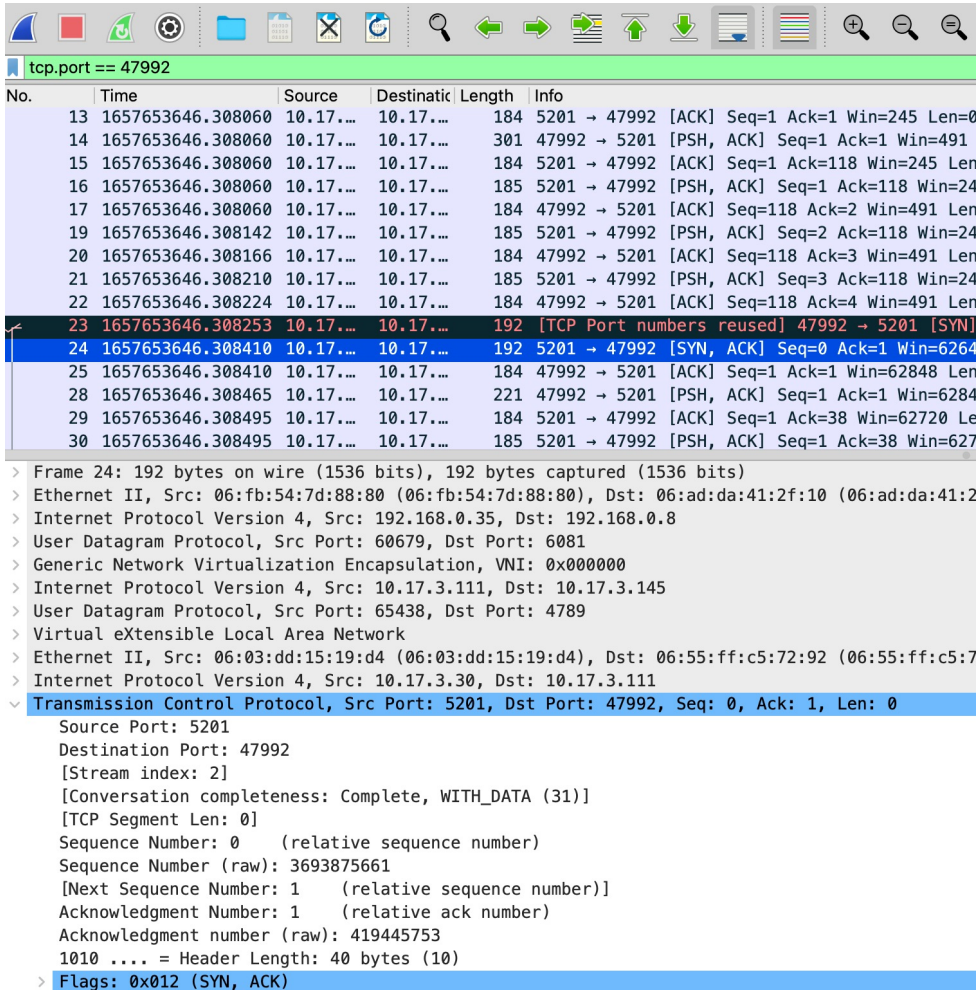

tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
2 1561140173.042147 30.31.40.43 5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
3 1561140173.042578 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14720 Len=0
4 1561140173.049106 30.31.40.43 5.16.193.64 SSHv2 81 Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
5 1561140173.049532 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14720 Len=0
6 1561140173.049609 5.16.193.64 30.31.40.43 SSHv2 79 Client: Protocol (SSH-2.0-OpenSSH_6.2)
7 1561140173.049735 5.16.193.64 30.31.40.43 TCP 1438 49607 → 22 [ACK] Seq=22 Ack=3248871230 Win=14720 Len=1380 [TCP segment of a reasse
8 1561140173.049865 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109114769 Win=14656 Len=0
9 1561140173.049954 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116149 Win=17536 Len=0
10 1561140173.050249 5.16.193.64 30.31.40.43 SSHv2 422 Client: Key Exchange Init
11 1561140173.050432 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=0
12 1561140173.050740 30.31.40.43 5.16.193.64 TCP 1438 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
13 1561140173.050741 30.31.40.43 5.16.193.64 SSHv2 118 Server: Key Exchange Init
14 1561140173.051029 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win=17536 Len=0
Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: Cisco_6c:ac:d2 (88:1d:fc:6c:ac:d2), Dst: F5Networ_c6:70:8b (00:23:e9:c6:70:8b)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 930
Internet Protocol Version 4, Src: 5.16.193.64, Dst: 30.31.40.43
Transmission Control Protocol, Src Port: 49607, Dst Port: 22, Seq: 0, Len: 0
Source Port: 49607
Destination Port: 22
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2927329239
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1010 .... = Header Length: 40 bytes (10)
Flags: 0x002 (SYN)
Window: 14600
[Calculated window size: 14600]
Checksum: 0xe22d [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window sca
[Timestamps]
0000 00 23 e9 c6 70 8b 88 1d fc 6c ac d2 81 00 03 a2
0010 08 00 45 00 00 3c 3a 99 40 00 3d 06 f6 88 05 10
0020 c1 40 1e 1f 28 2b c1 c7 00 16 ae 7b 7f d7 00 00
0030 00 00 a0 02 39 08 e2 2d 00 00 02 04 05 64 04 02
0040 08 0a c6 b6 69 98 00 00 00 00 01 03 03 07
Initial SYN shows up, Wireshark starts tracking relative sequence numbers


tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
2 1561140173.042147 30.31.40.43 5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
3 1561140173.042578 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14720 Len=0
4 1561140173.049106 30.31.40.43 5.16.193.64 SSHv2 81 Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
5 1561140173.049532 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14720 Len=0
6 1561140173.049609 5.16.193.64 30.31.40.43 SSHv2 79 Client: Protocol (SSH-2.0-OpenSSH_6.2)
7 1561140173.049735 5.16.193.64 30.31.40.43 TCP 1438 49607 → 22 [ACK] Seq=22 Ack=3248871230 Win=14720 Len=1380 [TCP segment of a reasse
8 1561140173.049865 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109114769 Win=14656 Len=0
9 1561140173.049954 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116149 Win=17536 Len=0
10 1561140173.050249 5.16.193.64 30.31.40.43 SSHv2 422 Client: Key Exchange Init
11 1561140173.050432 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=0
12 1561140173.050740 30.31.40.43 5.16.193.64 TCP 1438 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
13 1561140173.050741 30.31.40.43 5.16.193.64 SSHv2 118 Server: Key Exchange Init
14 1561140173.051029 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win=17536 Len=0
Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Cisco_6c:ac:d4 (88:1d:fc:6c:ac:d4), Dst: All-HSRP-routers_01 (00:00:0c:07:ac:01)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 505
Internet Protocol Version 4, Src: 30.31.40.43, Dst: 5.16.193.64
Transmission Control Protocol, Src Port: 22, Dst Port: 49607, Seq: 0, Ack: 4109114748, Len: 0
Source Port: 22
Destination Port: 49607
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3417985802
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 4109114748 (relative ack number)
Acknowledgment number (raw): 2741476691
0111 .... = Header Length: 28 bytes (7)
Flags: 0x012 (SYN, ACK)
Window: 14600
[Calculated window size: 14600]
Checksum: 0x1d59 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (8 bytes), Maximum segment size, No-Operation (NOP), Window scale
[Timestamps]
0000 00 00 0c 07 ac 01 88 1d fc 6c ac d4 81 00 01 f9
0010 08 00 45 00 00 30 00 00 40 00 3f 06 2f 2e 1e 1f
0020 28 2b 05 10 c1 40 00 16 c1 c7 cb ba 53 0a a3 67
0030 9d 53 70 12 39 08 1d 59 00 00 02 04 05 64 01 03
0040 03 06
SYN ACK arrives with the wrong ACK value


tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
2 1561140173.042147 30.31.40.43 5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
3 1561140173.042578 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14720 Len=0
4 1561140173.049106 30.31.40.43 5.16.193.64 SSHv2 81 Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
5 1561140173.049532 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14720 Len=0
6 1561140173.049609 5.16.193.64 30.31.40.43 SSHv2 79 Client: Protocol (SSH-2.0-OpenSSH_6.2)
7 1561140173.049735 5.16.193.64 30.31.40.43 TCP 1438 49607 → 22 [ACK] Seq=22 Ack=3248871230 Win=14720 Len=1380 [TCP segment of a reasse
8 1561140173.049865 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109114769 Win=14656 Len=0
9 1561140173.049954 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116149 Win=17536 Len=0
10 1561140173.050249 5.16.193.64 30.31.40.43 SSHv2 422 Client: Key Exchange Init
11 1561140173.050432 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=0
12 1561140173.050740 30.31.40.43 5.16.193.64 TCP 1438 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
13 1561140173.050741 30.31.40.43 5.16.193.64 SSHv2 118 Server: Key Exchange Init
14 1561140173.051029 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win=17536 Len=0
Frame 7: 1438 bytes on wire (11504 bits), 1438 bytes captured (11504 bits)
Ethernet II, Src: Cisco_6c:ac:d2 (88:1d:fc:6c:ac:d2), Dst: F5Networ_c6:70:8b (00:23:e9:c6:70:8b)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 930
Internet Protocol Version 4, Src: 5.16.193.64, Dst: 30.31.40.43
Transmission Control Protocol, Src Port: 49607, Dst Port: 22, Seq: 22, Ack: 3248871230, Len: 1380
Source Port: 49607
Destination Port: 22
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 1380]
Sequence Number: 22 (relative sequence number)
Sequence Number (raw): 2927329261
[Next Sequence Number: 1402 (relative sequence number)]
Acknowledgment Number: 3248871230 (relative ack number)
Acknowledgment number (raw): 2371889736
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 115
[Calculated window size: 14720]
[Window size scaling factor: 128]
Checksum: 0x77b6 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]
0020 c1 40 1e 1f 28 2b c1 c7 00 16 ae 7b 7f ed 8d 60
0030 2a 48 50 10 00 73 77 b6 00 00 00 00 06 cc 0a 14
0040 fe 3f fb 77 e8 6e 56 83 a8 08 4b 34 48 52 e8 8f
0050 00 00 00 b7 65 63 64 68 2d 73 68 61 32 2d 6e 69
0060 73 74 70 32 35 36 2c 65 63 64 68 2d 73 68 61 32
0070 2d 6e 69 73 74 70 33 38 34 2c 65 63 64 68 2d 73
0080 68 61 32 2d 6e 69 73 74 70 35 32 31 2c 64 69 66
Connection proceeds anyway!


tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
2 1561140173.042147 30.31.40.43 5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
3 1561140173.042578 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14720 Len=0
4 1561140173.049106 30.31.40.43 5.16.193.64 SSHv2 81 Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
5 1561140173.049532 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14720 Len=0
6 1561140173.049609 5.16.193.64 30.31.40.43 SSHv2 79 Client: Protocol (SSH-2.0-OpenSSH_6.2)
7 1561140173.049735 5.16.193.64 30.31.40.43 TCP 1438 49607 → 22 [ACK] Seq=22 Ack=3248871230 Win=14720 Len=1380 [TCP segment of a reasse
8 1561140173.049865 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109114769 Win=14656 Len=0
9 1561140173.049954 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116149 Win=17536 Len=0
10 1561140173.050249 5.16.193.64 30.31.40.43 SSHv2 422 Client: Key Exchange Init
11 1561140173.050432 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=0
12 1561140173.050740 30.31.40.43 5.16.193.64 TCP 1438 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
13 1561140173.050741 30.31.40.43 5.16.193.64 SSHv2 118 Server: Key Exchange Init
14 1561140173.051029 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win=17536 Len=0
Frame 8: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: Cisco_6c:ac:d4 (88:1d:fc:6c:ac:d4), Dst: All-HSRP-routers_01 (00:00:0c:07:ac:01)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 505
Internet Protocol Version 4, Src: 30.31.40.43, Dst: 5.16.193.64
Transmission Control Protocol, Src Port: 22, Dst Port: 49607, Seq: 24, Ack: 4109114769, Len: 0
Source Port: 22
Destination Port: 49607
[Stream index: 0]
[Calculated window size: 14656]
[Window size scaling factor: 64]
Checksum: 0x80ca [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
0000 00 00 0c 07 ac 01 88 1d fc 6c ac d4 81 00 01 f9
0010 08 00 45 00 00 28 3c 60 40 00 40 06 f1 d5 1e 1f
0020 28 2b 05 10 c1 40 00 16 c1 c7 cb ba 53 22 a3 67
0030 9d 68 50 10 00 e5 80 ca 00 00 00 00
Q: How can TCP reliability possibly work if the sequence numbers don't match??
... with the receiver acknowledging the wrong sequence #s

Glitch … or **Reality**?

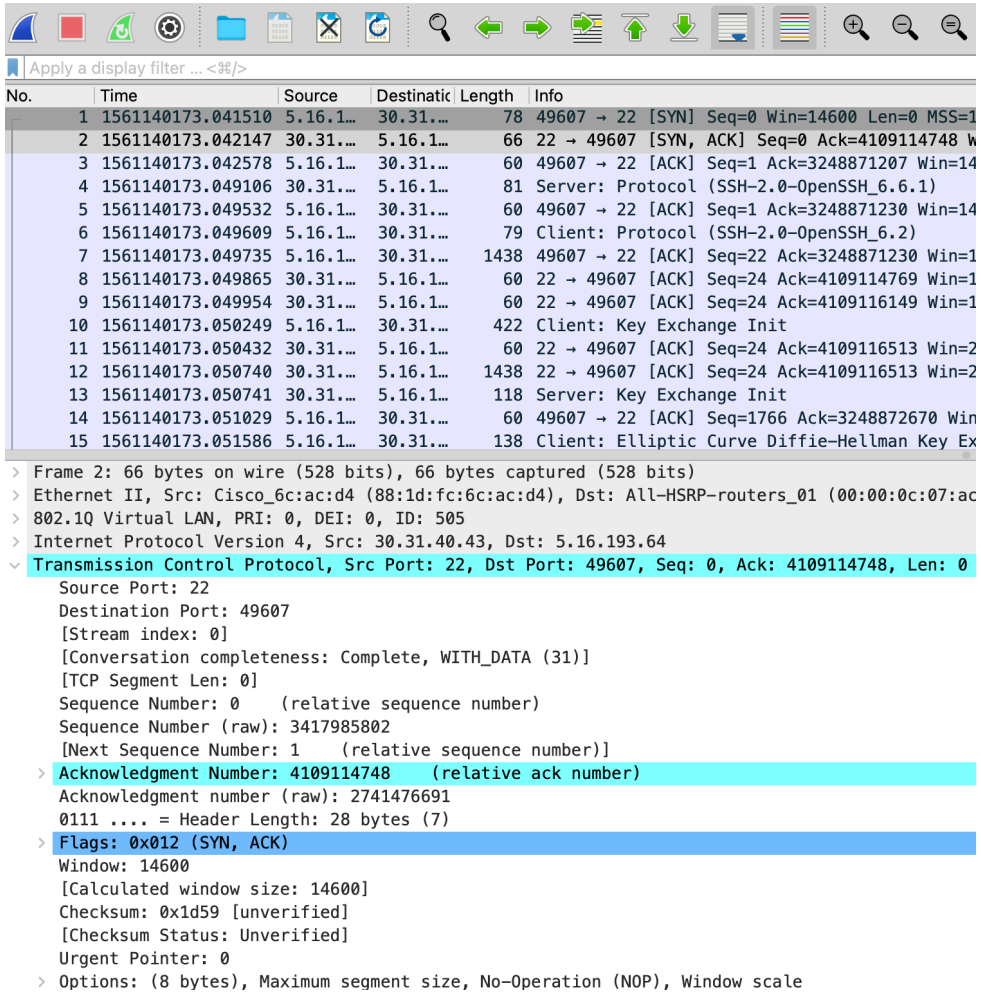


tcp.port == 47992

No.	Time	Source	Destinatic	Length	Info
13	1657653646.308060	10.17.…	10.17.…	184	5201 → 47992 [ACK] Seq=1 Ack=1 Win=245 Len=0
14	1657653646.308060	10.17.…	10.17.…	301	47992 → 5201 [PSH, ACK] Seq=1 Ack=1 Win=491
15	1657653646.308060	10.17.…	10.17.…	184	5201 → 47992 [ACK] Seq=1 Ack=118 Win=245 Len
16	1657653646.308060	10.17.…	10.17.…	185	5201 → 47992 [PSH, ACK] Seq=1 Ack=118 Win=24
17	1657653646.308060	10.17.…	10.17.…	184	47992 → 5201 [ACK] Seq=118 Ack=2 Win=491 Len
19	1657653646.308142	10.17.…	10.17.…	185	5201 → 47992 [PSH, ACK] Seq=2 Ack=118 Win=24
20	1657653646.308166	10.17.…	10.17.…	184	47992 → 5201 [ACK] Seq=118 Ack=3 Win=491 Len
21	1657653646.308210	10.17.…	10.17.…	185	5201 → 47992 [PSH, ACK] Seq=3 Ack=118 Win=24
22	1657653646.308224	10.17.…	10.17.…	184	47992 → 5201 [ACK] Seq=118 Ack=4 Win=491 Len
23	1657653646.308253	10.17.…	10.17.…	192	[TCP Port numbers reused] 47992 → 5201 [SYN]
24	1657653646.308410	10.17.…	10.17.…	192	5201 → 47992 [SYN, ACK] Seq=0 Ack=1 Win=6264
25	1657653646.308410	10.17.…	10.17.…	184	47992 → 5201 [ACK] Seq=1 Ack=1 Win=62848 Len
28	1657653646.308465	10.17.…	10.17.…	221	47992 → 5201 [PSH, ACK] Seq=1 Ack=1 Win=6284
29	1657653646.308495	10.17.…	10.17.…	184	5201 → 47992 [ACK] Seq=1 Ack=38 Win=62720 Le
30	1657653646.308495	10.17.…	10.17.…	185	5201 → 47992 [PSH, ACK] Seq=1 Ack=38 Win=627

```
> Frame 24: 192 bytes on wire (1536 bits), 192 bytes captured (1536 bits)
> Ethernet II, Src: 06:fb:54:7d:88:80 (06:fb:54:7d:88:80), Dst: 06:ad:da:41:2f:10 (06:ad:da:41:2
> Internet Protocol Version 4, Src: 192.168.0.35, Dst: 192.168.0.8
> User Datagram Protocol, Src Port: 60679, Dst Port: 6081
> Generic Network Virtualization Encapsulation, VNI: 0x000000
> Internet Protocol Version 4, Src: 10.17.3.111, Dst: 10.17.3.145
> User Datagram Protocol, Src Port: 65438, Dst Port: 4789
> Virtual eXtensible Local Area Network
> Ethernet II, Src: 06:03:dd:15:19:d4 (06:03:dd:15:19:d4), Dst: 06:55:ff:c5:72:92 (06:55:ff:c5:7
> Internet Protocol Version 4, Src: 10.17.3.30, Dst: 10.17.3.111
v Transmission Control Protocol, Src Port: 5201, Dst Port: 47992, Seq: 0, Ack: 1, Len: 0
    Source Port: 5201
    Destination Port: 47992
    [Stream index: 2]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3693875661
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 419445753
    1010 .... = Header Length: 40 bytes (10)
  > Flags: 0x012 (SYN, ACK)
```

Glitch … or **Reality**?

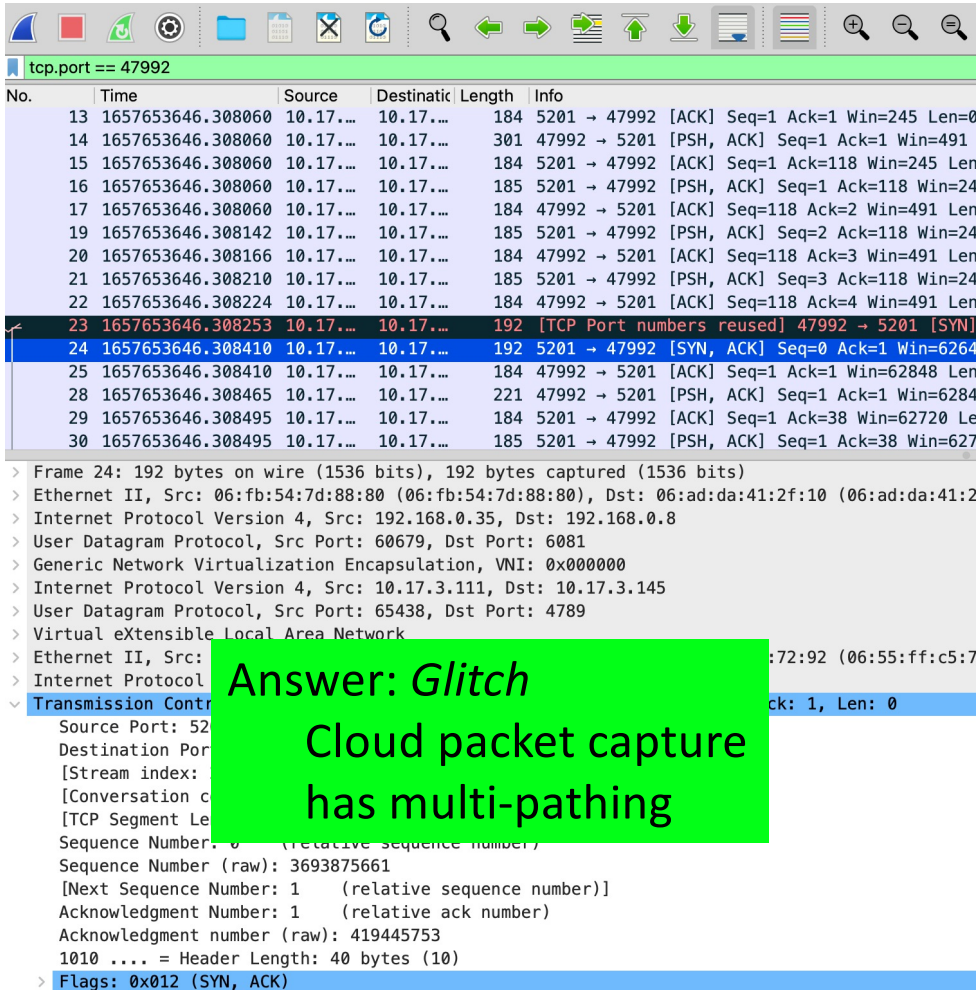


Apply a display filter … <⌘/>

No.	Time	Source	Destinatic	Length	Info
1	1561140173.041510	5.16.1…	30.31.…	78	49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1
2	1561140173.042147	30.31.…	5.16.1…	66	22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 W
3	1561140173.042578	5.16.1…	30.31.…	60	49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14
4	1561140173.049106	30.31.…	5.16.1…	81	Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
5	1561140173.049532	5.16.1…	30.31.…	60	49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14
6	1561140173.049609	5.16.1…	30.31.…	79	Client: Protocol (SSH-2.0-OpenSSH_6.2)
7	1561140173.049735	5.16.1…	30.31.…	1438	49607 → 22 [ACK] Seq=22 Ack=3248871230 Win=1
8	1561140173.049865	30.31.…	5.16.1…	60	22 → 49607 [ACK] Seq=24 Ack=4109114769 Win=1
9	1561140173.049954	30.31.…	5.16.1…	60	22 → 49607 [ACK] Seq=24 Ack=4109116149 Win=1
10	1561140173.050249	5.16.1…	30.31.…	422	Client: Key Exchange Init
11	1561140173.050432	30.31.…	5.16.1…	60	22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=2
12	1561140173.050740	30.31.…	5.16.1…	1438	22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=2
13	1561140173.050741	30.31.…	5.16.1…	118	Server: Key Exchange Init
14	1561140173.051029	5.16.1…	30.31.…	60	49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win
15	1561140173.051586	5.16.1…	30.31.…	138	Client: Elliptic Curve Diffie-Hellman Key Ex

```
> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Cisco_6c:ac:d4 (88:1d:fc:6c:ac:d4), Dst: All-HSRP-routers_01 (00:00:0c:07:ac
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 505
> Internet Protocol Version 4, Src: 30.31.40.43, Dst: 5.16.193.64
v Transmission Control Protocol, Src Port: 22, Dst Port: 49607, Seq: 0, Ack: 4109114748, Len: 0
    Source Port: 22
    Destination Port: 49607
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3417985802
    [Next Sequence Number: 1    (relative sequence number)]
  > Acknowledgment Number: 4109114748    (relative ack number)
    Acknowledgment number (raw): 2741476691
    0111 .... = Header Length: 28 bytes (7)
  > Flags: 0x012 (SYN, ACK)
    Window: 14600
    [Calculated window size: 14600]
    Checksum: 0x1d59 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (8 bytes), Maximum segment size, No-Operation (NOP), Window scale
```

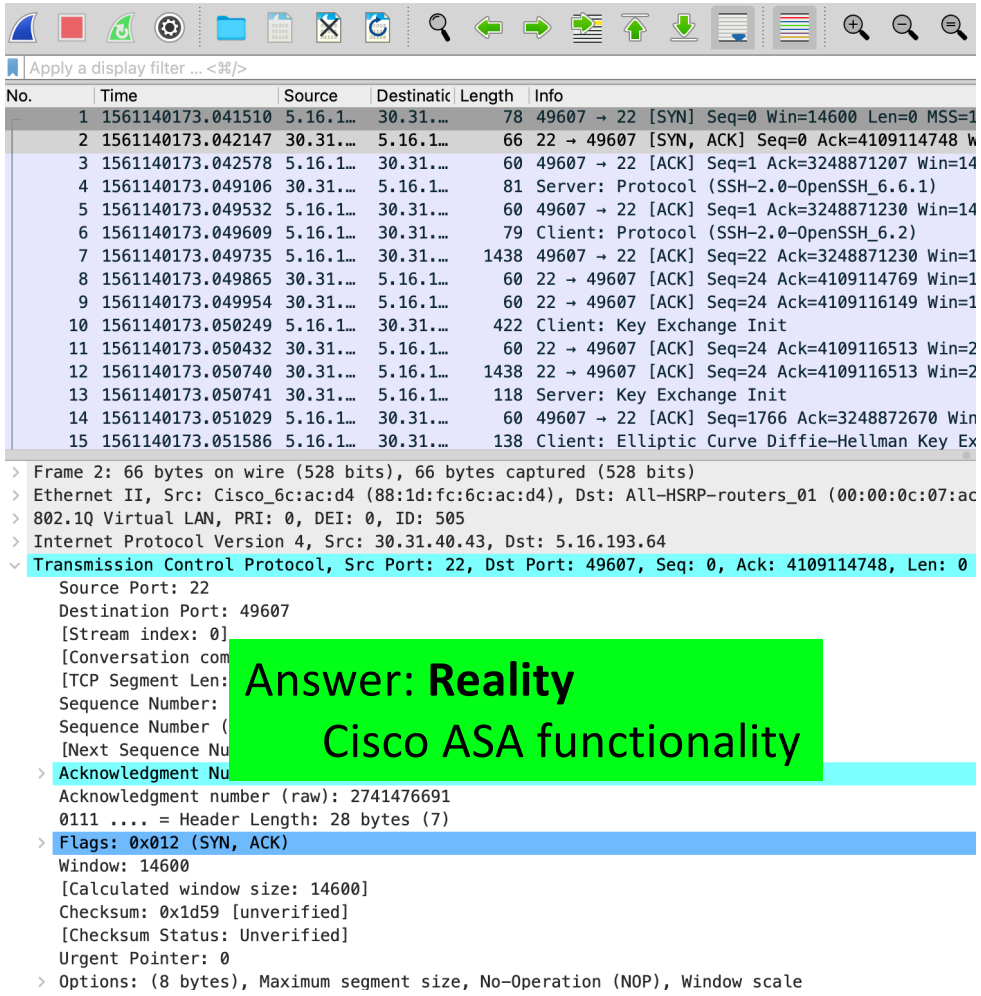
Glitch!



Answer: *Glitch*

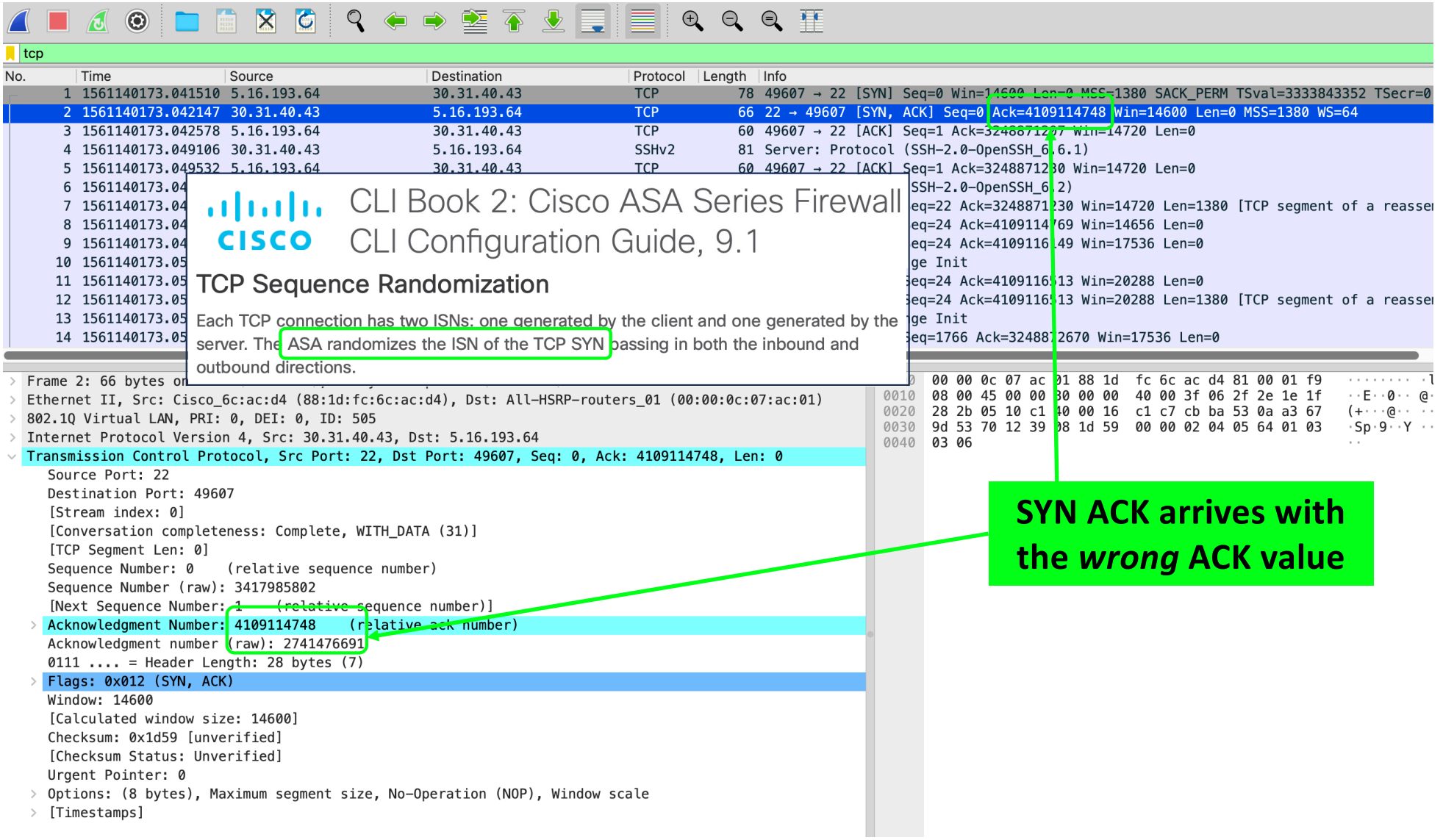
Cloud packet capture has multi-pathing

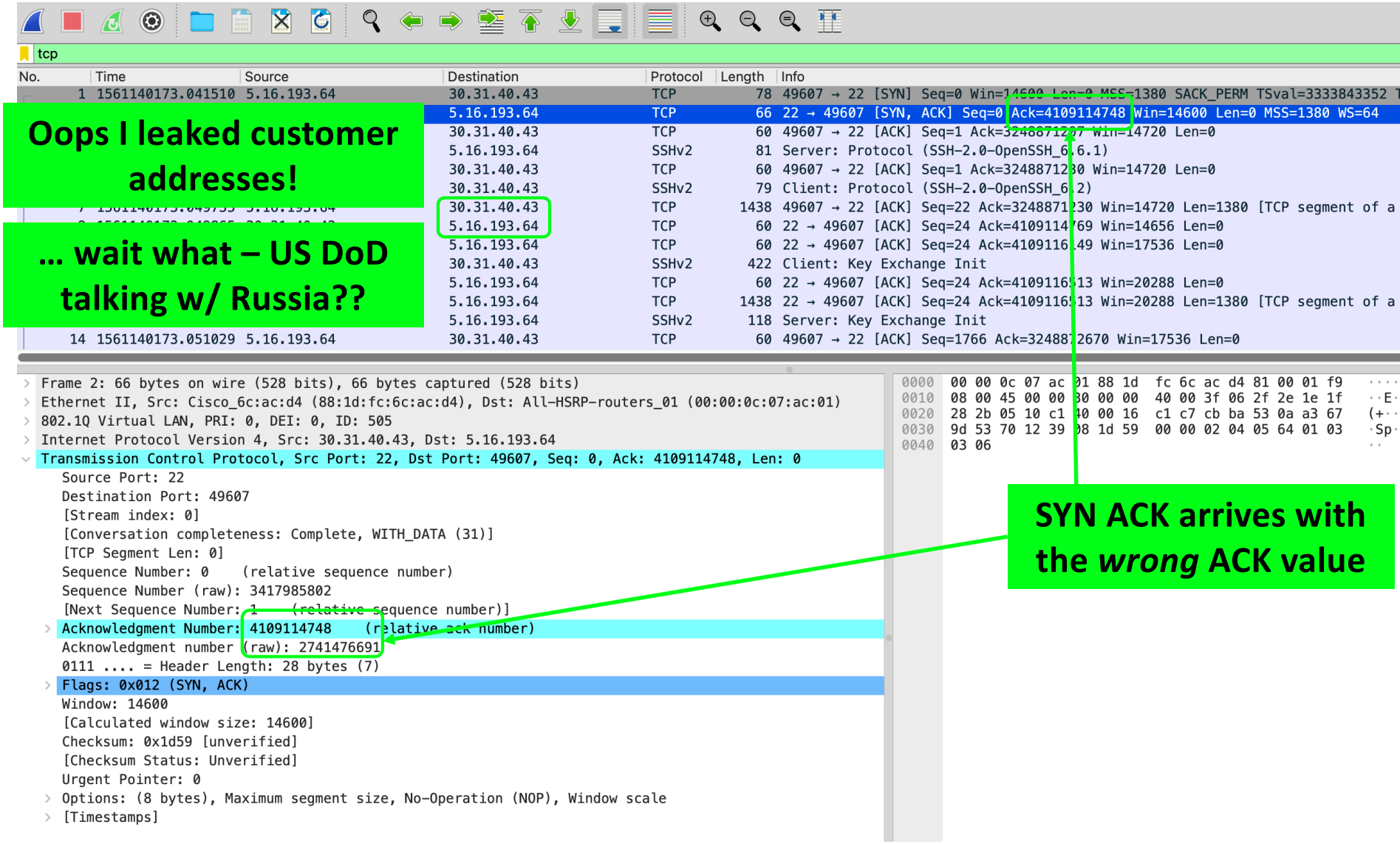
Reality!

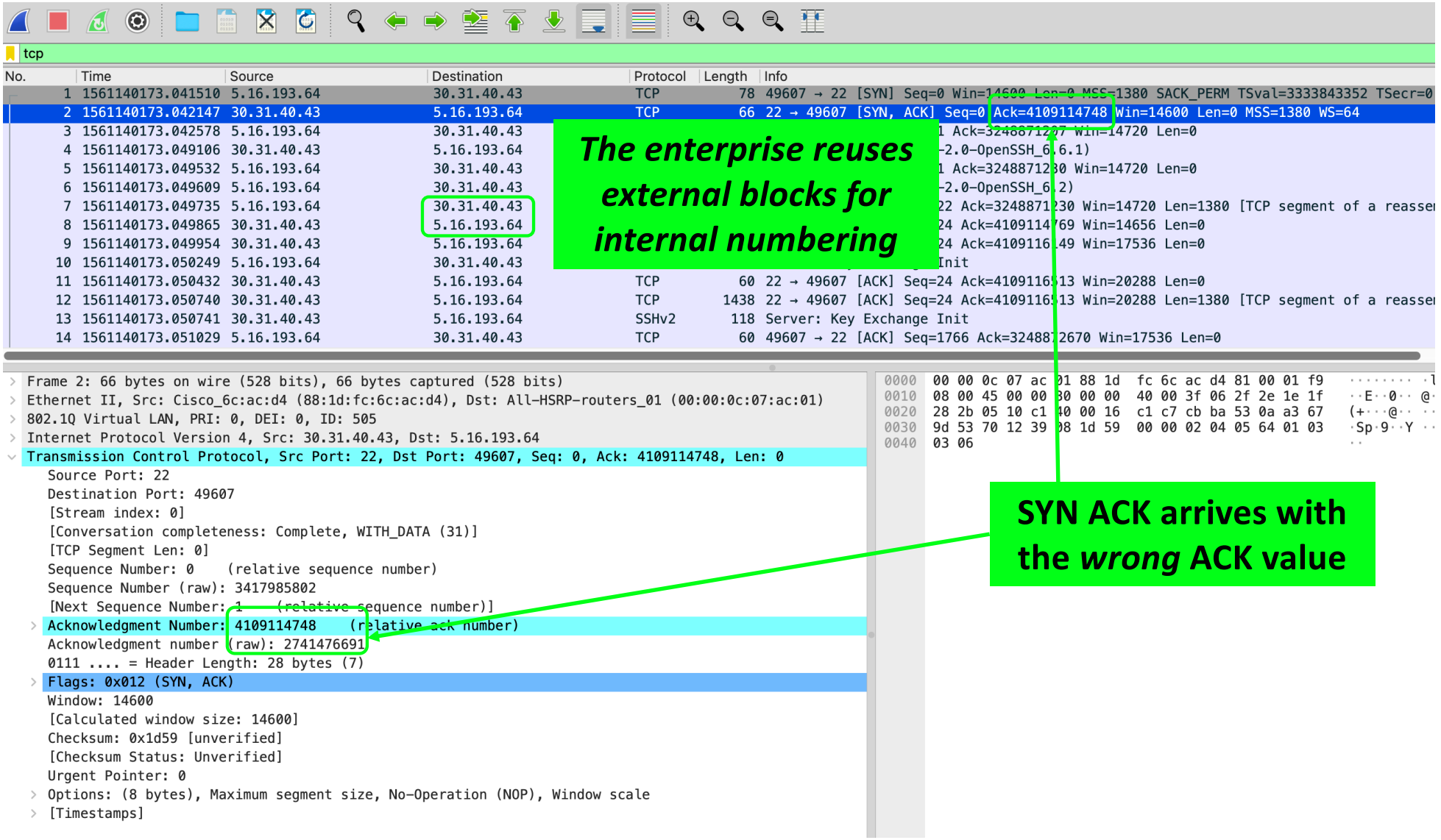


Answer: **Reality**

Cisco ASA functionality


tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
2 1561140173.042147 30.31.40.43 5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
3 1561140173.042578 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14720 Len=0
4 1561140173.049106 30.31.40.43 5.16.193.64 SSHv2 81 Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
5 1561140173.049532 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14720 Len=0
6 1561140173.04 SSH-2.0-OpenSSH_6.2)
7 1561140173.04 eq=22 Ack=3248871230 Win=14720 Len=1380 [TCP segment of a reasse
8 1561140173.04 eq=24 Ack=4109114769 Win=14656 Len=0
9 1561140173.04 eq=24 Ack=4109116149 Win=17536 Len=0
10 1561140173.05 ge Init
11 1561140173.05 eq=24 Ack=4109116513 Win=20288 Len=0
12 1561140173.05 eq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
13 1561140173.05 ge Init
14 1561140173.05 eq=1766 Ack=3248872670 Win=17536 Len=0
CLI Book 2: Cisco ASA Series Firewall CLI Configuration Guide, 9.1
TCP Sequence Randomization
Each TCP connection has two ISNs: one generated by the client and one generated by the server. The ASA randomizes the ISN of the TCP SYN passing in both the inbound and outbound directions.
Frame 2: 66 bytes on
Ethernet II, Src: Cisco_6c:ac:d4 (88:1d:fc:6c:ac:d4), Dst: All-HSRP-routers_01 (00:00:0c:07:ac:01)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 505
Internet Protocol Version 4, Src: 30.31.40.43, Dst: 5.16.193.64
Transmission Control Protocol, Src Port: 22, Dst Port: 49607, Seq: 0, Ack: 4109114748, Len: 0
Source Port: 22
Destination Port: 49607
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3417985802
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 4109114748 (relative ack number)
Acknowledgment number (raw): 2741476691
0111 .... = Header Length: 28 bytes (7)
Flags: 0x012 (SYN, ACK)
Window: 14600
[Calculated window size: 14600]
Checksum: 0x1d59 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (8 bytes), Maximum segment size, No-Operation (NOP), Window scale
[Timestamps]
0000 00 00 0c 07 ac 01 88 1d fc 6c ac d4 81 00 01 f9
0010 08 00 45 00 00 30 00 00 40 00 3f 06 2f 2e 1e 1f
0020 28 2b 05 10 c1 40 00 16 c1 c7 cb ba 53 0a a3 67
0030 9d 53 70 12 39 08 1d 59 00 00 02 04 05 64 01 03
0040 03 06
SYN ACK arrives with the wrong ACK value


Oops I leaked customer addresses!
… wait what – US DoD talking w/ Russia??
tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871207 Win=14720 Len=0
5.16.193.64 SSHv2 81 Server: Protocol (SSH-2.0-OpenSSH_6.6.1)
30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1 Ack=3248871230 Win=14720 Len=0
30.31.40.43 SSHv2 79 Client: Protocol (SSH-2.0-OpenSSH_6.2)
30.31.40.43 TCP 1438 49607 → 22 [ACK] Seq=22 Ack=3248871230 Win=14720 Len=1380 [TCP segment of a reasse
5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109114769 Win=14656 Len=0
5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116149 Win=17536 Len=0
30.31.40.43 SSHv2 422 Client: Key Exchange Init
5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=0
5.16.193.64 TCP 1438 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
5.16.193.64 SSHv2 118 Server: Key Exchange Init
14 1561140173.051029 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win=17536 Len=0
Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Cisco_6c:ac:d4 (88:1d:fc:6c:ac:d4), Dst: All-HSRP-routers_01 (00:00:0c:07:ac:01)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 505
Internet Protocol Version 4, Src: 30.31.40.43, Dst: 5.16.193.64
Transmission Control Protocol, Src Port: 22, Dst Port: 49607, Seq: 0, Ack: 4109114748, Len: 0
Source Port: 22
Destination Port: 49607
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3417985802
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 4109114748 (relative ack number)
Acknowledgment number (raw): 2741476691
0111 .... = Header Length: 28 bytes (7)
Flags: 0x012 (SYN, ACK)
Window: 14600
[Calculated window size: 14600]
Checksum: 0x1d59 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (8 bytes), Maximum segment size, No-Operation (NOP), Window scale
[Timestamps]
0000 00 00 0c 07 ac 01 88 1d fc 6c ac d4 81 00 01 f9
0010 08 00 45 00 00 30 00 00 40 00 3f 06 2f 2e 1e 1f
0020 28 2b 05 10 c1 40 00 16 c1 c7 cb ba 53 0a a3 67
0030 9d 53 70 12 39 08 1d 59 00 00 02 04 05 64 01 03
0040 03 06
SYN ACK arrives with the wrong ACK value


tcp
No. Time Source Destination Protocol Length Info
1 1561140173.041510 5.16.193.64 30.31.40.43 TCP 78 49607 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1380 SACK_PERM TSval=3333843352 TSecr=0
2 1561140173.042147 30.31.40.43 5.16.193.64 TCP 66 22 → 49607 [SYN, ACK] Seq=0 Ack=4109114748 Win=14600 Len=0 MSS=1380 WS=64
3 1561140173.042578 5.16.193.64 30.31.40.43
4 1561140173.049106 30.31.40.43 5.16.193.64
5 1561140173.049532 5.16.193.64 30.31.40.43
6 1561140173.049609 5.16.193.64 30.31.40.43
7 1561140173.049735 5.16.193.64 30.31.40.43
8 1561140173.049865 30.31.40.43 5.16.193.64
9 1561140173.049954 30.31.40.43 5.16.193.64
10 1561140173.050249 5.16.193.64 30.31.40.43
11 1561140173.050432 30.31.40.43 5.16.193.64 TCP 60 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=0
12 1561140173.050740 30.31.40.43 5.16.193.64 TCP 1438 22 → 49607 [ACK] Seq=24 Ack=4109116513 Win=20288 Len=1380 [TCP segment of a reasse
13 1561140173.050741 30.31.40.43 5.16.193.64 SSHv2 118 Server: Key Exchange Init
14 1561140173.051029 5.16.193.64 30.31.40.43 TCP 60 49607 → 22 [ACK] Seq=1766 Ack=3248872670 Win=17536 Len=0
The enterprise reuses external blocks for internal numbering
Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Cisco_6c:ac:d4 (88:1d:fc:6c:ac:d4), Dst: All-HSRP-routers_01 (00:00:0c:07:ac:01)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 505
Internet Protocol Version 4, Src: 30.31.40.43, Dst: 5.16.193.64
Transmission Control Protocol, Src Port: 22, Dst Port: 49607, Seq: 0, Ack: 4109114748, Len: 0
Source Port: 22
Destination Port: 49607
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3417985802
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 4109114748 (relative ack number)
Acknowledgment number (raw): 2741476691
0111 .... = Header Length: 28 bytes (7)
Flags: 0x012 (SYN, ACK)
Window: 14600
[Calculated window size: 14600]
Checksum: 0x1d59 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (8 bytes), Maximum segment size, No-Operation (NOP), Window scale
[Timestamps]
0000 00 00 0c 07 ac 01 88 1d fc 6c ac d4 81 00 01 f9
0010 08 00 45 00 00 30 00 00 40 00 3f 06 2f 2e 1e 1f
0020 28 2b 05 10 c1 40 00 16 c1 c7 cb ba 53 0a a3 67
0030 9d 53 70 12 39 08 1d 59 00 00 02 04 05 64 01 03
0040 03 06
SYN ACK arrives with the wrong ACK value

Reflecting on all this… in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us

We find Zeek has fans at a *Very Big Company*!

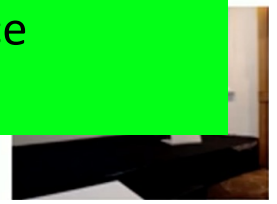
Reflecting on all this... in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- Howe assive
  - Exploiting luck where we found it
  - Changing questions to fit opportu
  - Building on "secret weapons"
  - Brute force effort to overcome pr
  - Actively selling results to those w

Startups 101: Products!

More *Aha's* relating to Startups 101:

- Why you don't want to make a profit
- Why you're supposed to run out of money
- The big deal regarding subscriptions
- What's up with always getting asked for 1-10 ratings
- The #1 thing a good sales person does
- What "Burn:NARR ratio" is & its importance
- Why VCs care about the *Rule Of 40*



Reflecting on all this… in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us

Zeek Community +
Zeek practical difficulties =
Migration Effect

Reflecting on all this… in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us

Need a SaaS story + some ML, please? $$$

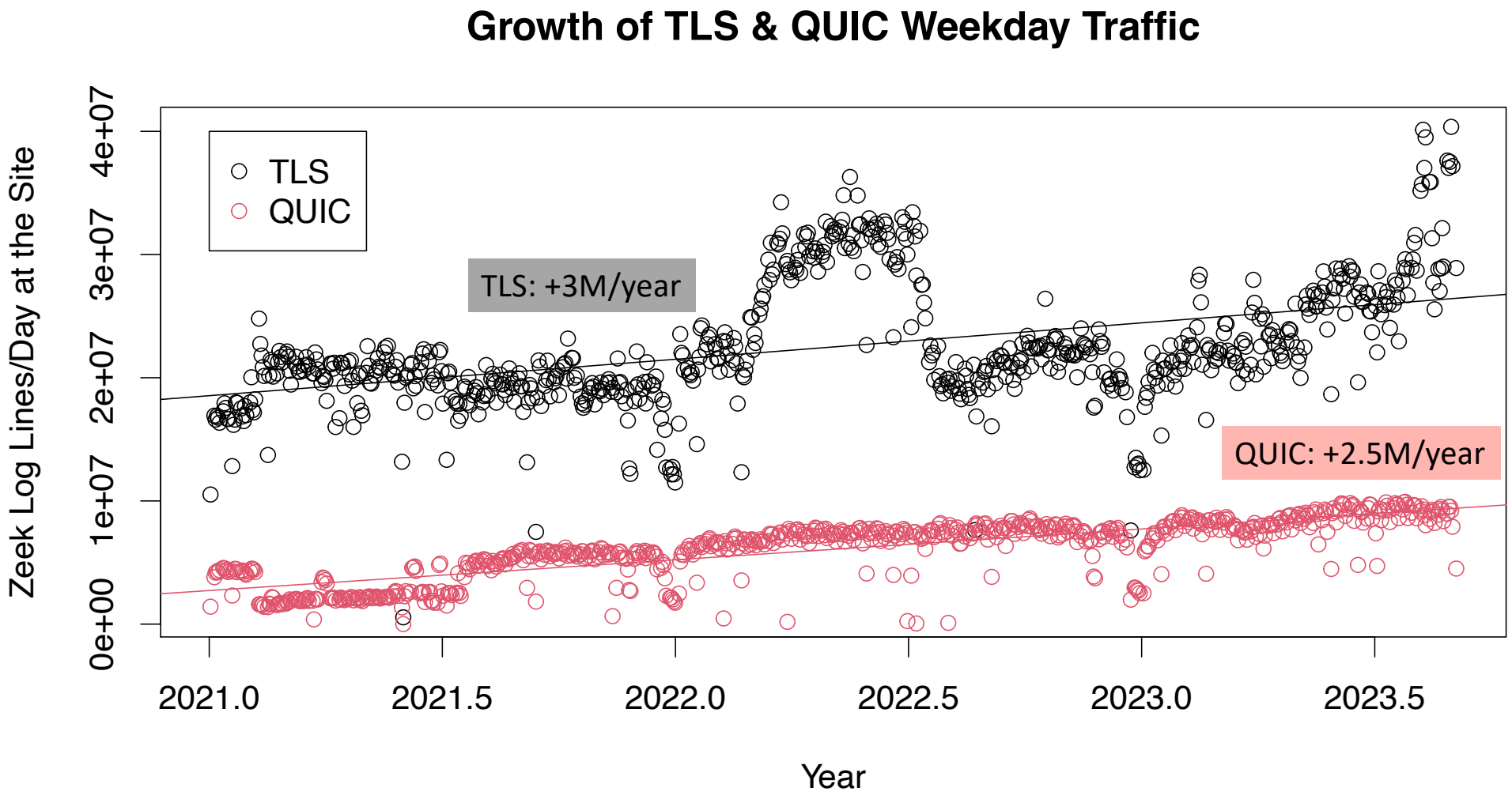
The Inhomogeneous Future: How Pervasive Is QUIC?

F5

PUBLISHED
FEBRUARY 22,
2021

QUIC (not an acronym) is a unique beast, but is best thought of as a new transport protocol that solves many longstanding problems in the internet and captures most of the value provided by TCP, TLS, SCTP, IPSec, and HTTP/2. There is a new version of HTTP called HTTP/3 that is designed to work over UDP/QUIC instead of TCP/TLS.

Google deployed an early version of HTTP over QUIC in its browsers and web servers as early as 2014. An IETF standardization process began in 2016. After much evolution and data gathering, this will result in the first batch of RFCs in early 2021. Several major internet companies, including F5, have either shipped products that use QUIC or have deployed QUIC in their infrastructure. As of October 2020, 75% of Facebook's traffic was over HTTP/3 and QUIC.


Growth of TLS & QUIC Weekday Traffic
Zeek Log Lines/Day at the Site
0e+00
1e+07
2e+07
3e+07
4e+07
TLS
QUIC
TLS: +3M/year
QUIC: +2.5M/year
2021.0
2021.5
2022.0
2022.5
2023.0
2023.5
Year

Site	IPv6	QUIC:(TLS+QUIC)
LBNL	62%/2%/0.14%	19%

Proportion of connections as of late 2023

The Inhomogeneous Future: How Pervasive Is TLS v1.3? (*)

* (Hides certs; may hide SNI)

Tracking the deployment of TLS 1.3 on the Web: A story of experimentation and centralization

Ralph Holz[1,2] , Jens Hiller[3] , Johanna Amann[4,2] , Abbas Razaghpanah[4] , Thomas Jost[3] , Narseo Vallina-Rodriguez[4,5] , Oliver Hohlfeld[6]
[1]University of Twente, [2]University of Sydney, [3]RWTH Aachen University, [4]ICSI, [5]IMDEA Networks, [6]Brandenburg University of Technology
r.holz@utwente.nl,{hiller,jost}@comsys.rwth-aachen.de,{johanna,abbas}@icir.org,narseo.vallina@imdea.org,hohlfeld@b-tu.de

ACM SIGCOMM Computer Communication Review

Volume 50 Issue 3, July 2020

For a profound view, we combine and analyze data from active domain scans, passive monitoring of large networks, and a crowd-sourcing effort on Android devices. In contrast to TLS 1.2, where adoption took more than five years and was prompted by severe attacks on previous versions, TLS 1.3 is deployed surprisingly speedily and without security concerns calling for it. Just 15 months after standardization, it is used in about 20% of connections we observe.

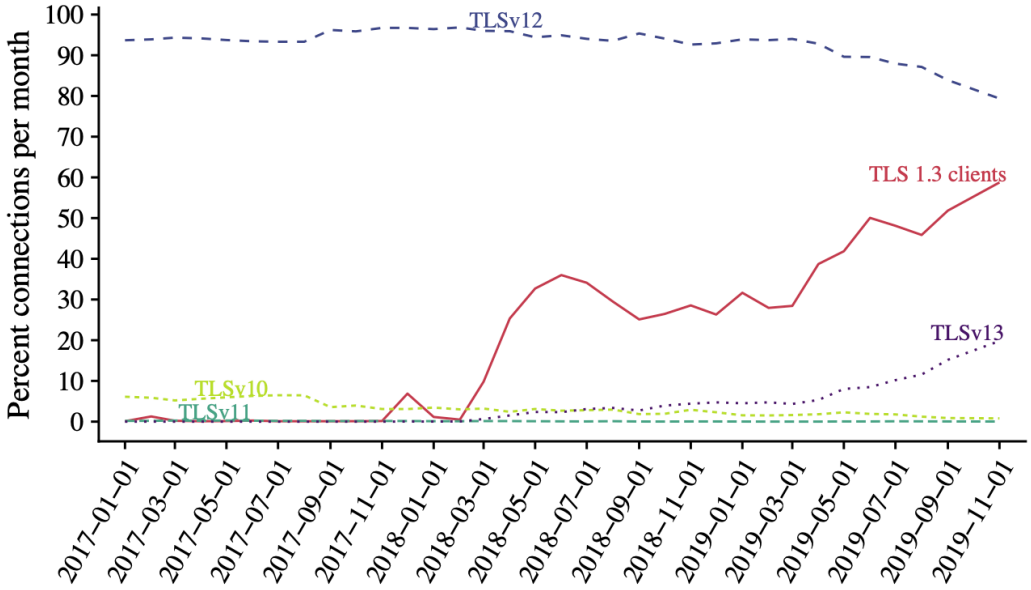
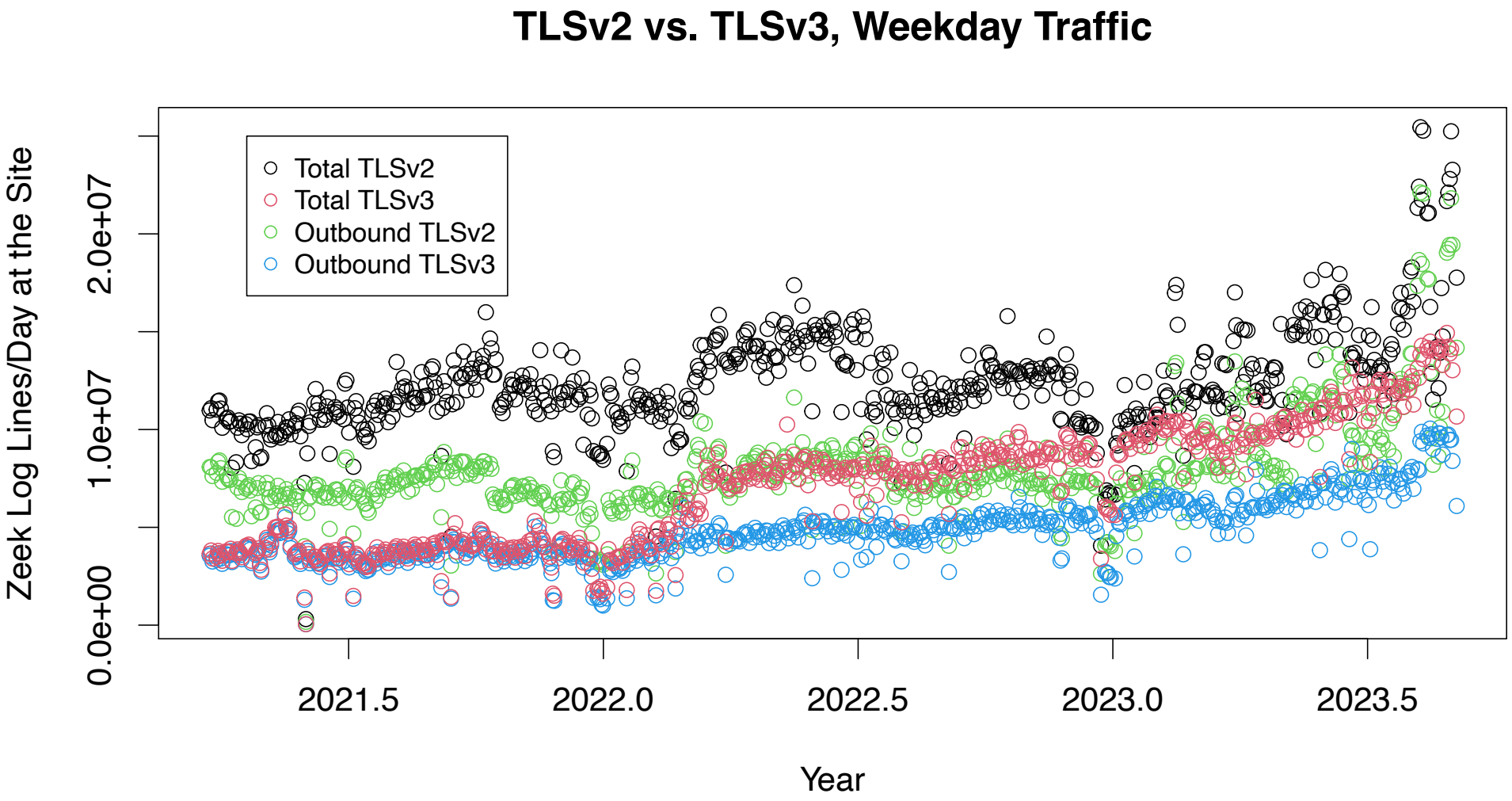
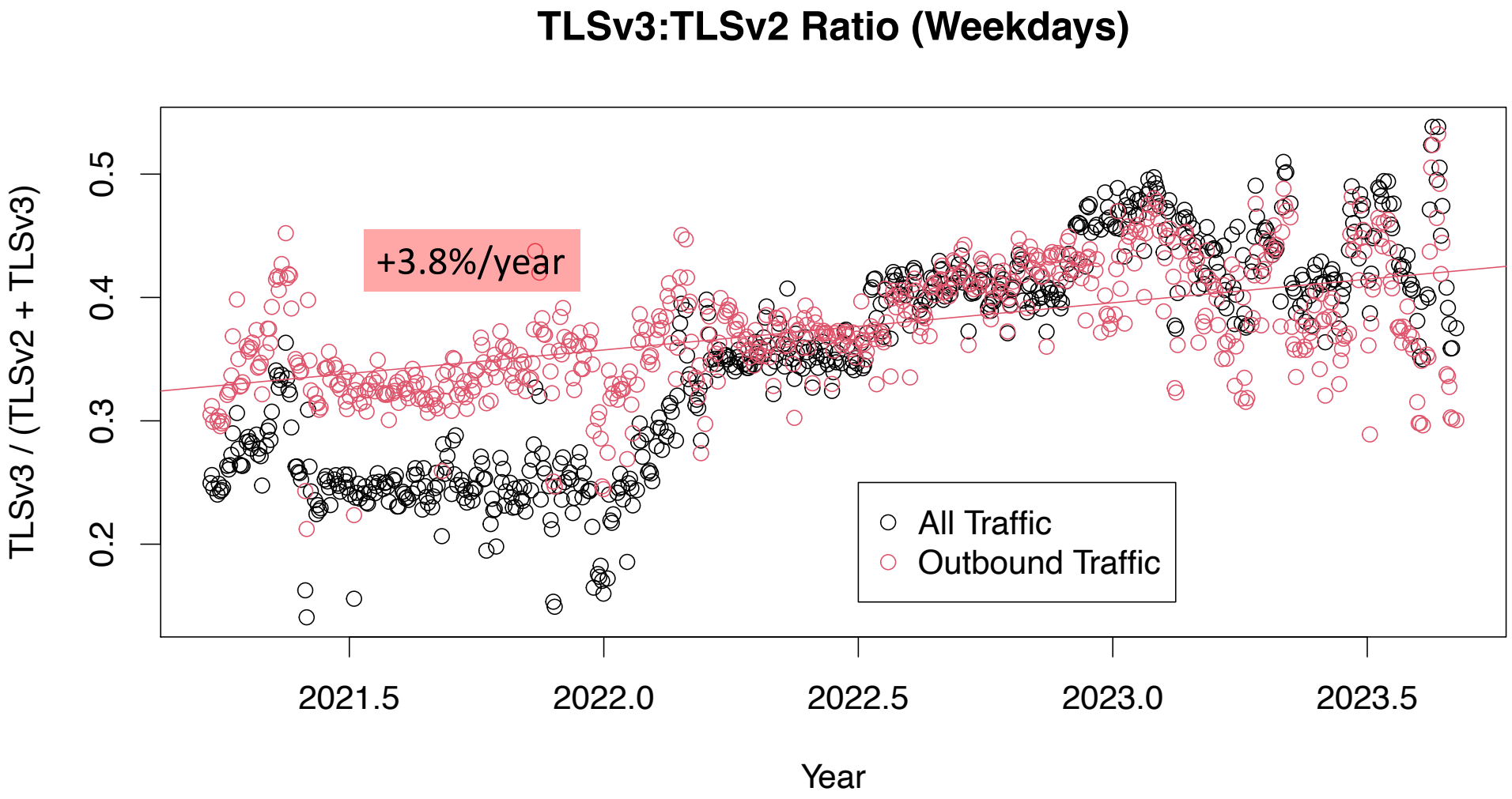


Figure 3: TLS 1.3 connections, view by Notary. Servers select TLS versions from client offers; clients offering TLS 1.3 in red.


TLSv2 vs. TLSv3, Weekday Traffic
Total TLSv2
Total TLSv3
Outbound TLSv2
Outbound TLSv3
Zeek Log Lines/Day at the Site
0.0e+00
1.0e+07
2.0e+07
2021.5
2022.0
2022.5
2023.0
2023.5
Year

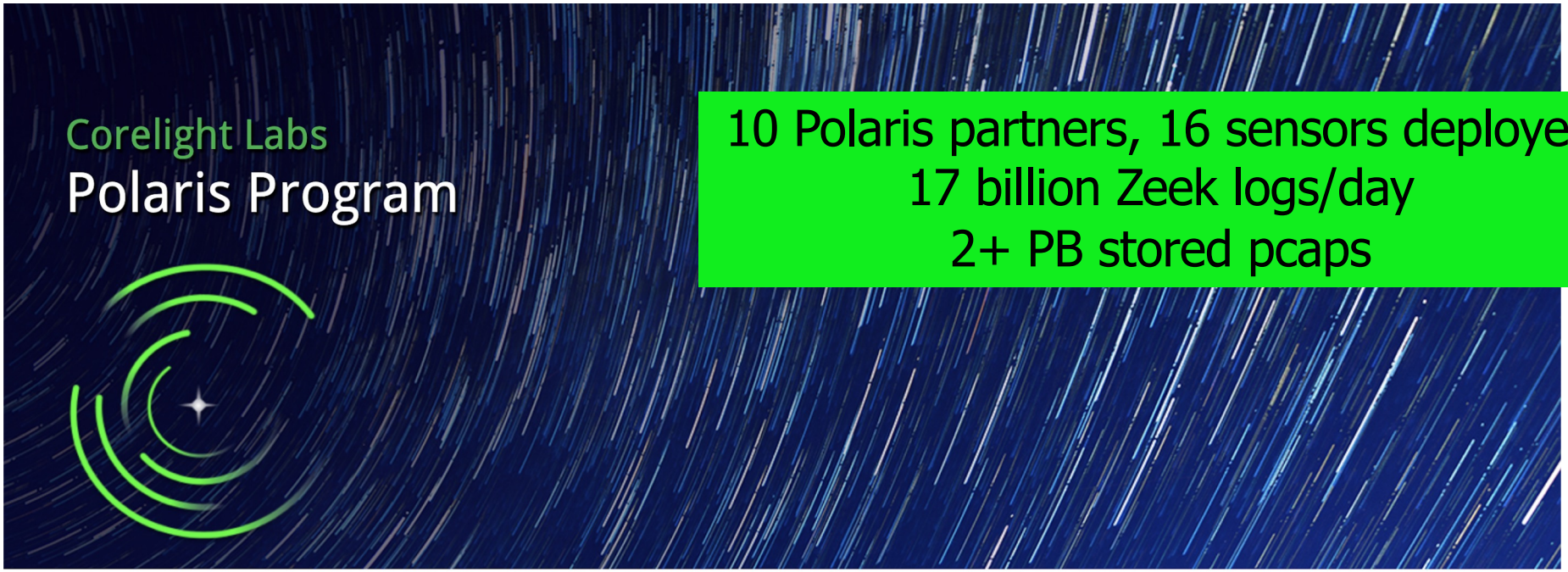

TLSv3:TLSv2 Ratio (Weekdays)
TLSv3 / (TLSv2 + TLSv3)
0.2
0.3
0.4
0.5
+3.8%/year
All Traffic
Outbound Traffic
2021.5
2022.0
2022.5
2023.0
2023.5
Year

Site	IPv6	QUIC:(TLS+QUIC)	TLS 1.3:TLS
LBNL	62%/2%/0.14%	19%	39%

Proportion of connections as of late 2023

Reflecting on all this… in the context of founding a startup

- We didn't intend to do **any** of that
  - When we started, we neither understood the problems nor the solutions
  - Serendipity played key role
  - At every given point in time, there was a disaster to deal with
- However, we were not passive either
  - Exploiting luck where we found it
  - Changing questions to fit opportunities
  - Building on "secret weapons"
  - Brute force effort to overcome problems that didn't have clean answers
  - Actively selling results to those who could help us



There is no substitute for access to extensive operational traffic. That's why the Polaris Program was created, and why it's vital to our research. Polaris is a partnership between Corelight and a select few of our customers that allows us to tap into the complexity of modern enterprise networks — vastly exceeding what's available in simulations or lab setups.

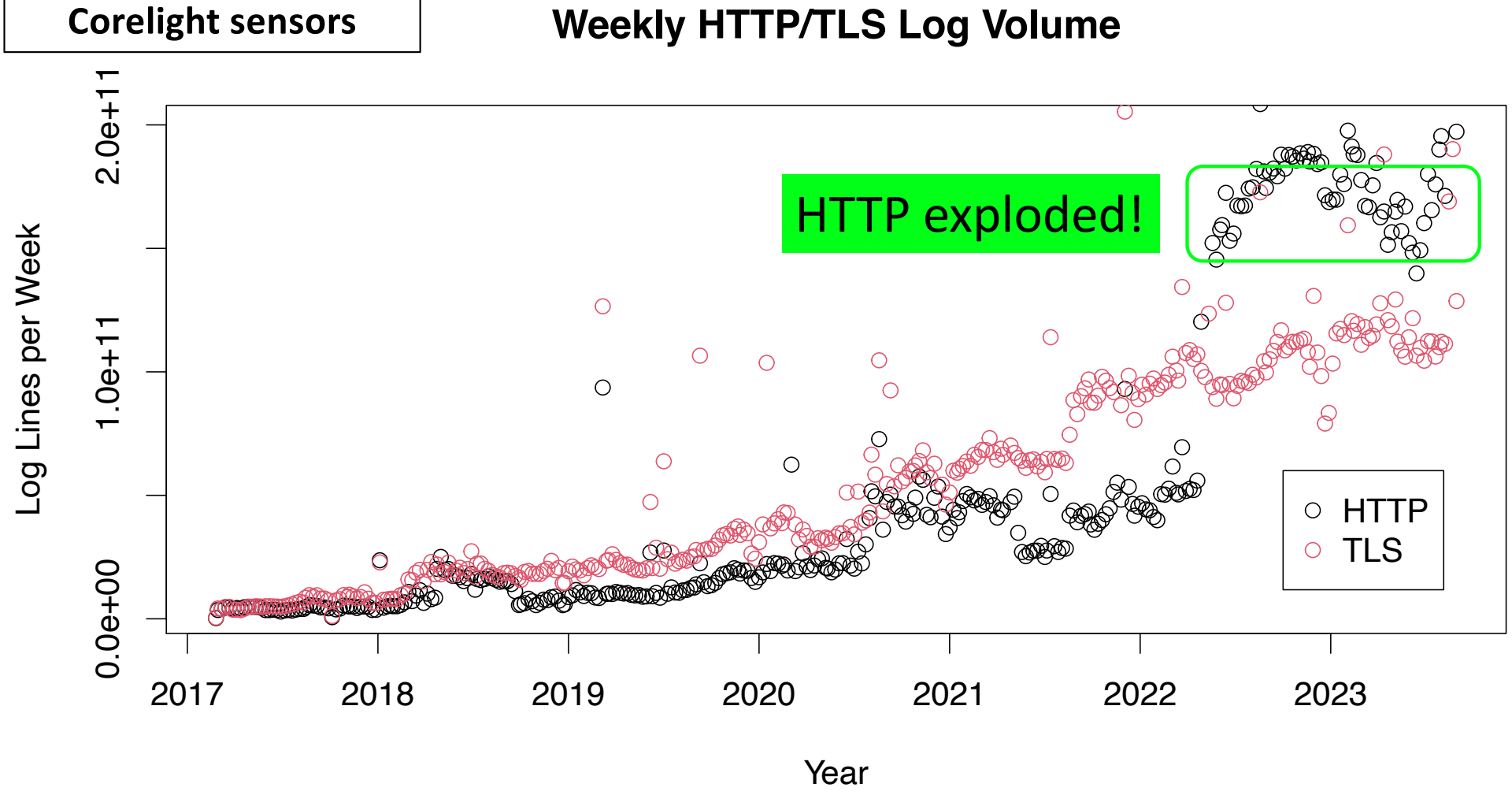
Polaris partners grant Corelight research access to their operational networks, and in return, receive expert attention to their networks via ongoing interactions with our research team and early access to the new capabilities we develop.

Site	IPv6	QUIC:(TLS+QUIC)	TLS 1.3:TLS	Encrypted SMB
LBNL	62%/2%/0.14%	19%	39%	
Polaris A	*0%*	9%	36%	
Polaris B	10^{-4}	10%	26%	
Polaris C	1.7%	10%	27%	
Polaris D	*none*	*0%*	20%	
Polaris E	*none*	*none*	2%*	*none*†
Polaris F	*none*	*none*	5%	*none*†
Polaris G	*0%*	*none*	13%*	*none*†
Polaris H	1.8%	1%	26%	
Polaris I	*none*	*0%*	12%	2.3%†

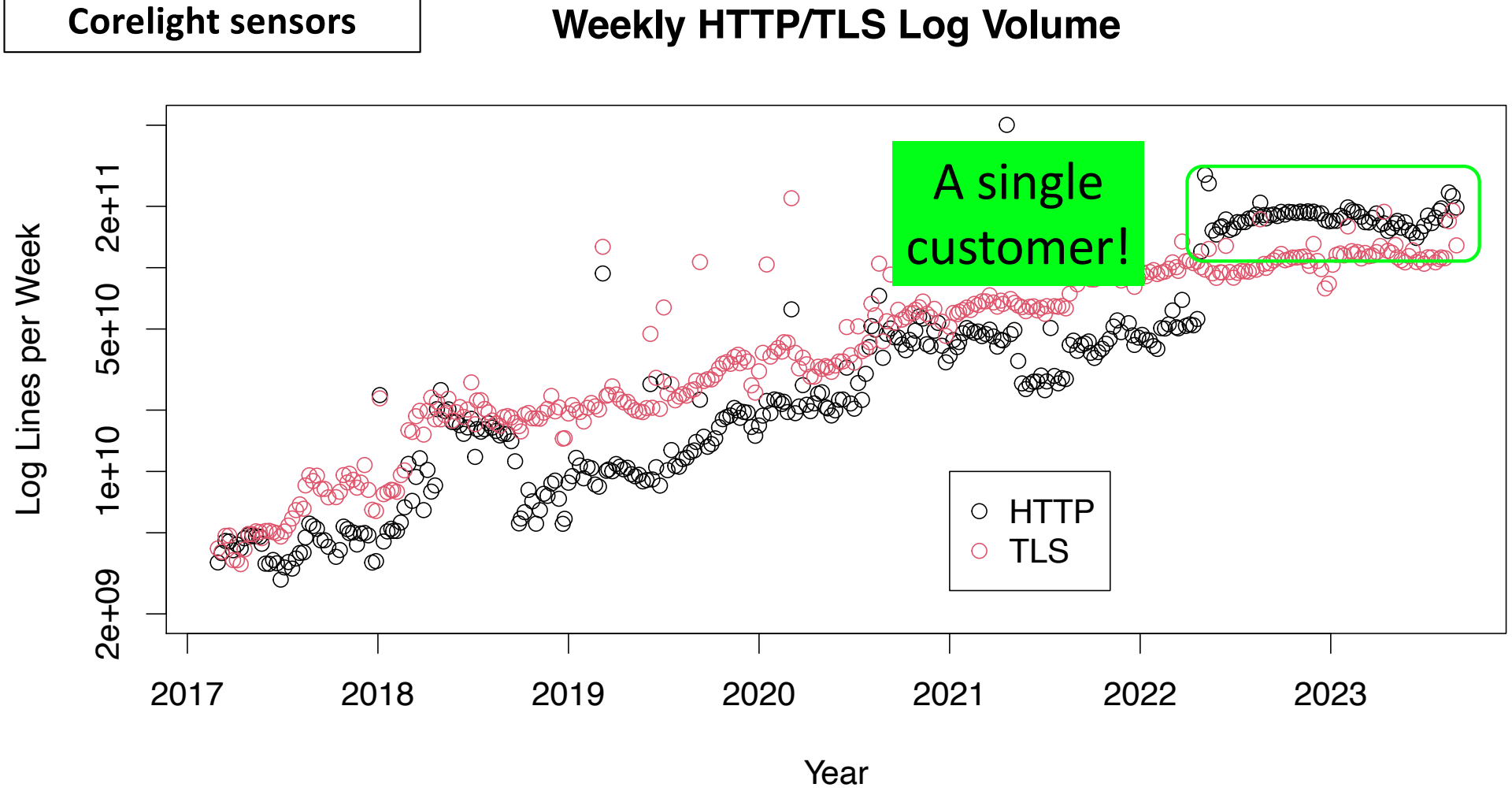
* More TLS 1.0 than 1.3
† 0.1-1% SMBv1

Okay But At Least Not Much Clear-Text HTTP, Right?

Telemetry from online Corelight sensors

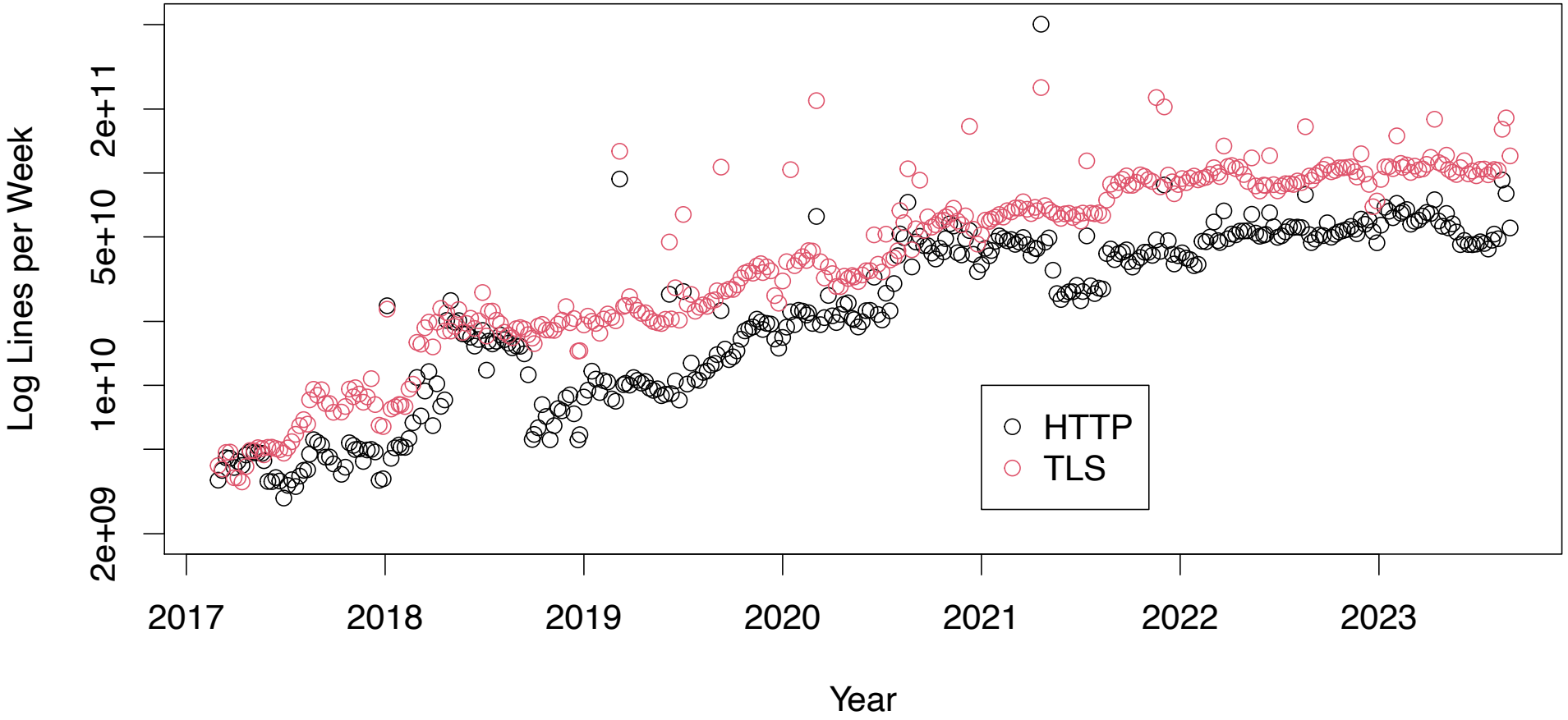


Telemetry from online Corelight sensors

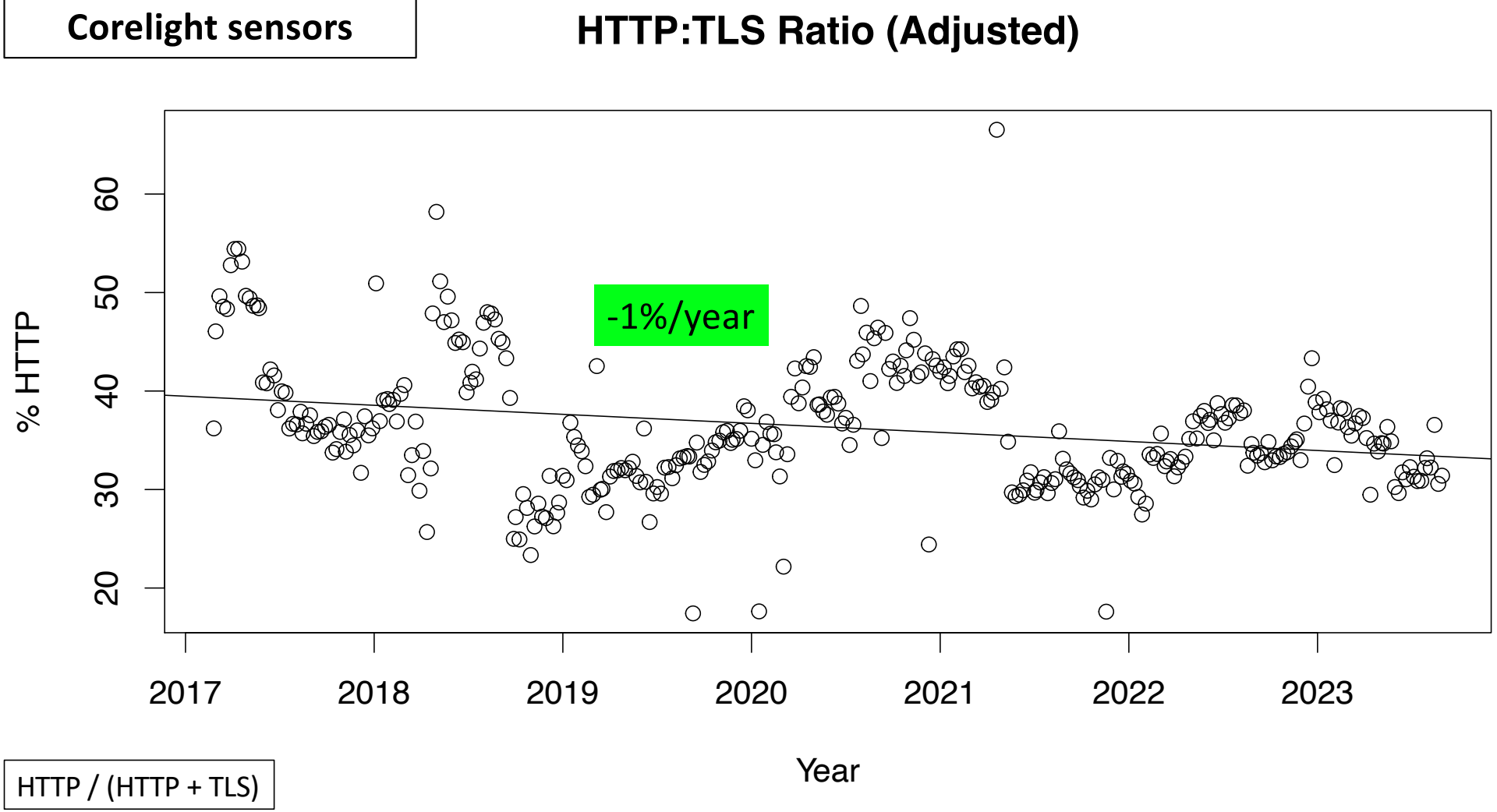


Telemetry from online Corelight sensors

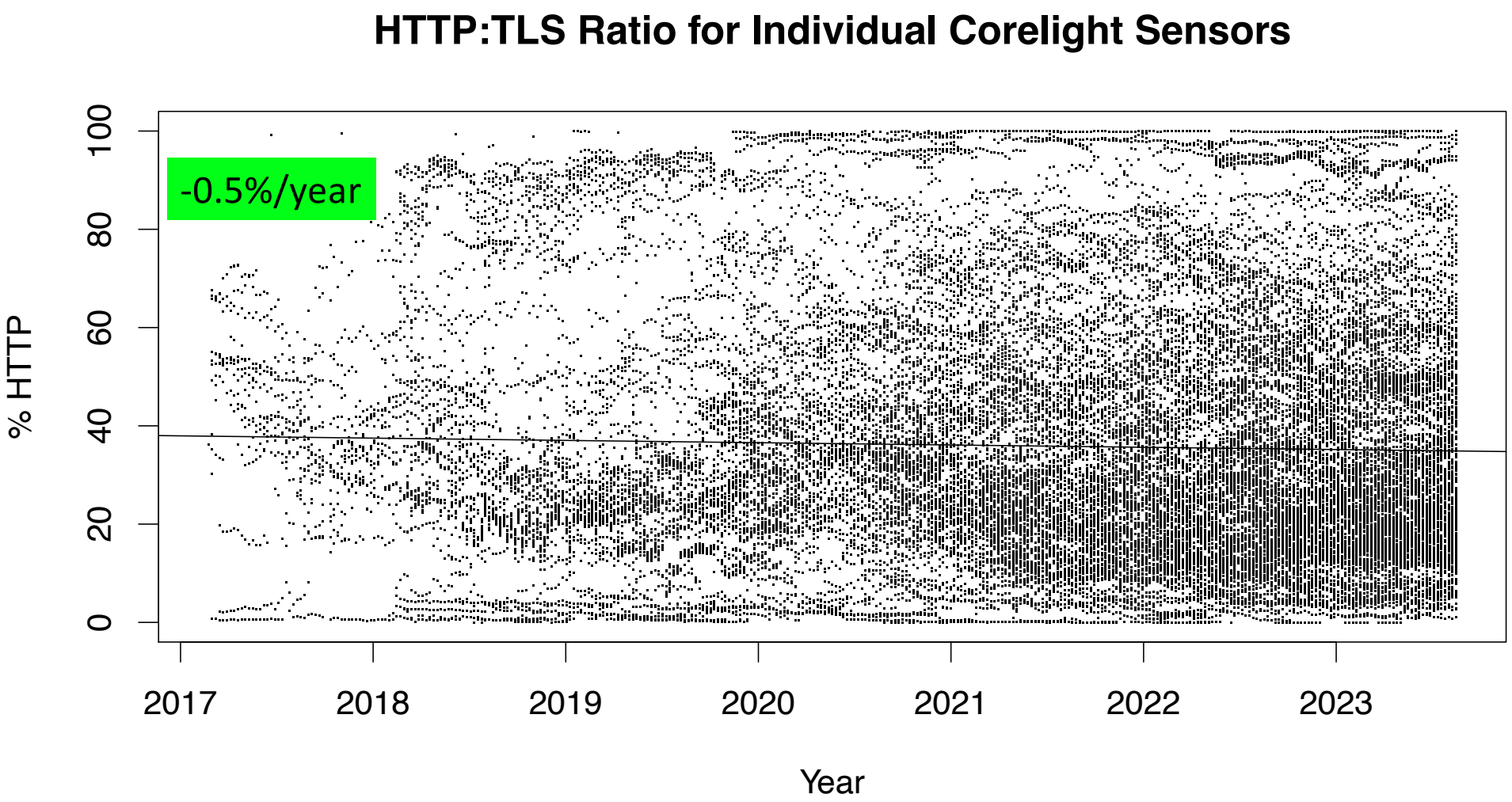
Adjusted Weekly HTTP/TLS Log Volume



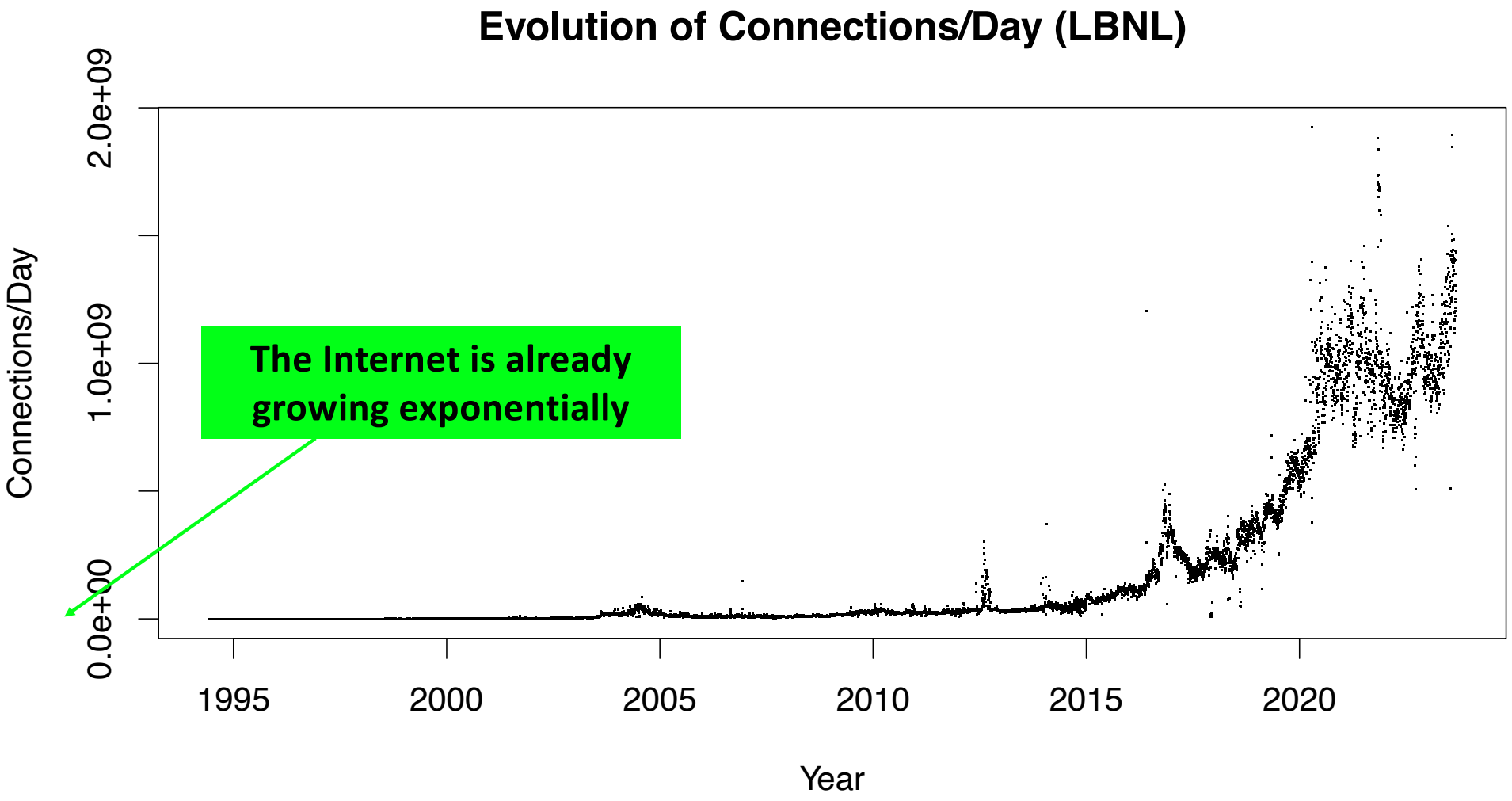
Telemetry from online Corelight sensors

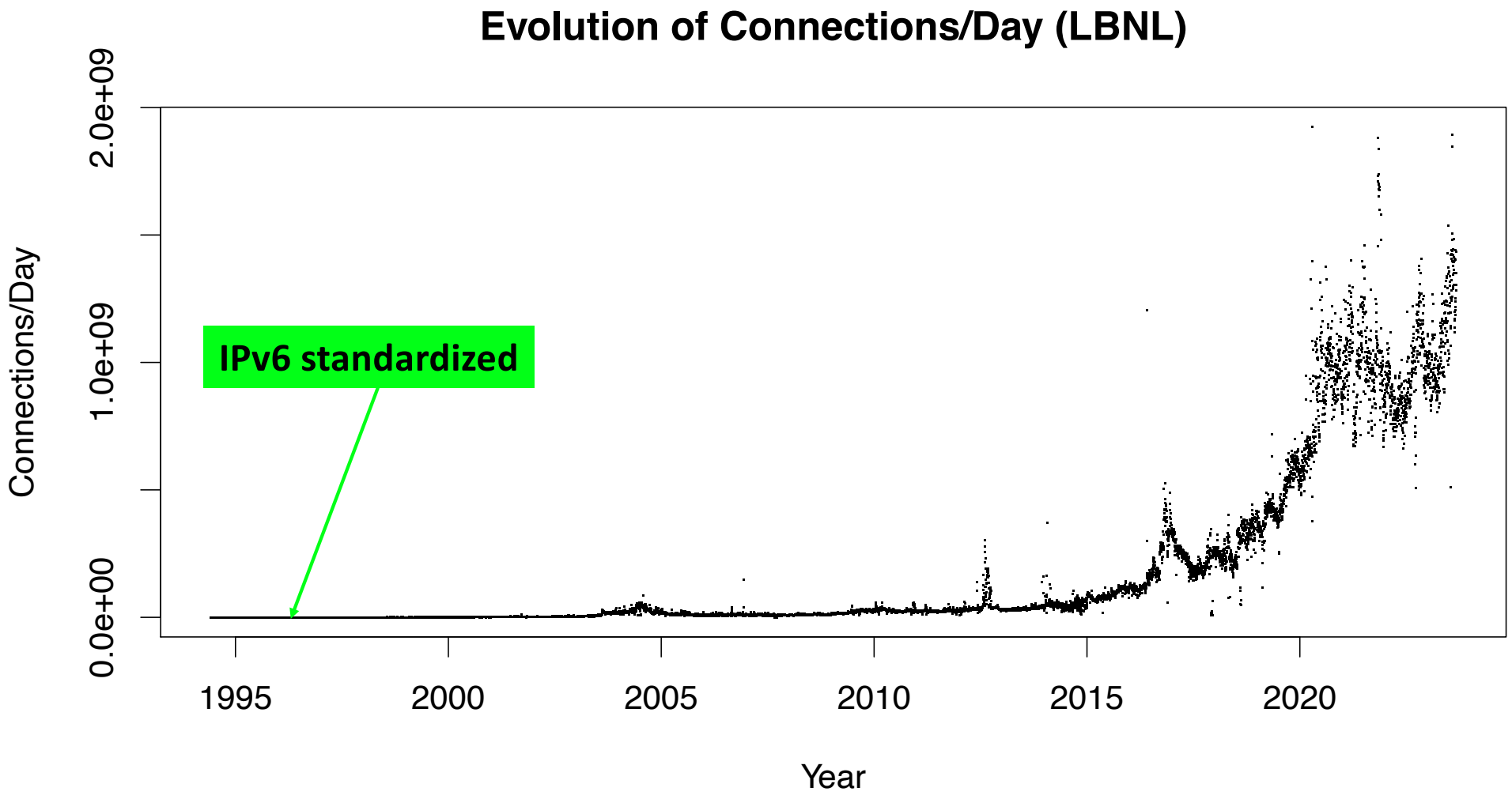


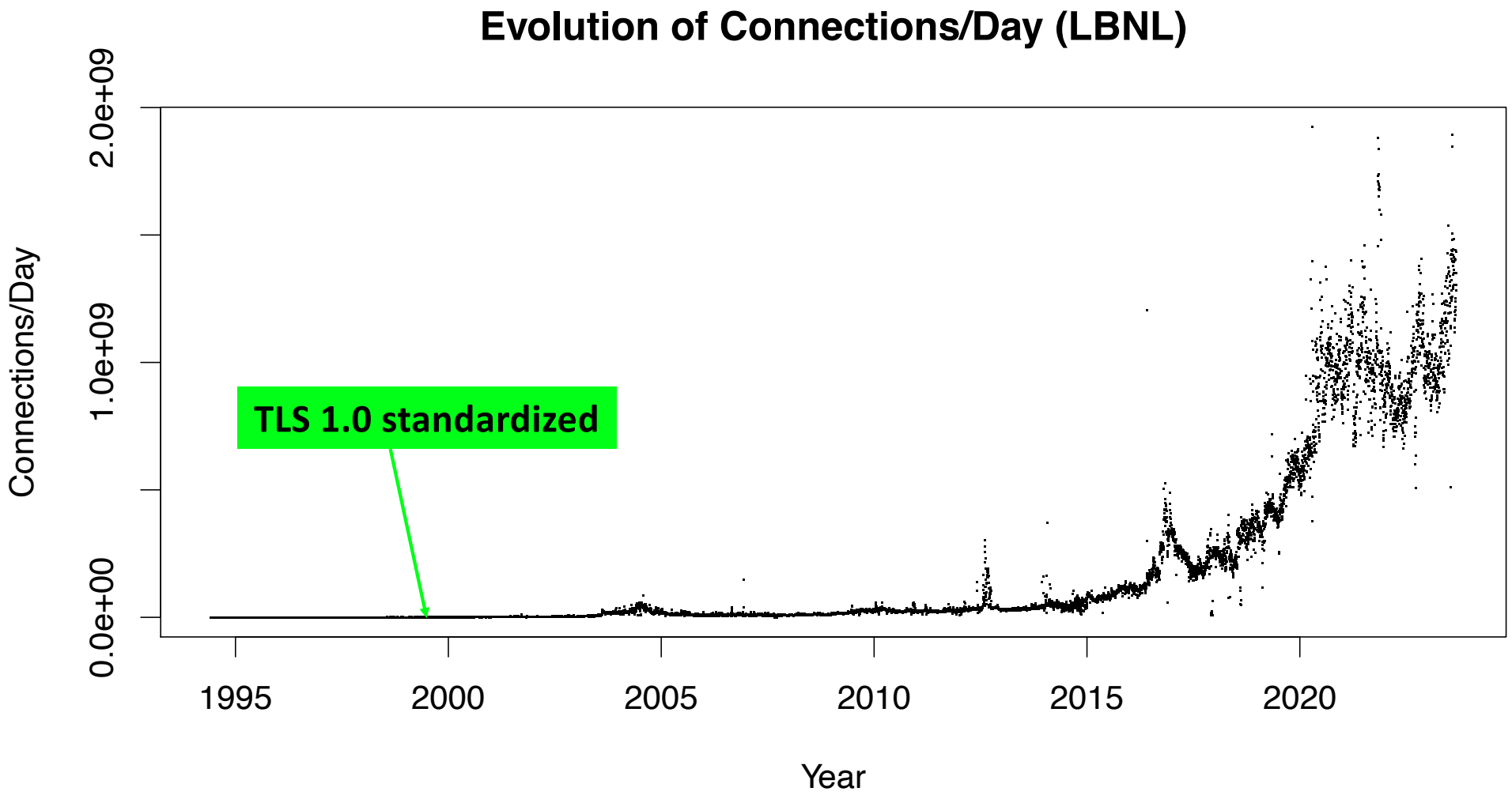
HTTP / (HTTP + TLS)

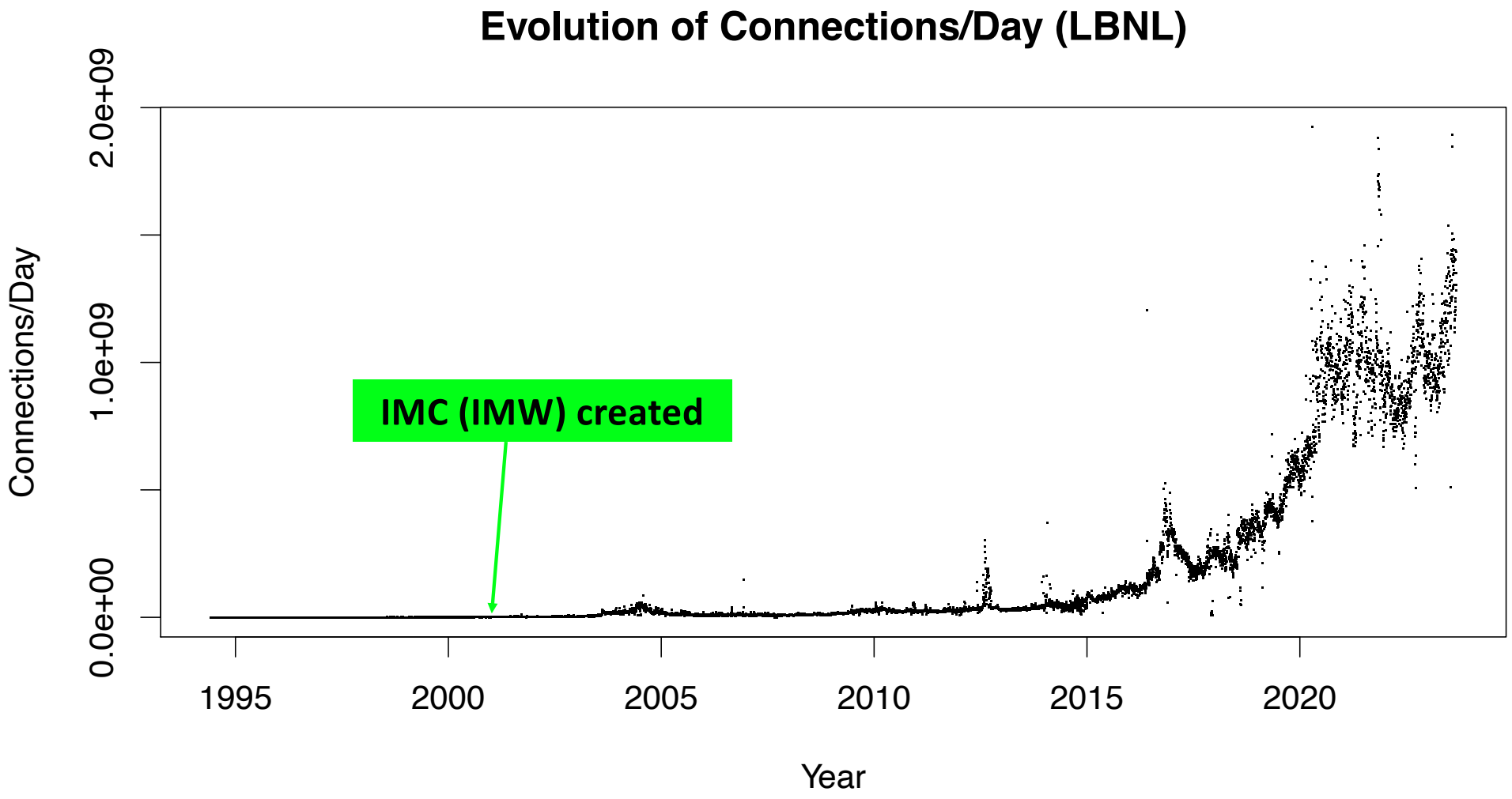

HTTP:TLS Ratio for Individual Corelight Sensors
-0.5%/year
% HTTP
0
20
40
60
80
100
2017
2018
2019
2020
2021
2022
2023
Year

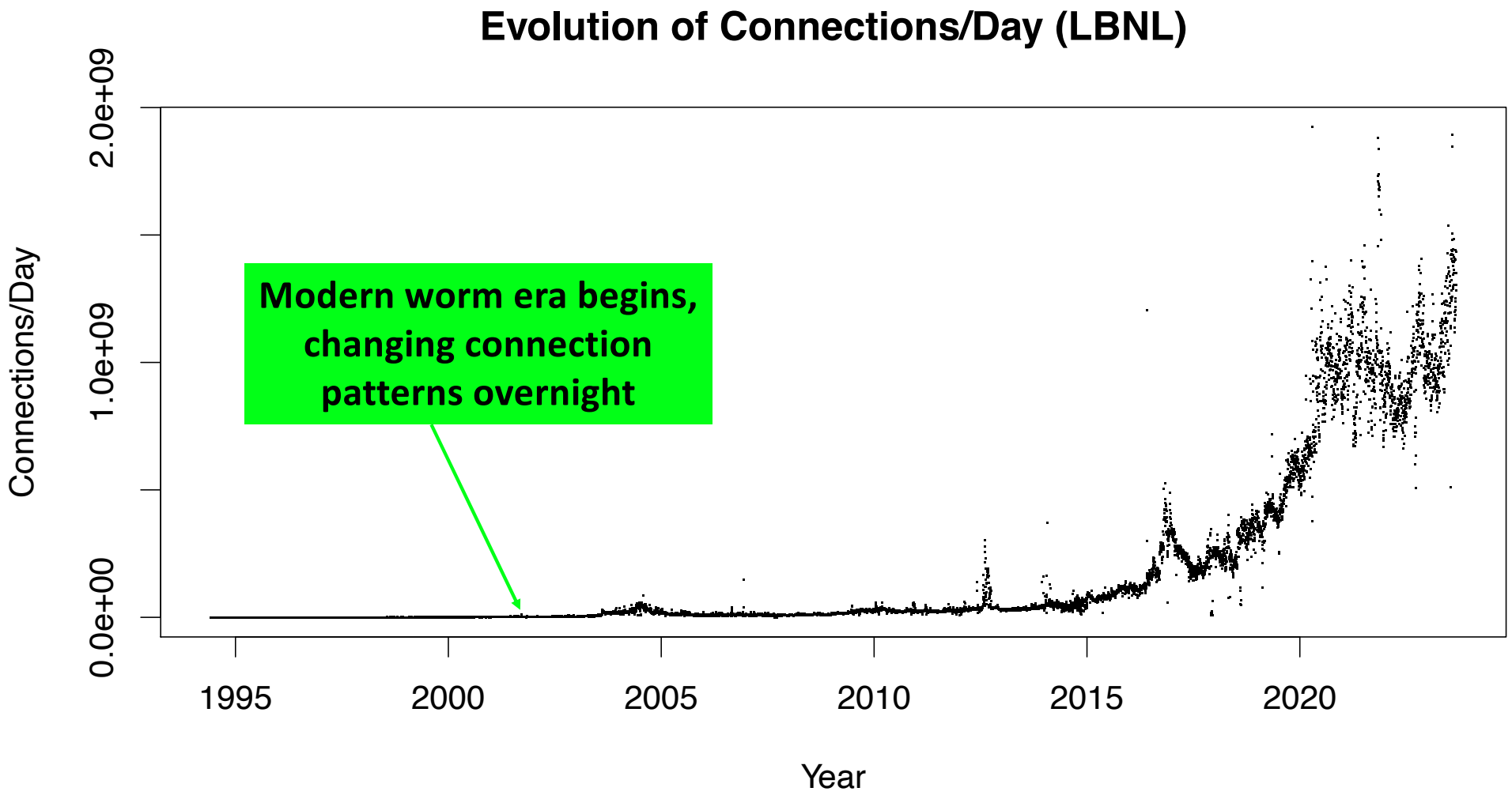
The Future is Already Here …

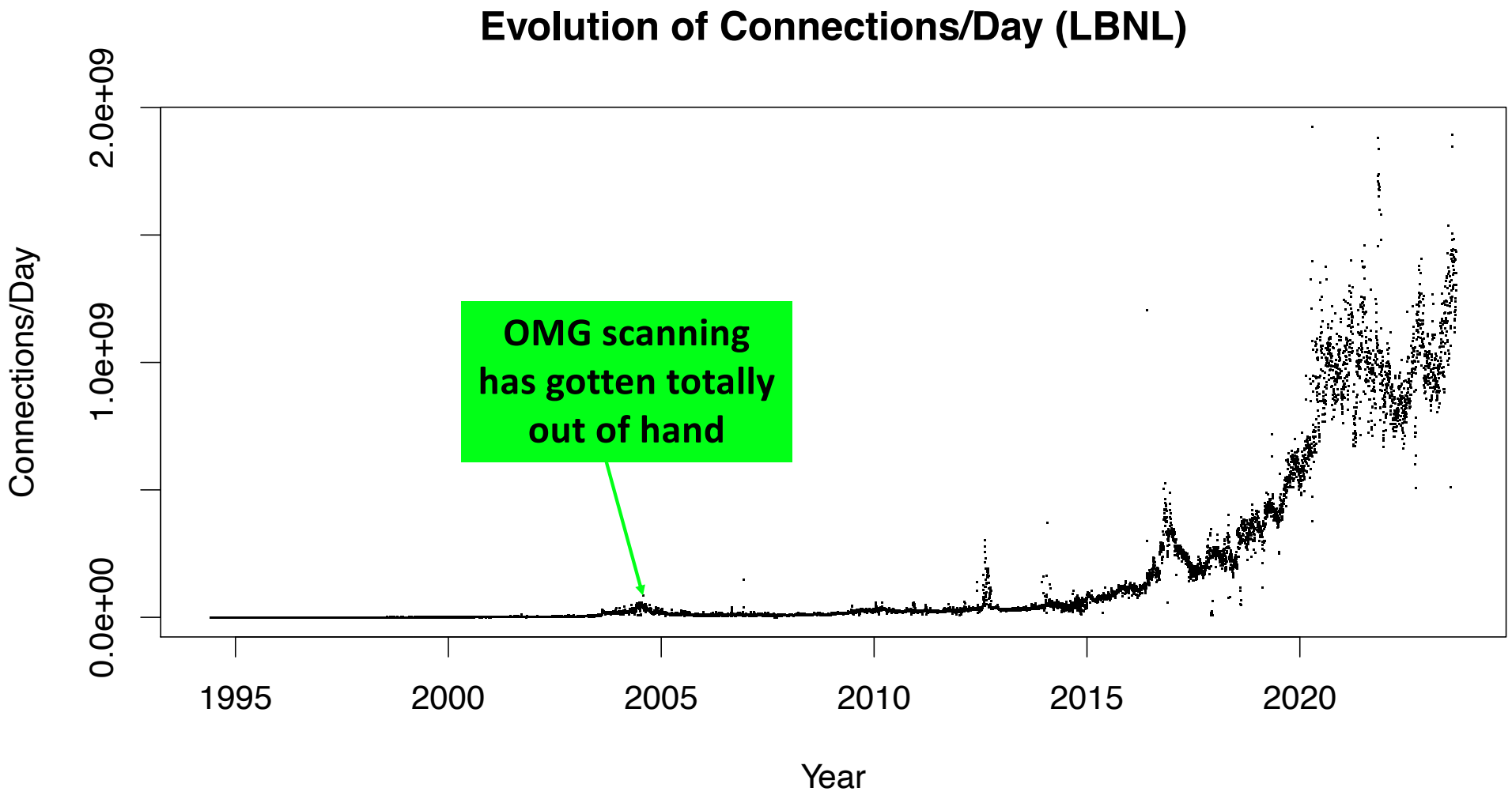

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
The Internet is already
growing exponentially

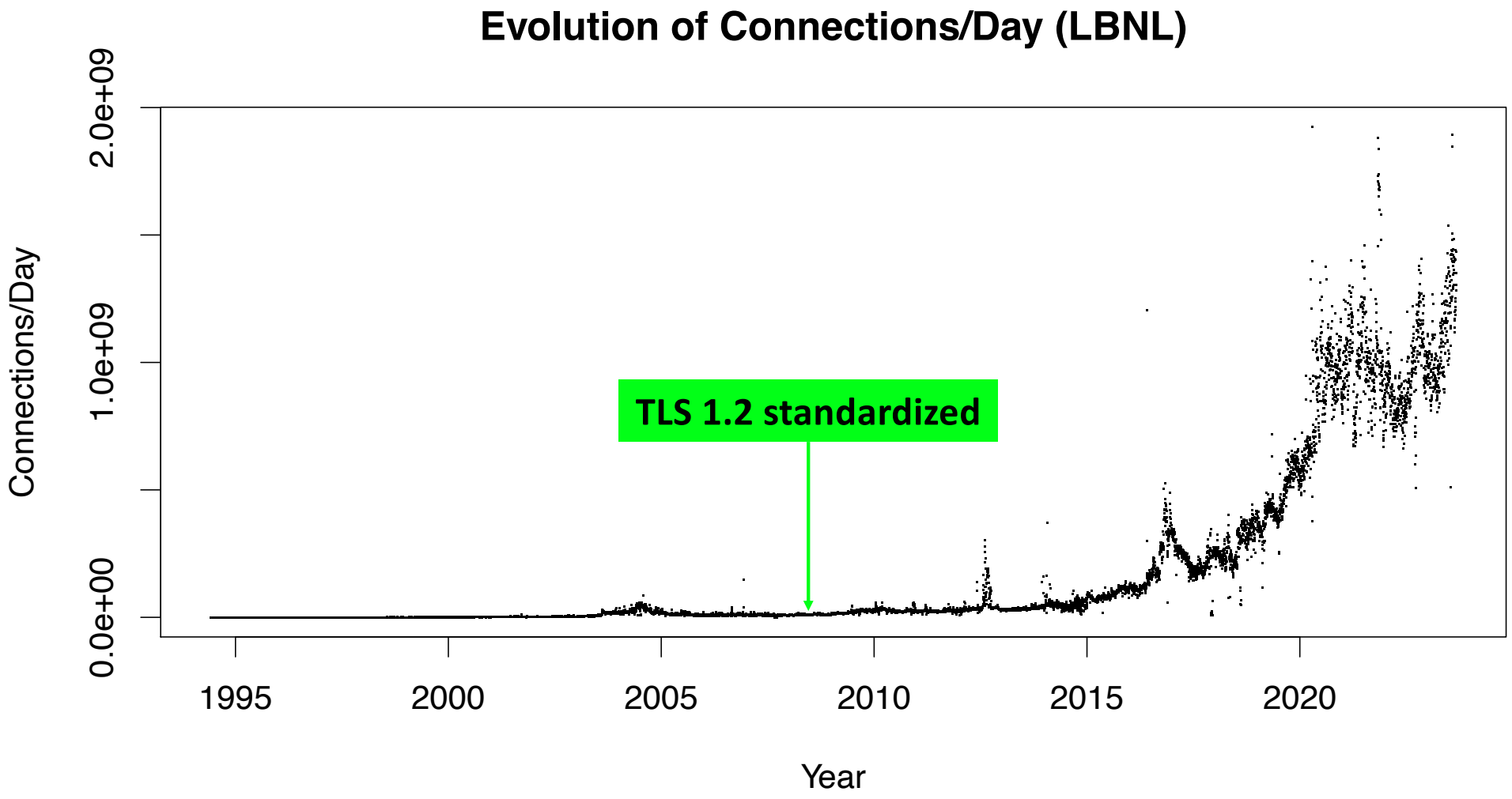

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
IPv6 standardized
1995
2000
2005
2010
2015
2020
Year

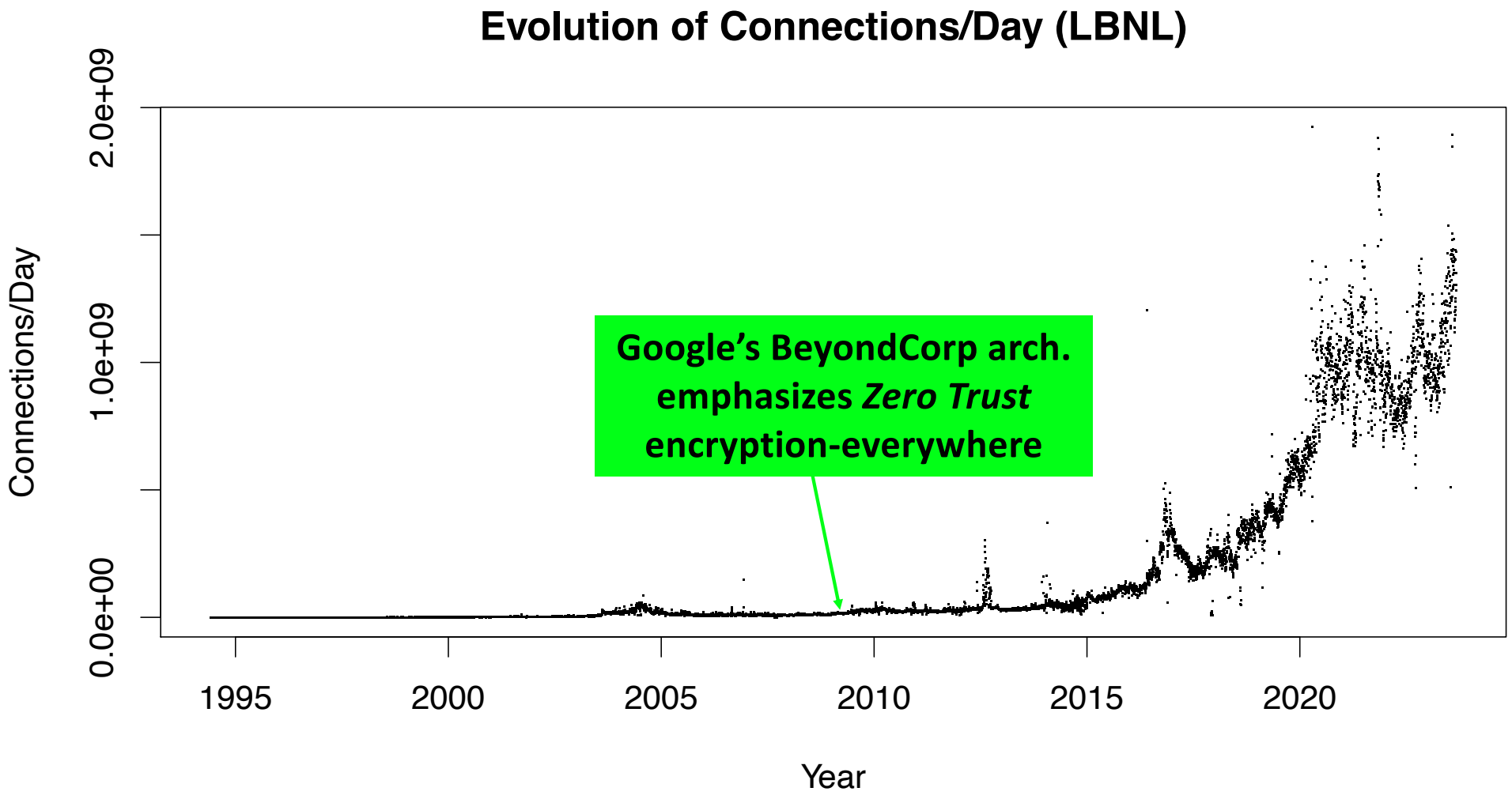

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
TLS 1.0 standardized
1995
2000
2005
2010
2015
2020
Year

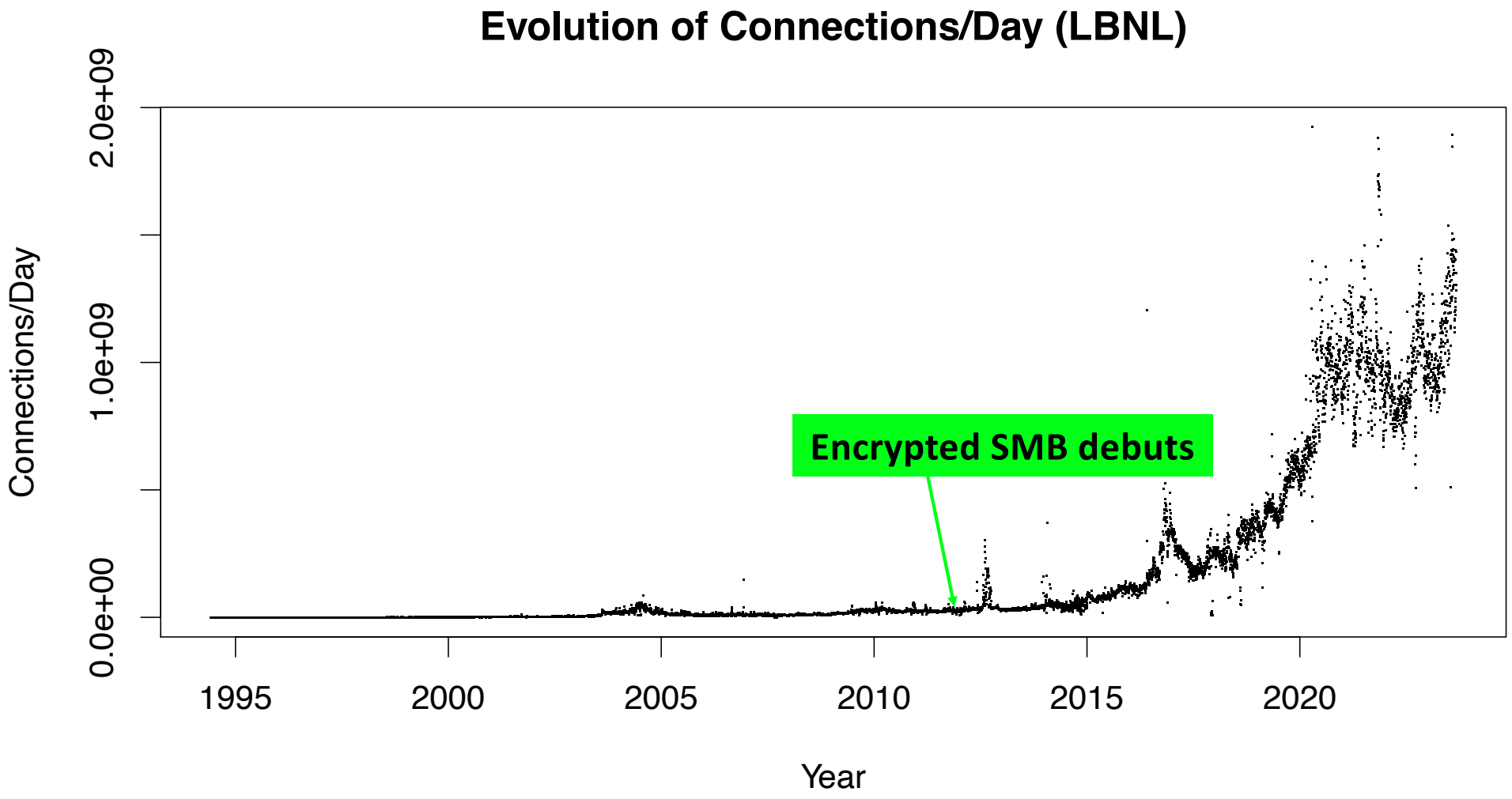

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
IMC (IMW) created
1995
2000
2005
2010
2015
2020
Year

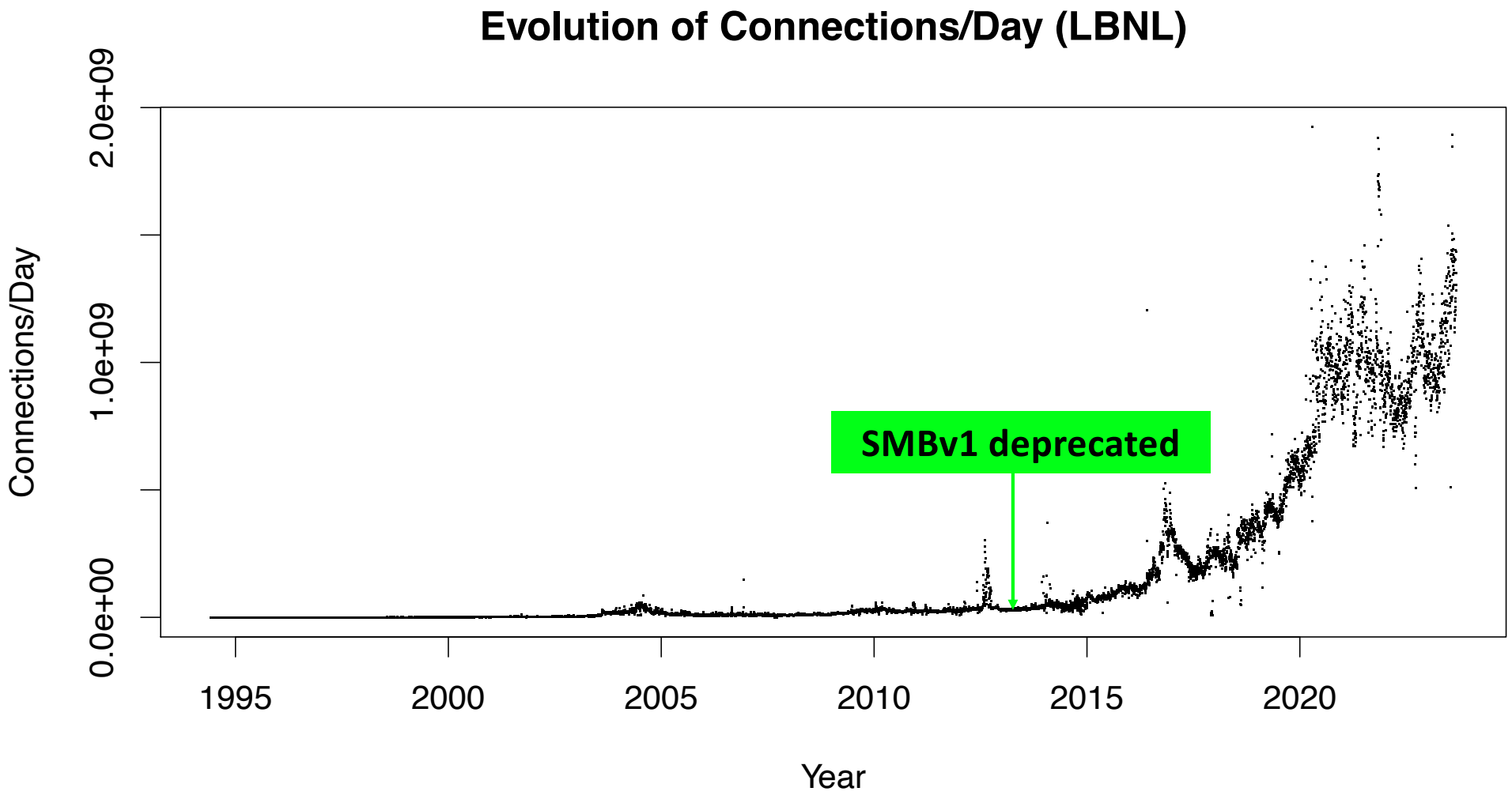

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
Modern worm era begins, changing connection patterns overnight
1995
2000
2005
2010
2015
2020
Year

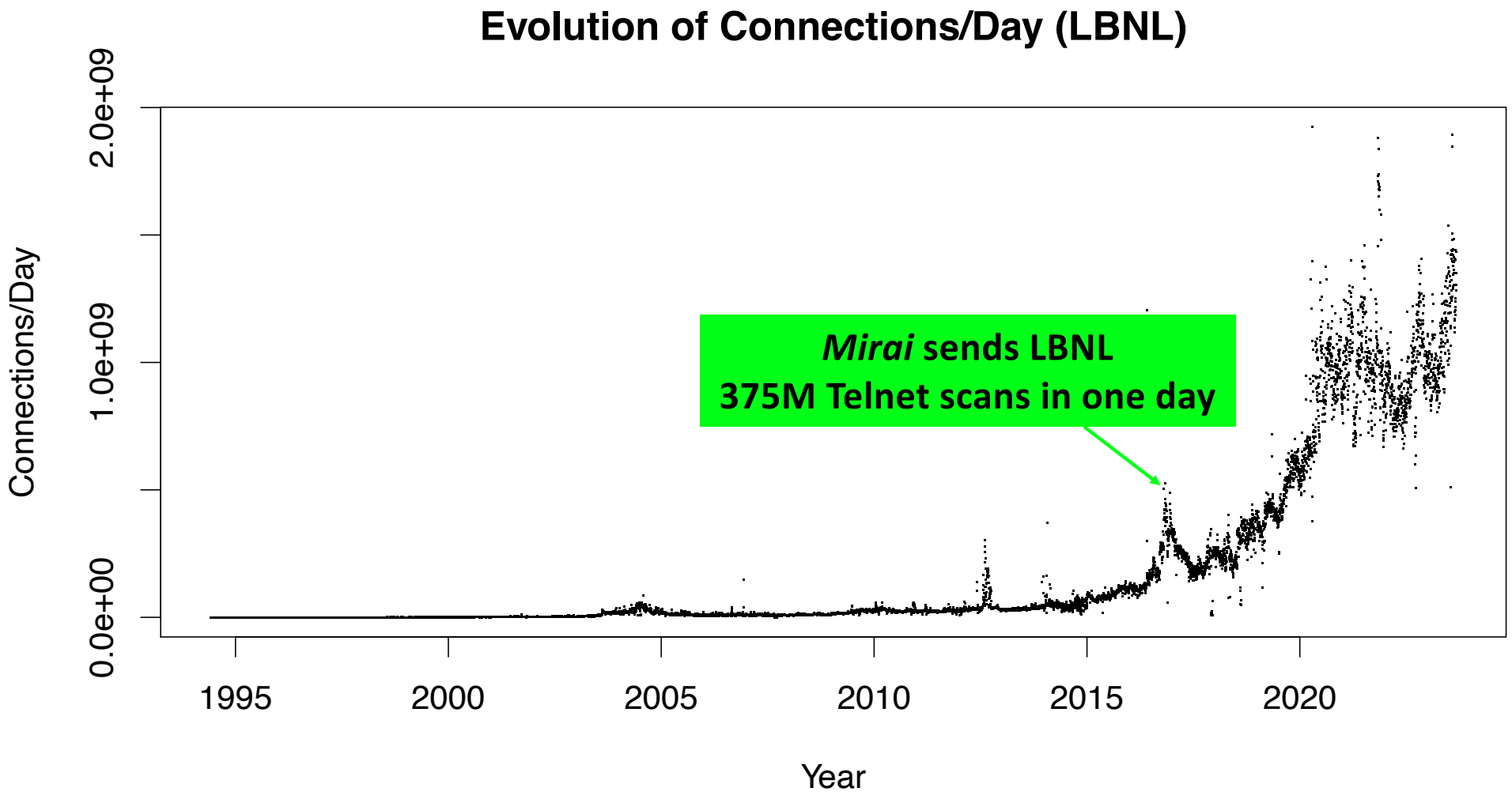

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
OMG scanning
has gotten totally
out of hand
1995
2000
2005
2010
2015
2020
Year

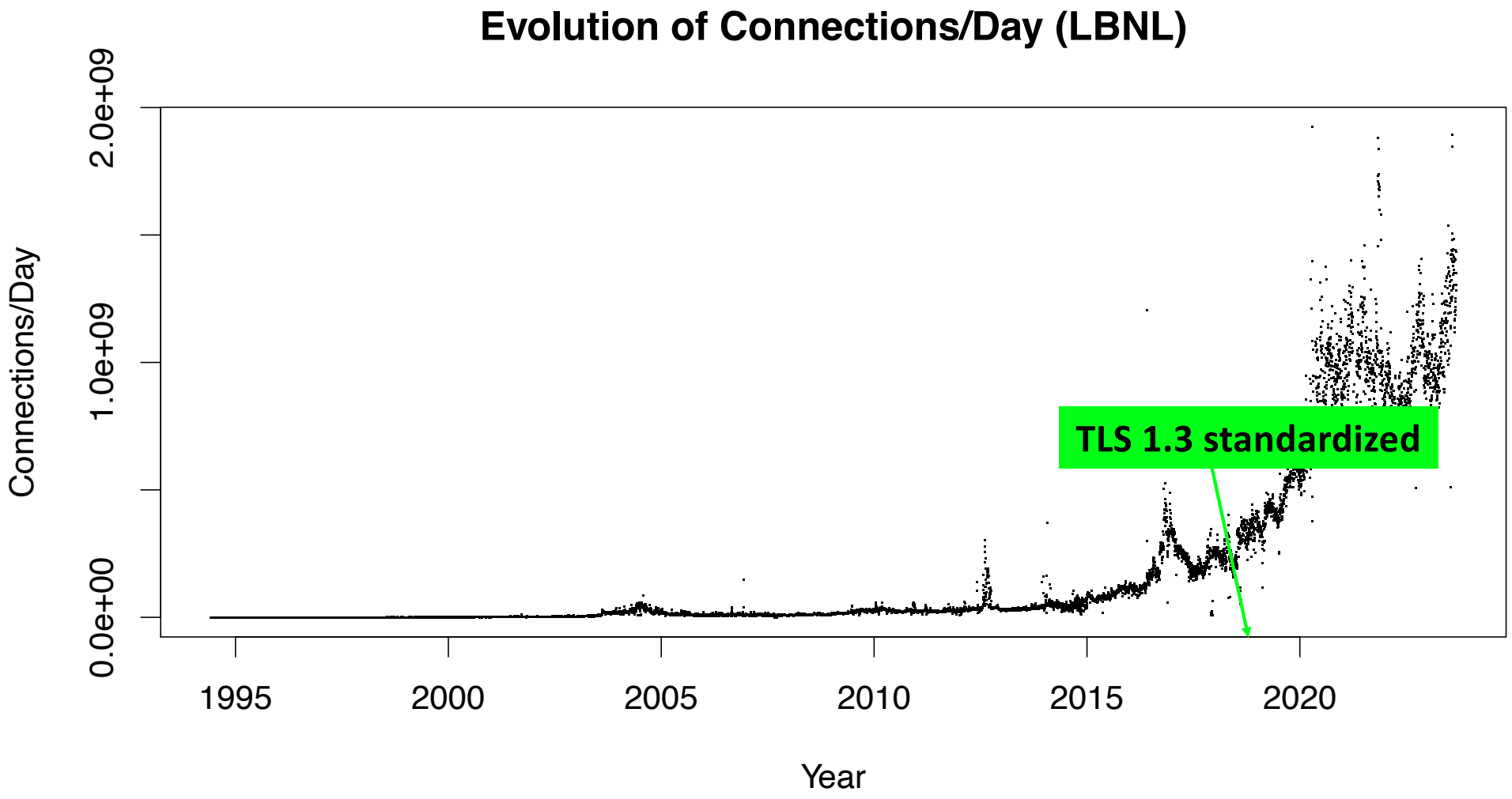

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
TLS 1.2 standardized
1995
2000
2005
2010
2015
2020
Year

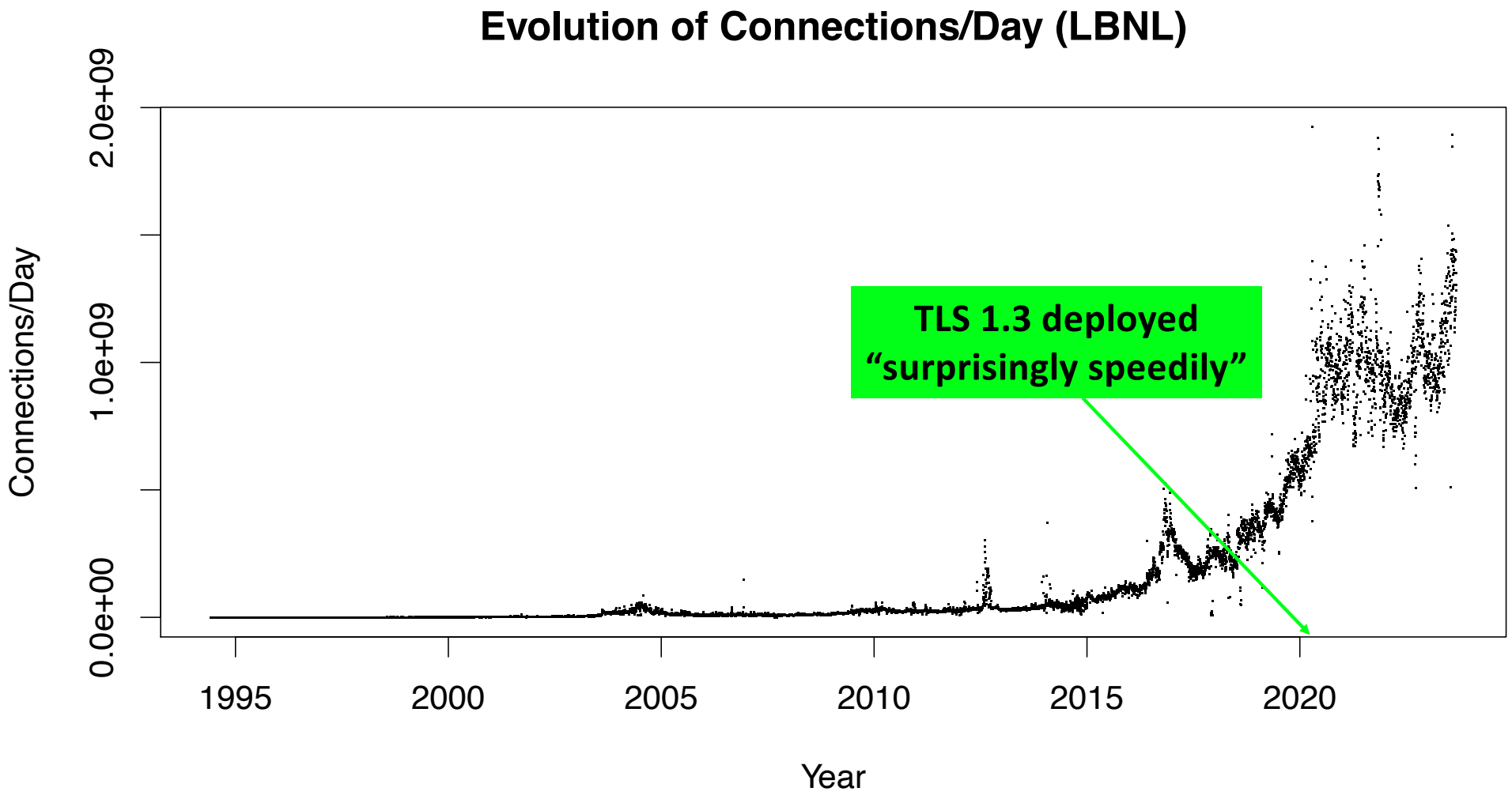

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
Google's BeyondCorp arch.
emphasizes *Zero Trust*
encryption-everywhere

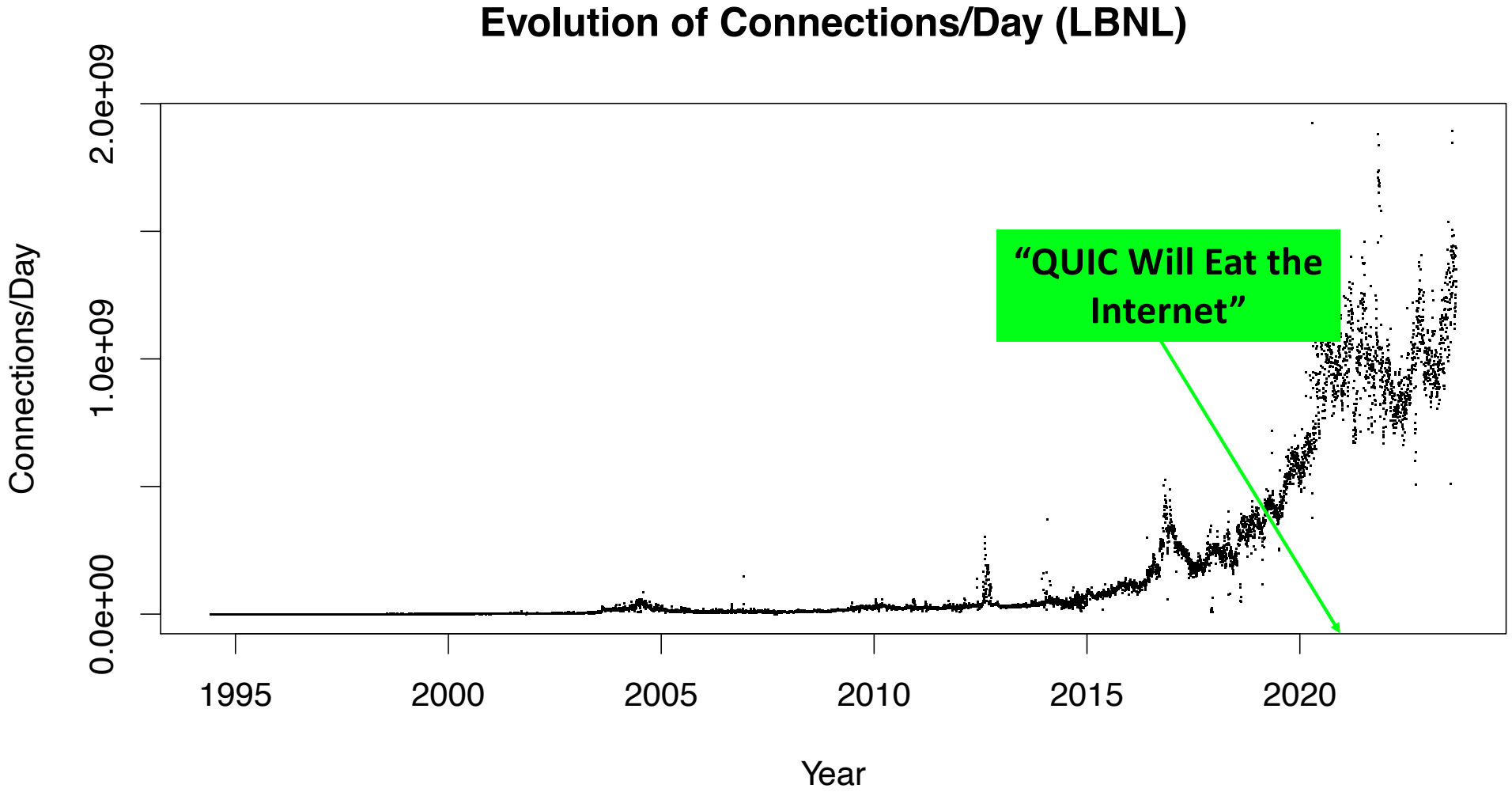

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
Encrypted SMB debuts

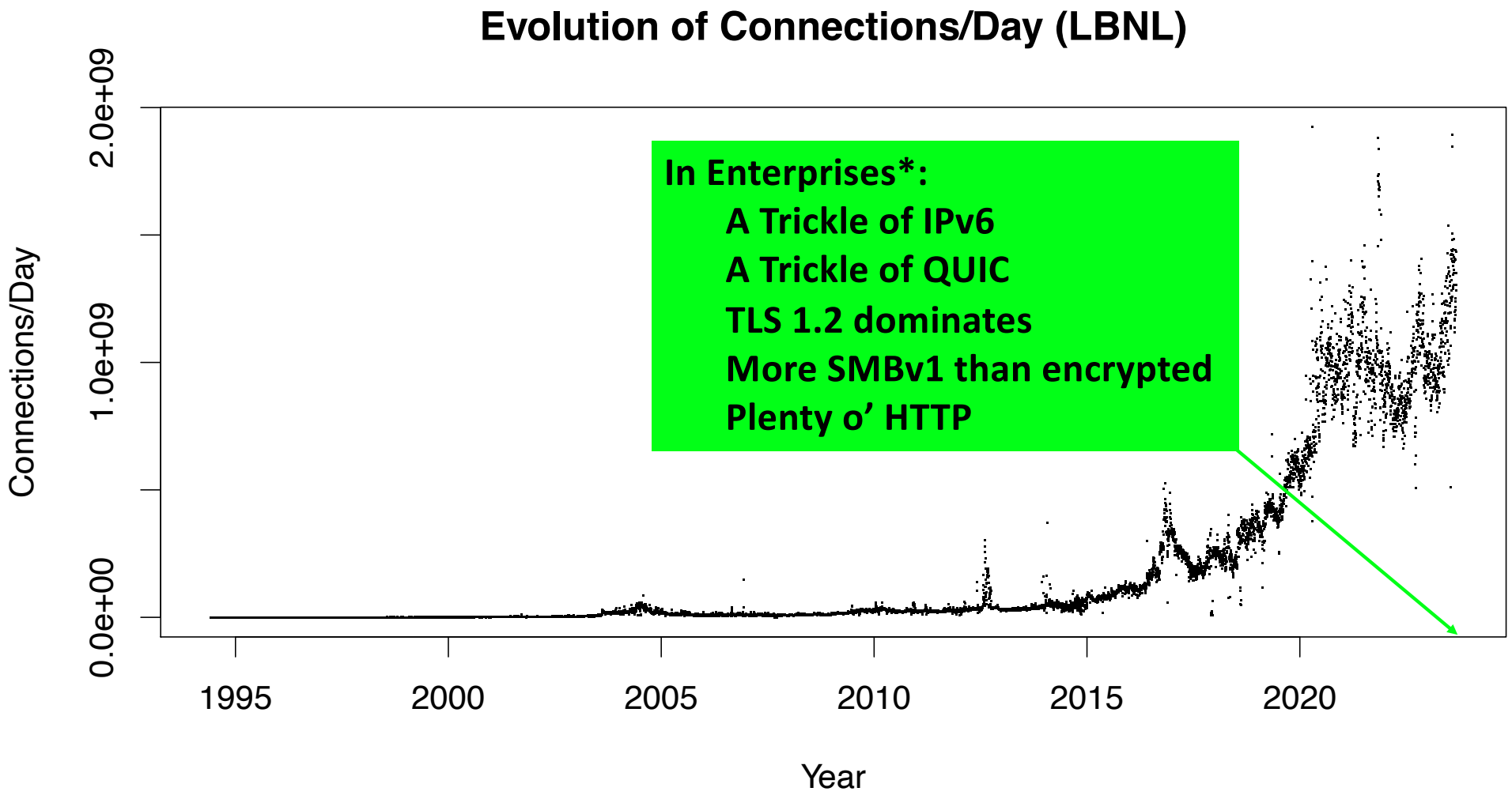

Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
SMBv1 deprecated
1995
2000
2005
2010
2015
2020
Year


Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
Mirai sends LBNL
375M Telnet scans in one day


Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
TLS 1.3 standardized


Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
TLS 1.3 deployed
“surprisingly speedily”


Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
“QUIC Will Eat the Internet”


Evolution of Connections/Day (LBNL)
Connections/Day
0.0e+00
1.0e+09
2.0e+09
1995
2000
2005
2010
2015
2020
Year
In Enterprises*:
A Trickle of IPv6
A Trickle of QUIC
TLS 1.2 dominates
More SMBv1 than encrypted
Plenty o' HTTP

ENTERPRISES, WILLIAM GIBSON & THE ART OF INTERNET MEASUREMENT

Vern Paxson, Co-Founder and Chief Scientist, Corelight Inc.

Professor of the Graduate School, UC Berkeley

vern@corelight.com