31st NMRG Meeting

REVEALING MIDDLEBOXES INTERFERENCE WITH TRACEBOX

Presented by: Fabien Duchêne*

Gregory Detal*, Benjamin Hesmans*, Olivier Bonaventure*, Yves Vanaubel° and Benoit Donnet°.

*Université Catholique de Louvain
°Université de Liège

http://www.tracebox.org

Outline

- **Middleboxes interference**
- Detect packet modifications with ICMP
- Measurements results
- Tracebox

The end-to-end principle …

The end-to-end principle …

Application
Transport
Network
Data link
Physical

Application
Transport
Network
Data link
Physical

The end-to-end principle …

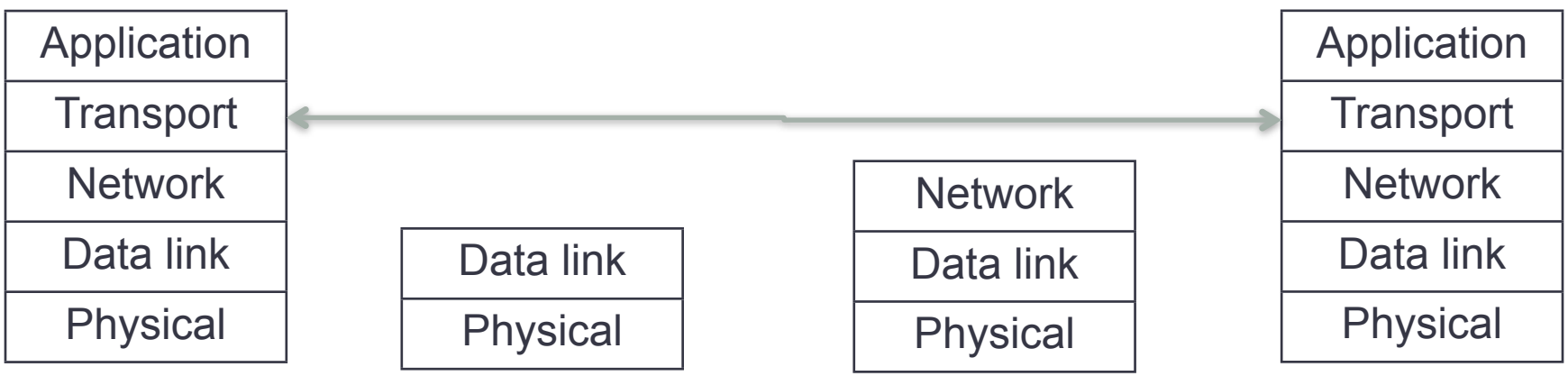
Application
Transport
Network
Data link
Physical

Data link
Physical

Network
Data link
Physical

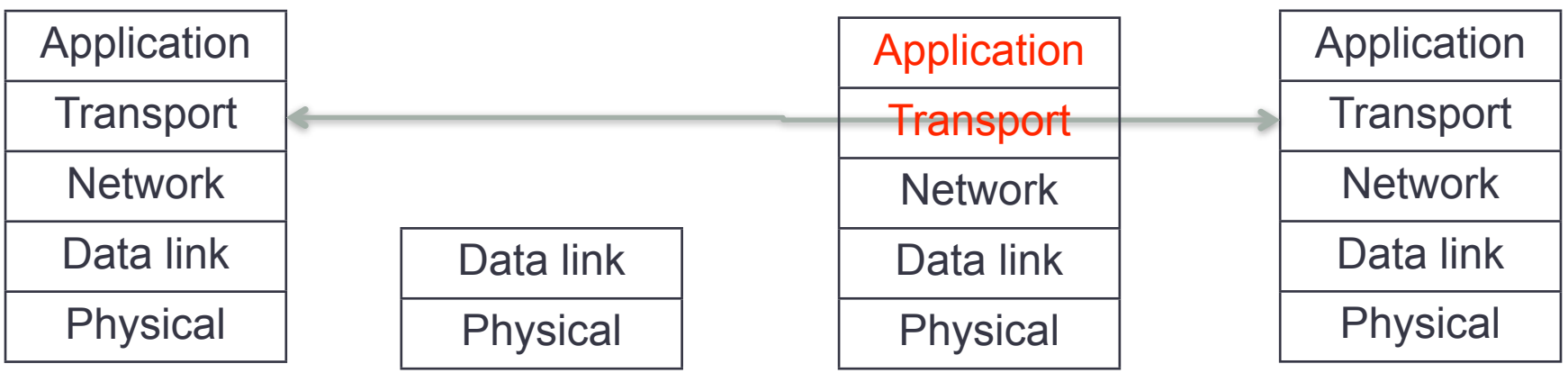
Application
Transport
Network
Data link
Physical

The end-to-end principle …

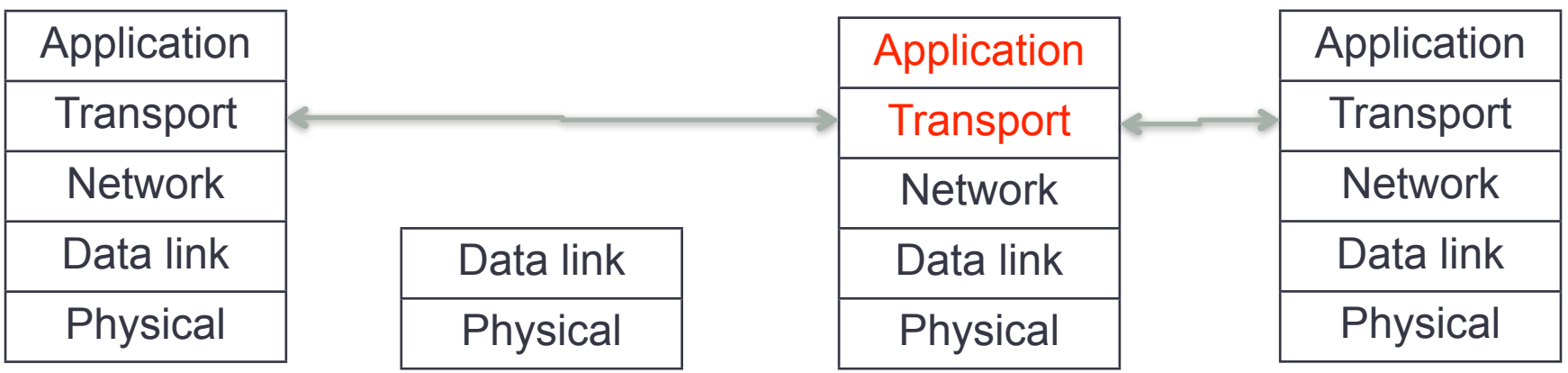


… does not hold ☹

… does not hold ☹



… does not hold ☹



How transparent is the Internet ?

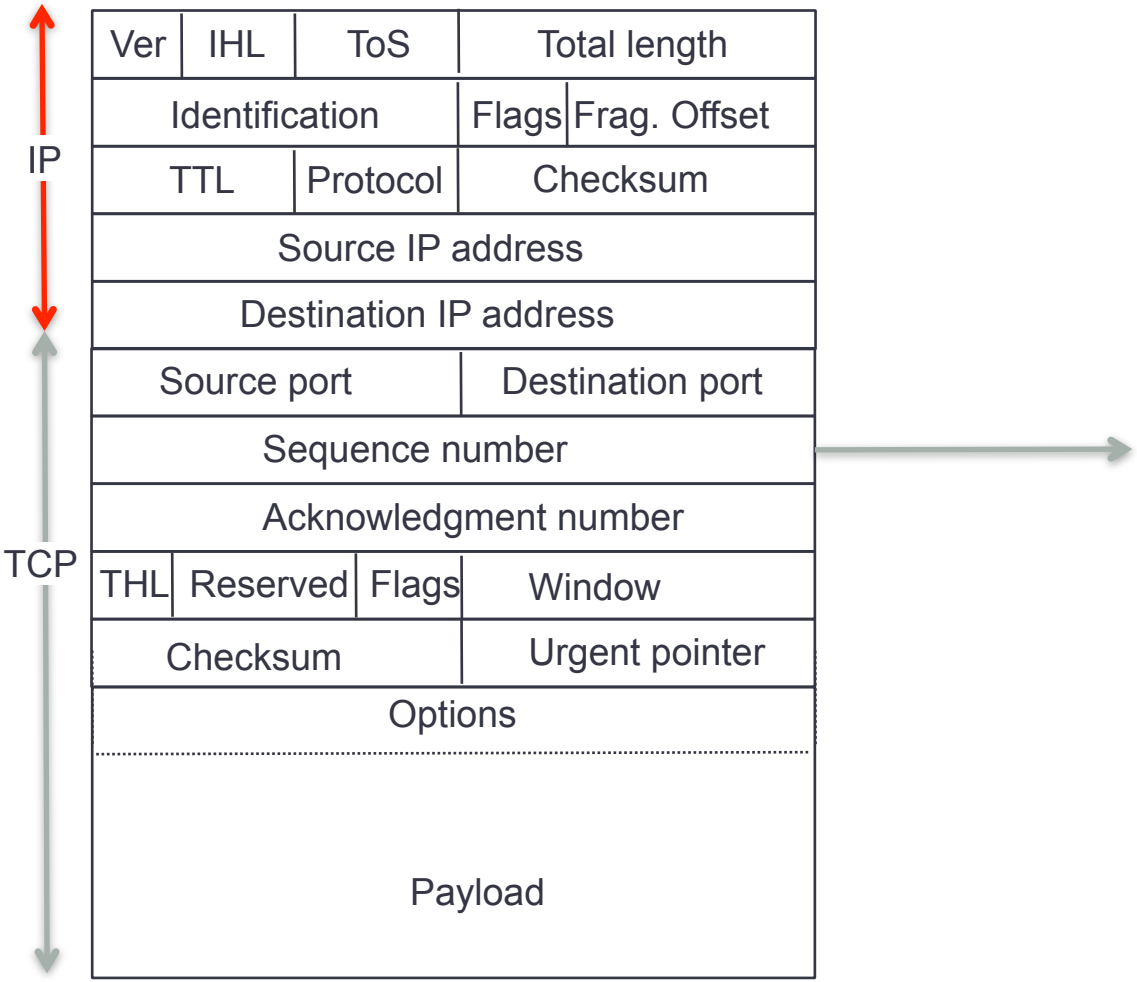
- 25th September 2010 to 30th April 2011
- 142 access networks
- 24 countries
- Craft TCP segments using custom scripts
- Sent specific TCP segments from client to a server in Japan

Table 2: Experiment Venues

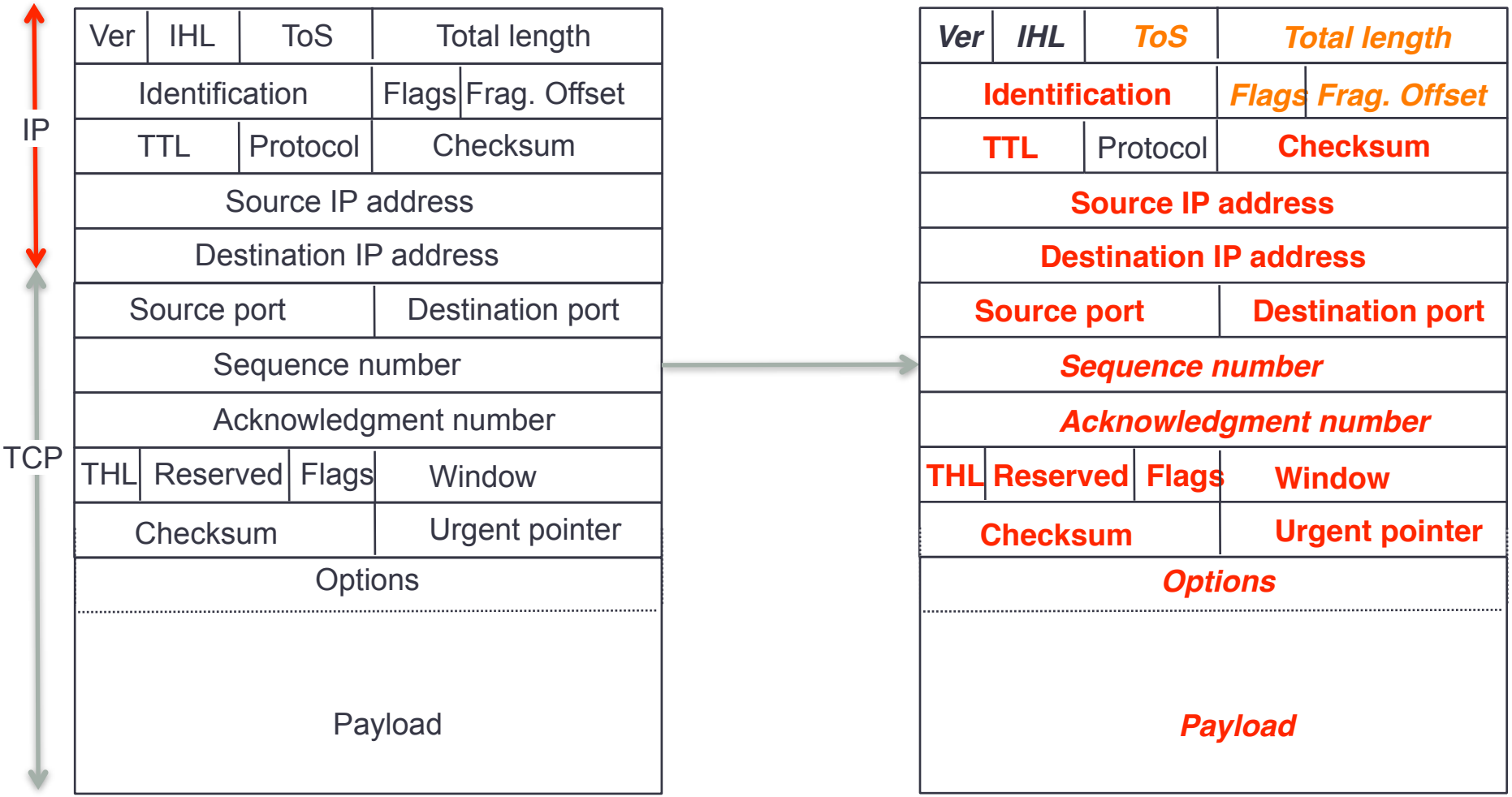
Country	*Home*	*Hotspot*	*Cellular*	*Univ*	*Ent*	*Hosting*	Total
Australia	0	2	0	0	0	1	3
Austria	0	0	0	0	1	0	1
Belgium	4	0	0	1	0	0	5
Canada	1	0	1	0	1	0	3
Chile	0	0	0	0	1	0	1
China	0	7	0	0	0	0	7
Czech	0	2	0	0	0	0	2
Denmark	0	2	0	0	0	0	2
Finland	1	0	0	3	2	0	6
Germany	3	1	3	4	1	0	12
Greece	2	0	1	0	0	0	3
Indonesia	0	0	0	3	0	0	3
Ireland	0	0	0	0	0	1	1
Italy	1	0	0	0	1	0	2
Japan	19	10	7	3	2	0	41
Romania	1	0	0	0	0	0	1
Russia	0	1	0	0	0	0	1
Spain	0	1	0	1	0	0	2
Sweden	1	0	0	0	0	0	1
Switzerland	2	0	0	0	0	0	2
Thailand	0	0	0	0	2	0	2
U.K.	10	4	4	2	1	1	22
U.S.	3	4	4	0	4	2	17
Vietnam	1	0	0	0	1	0	2
Total	49	34	20	17	17	5	142

Honda, Michio, et al. "*Is it still possible to extend TCP?*" Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. ACM, 2011.

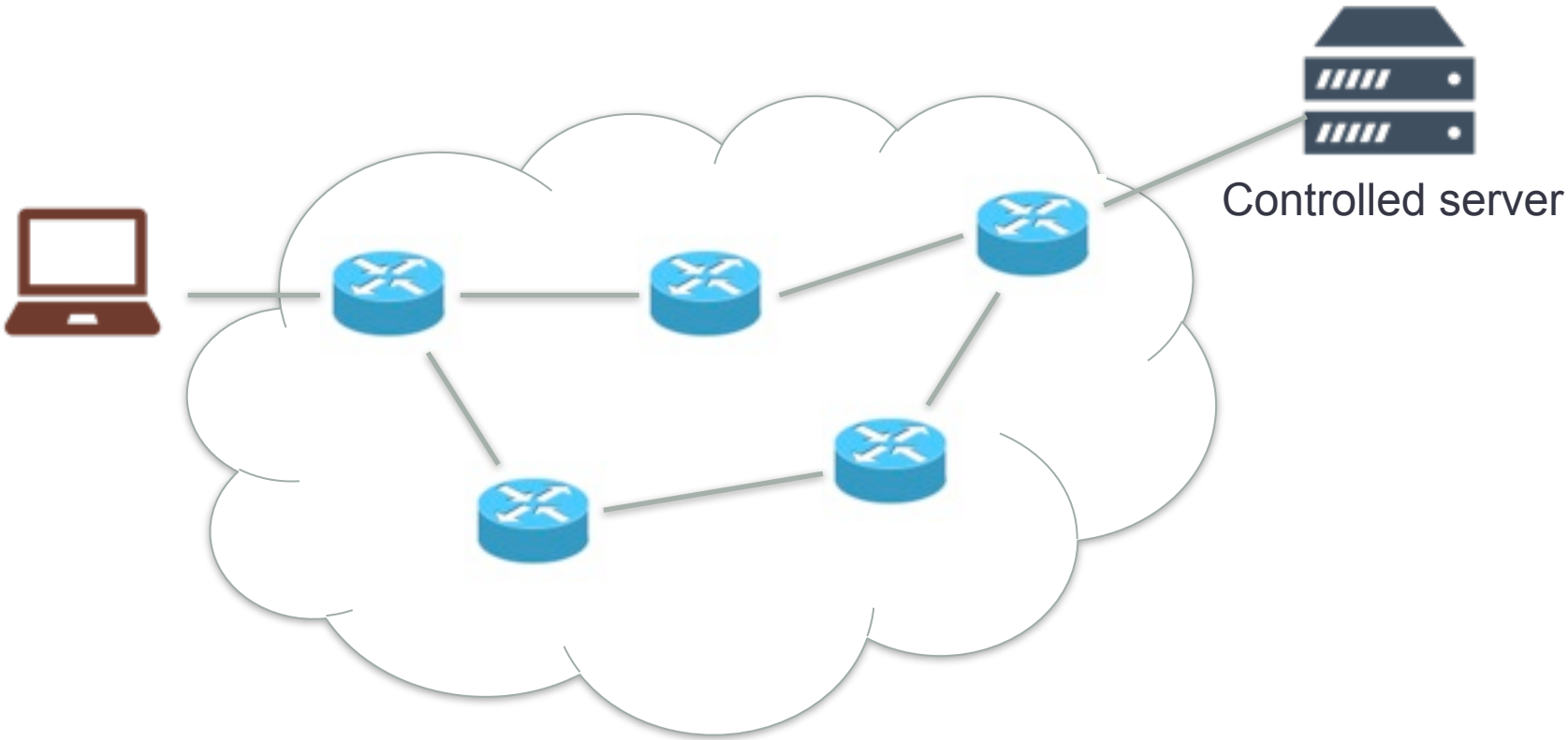
TCP Segments on the today's Internet



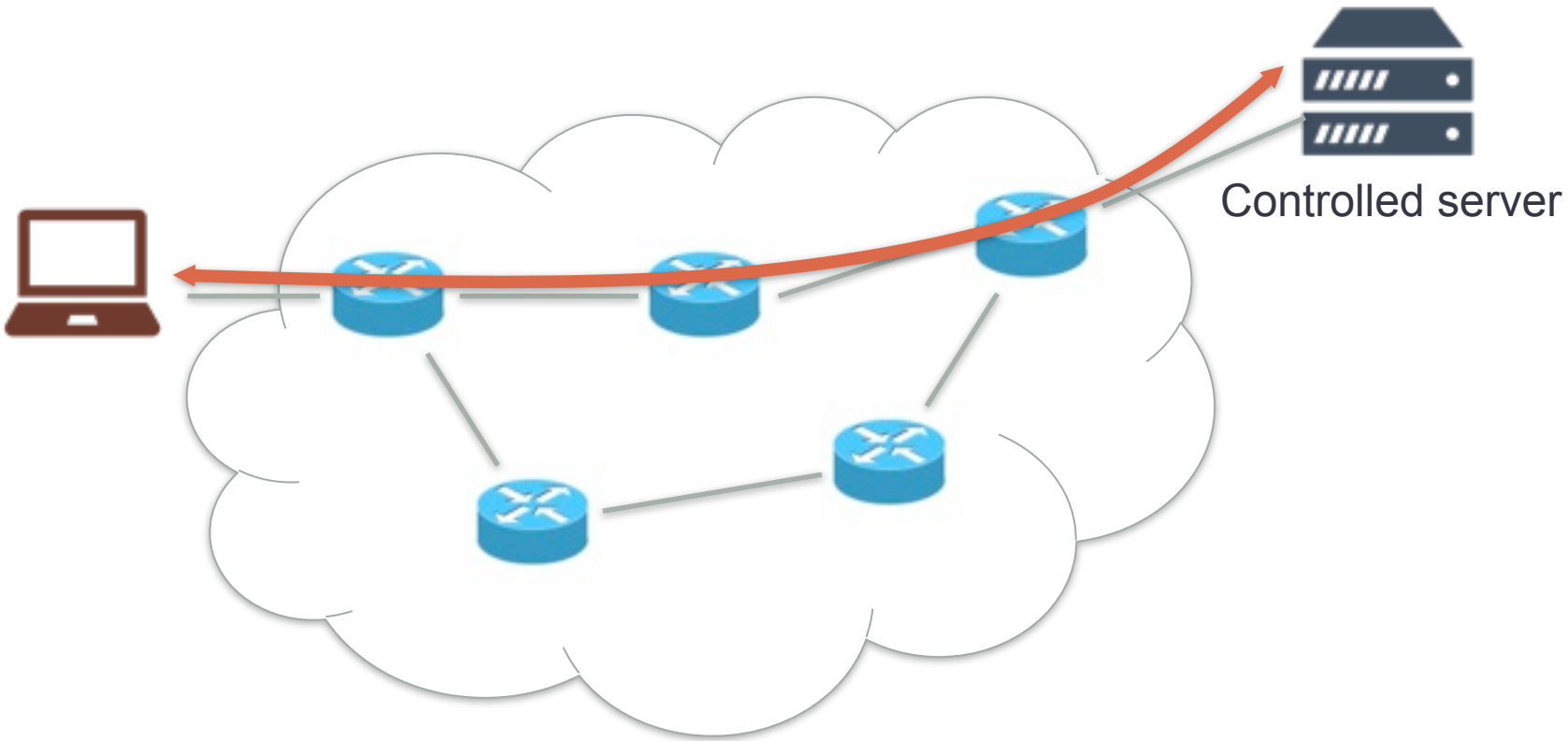
TCP Segments on the today's Internet



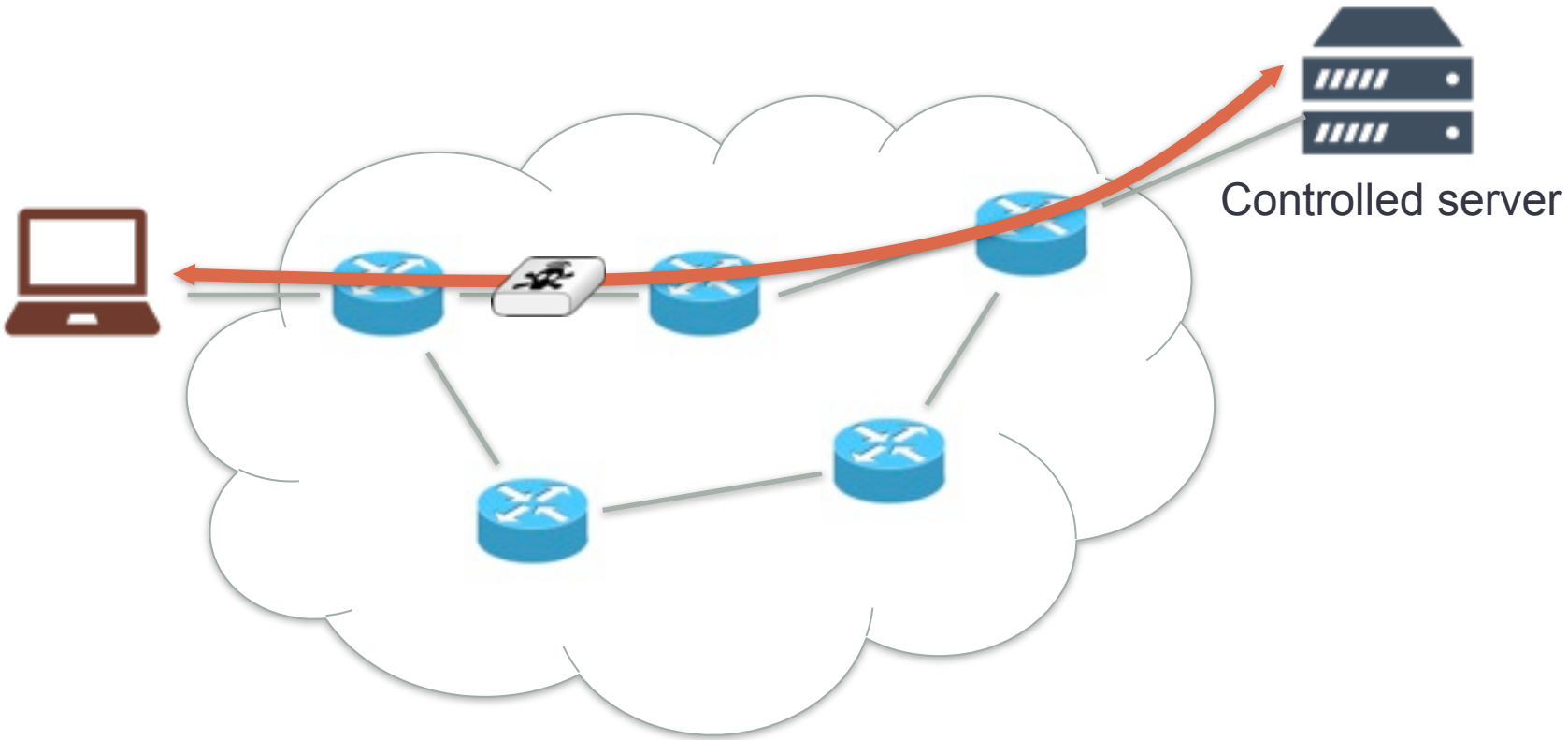
Controlling a server allows to detect middleboxes on one path



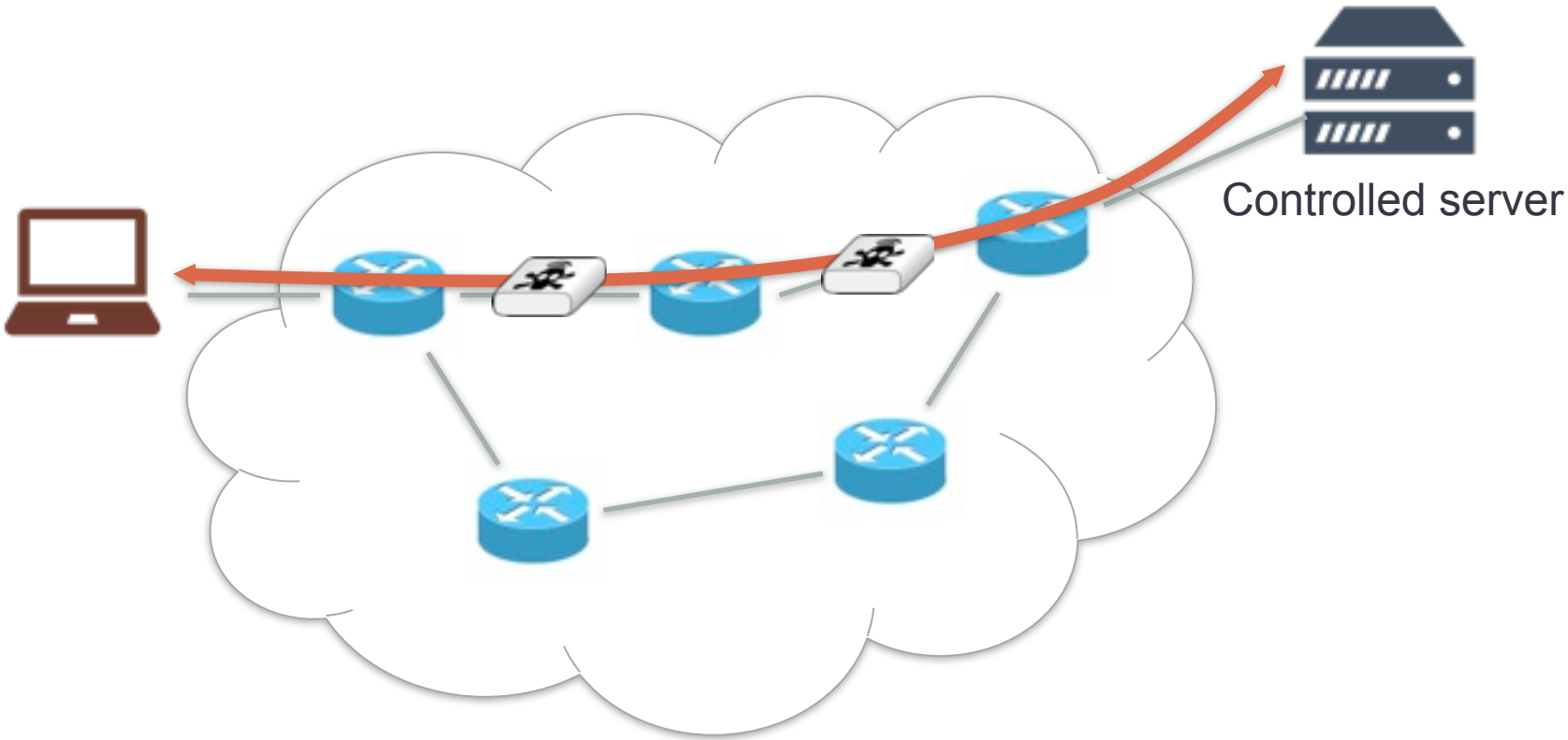
Controlling a server allows to detect middleboxes on one path



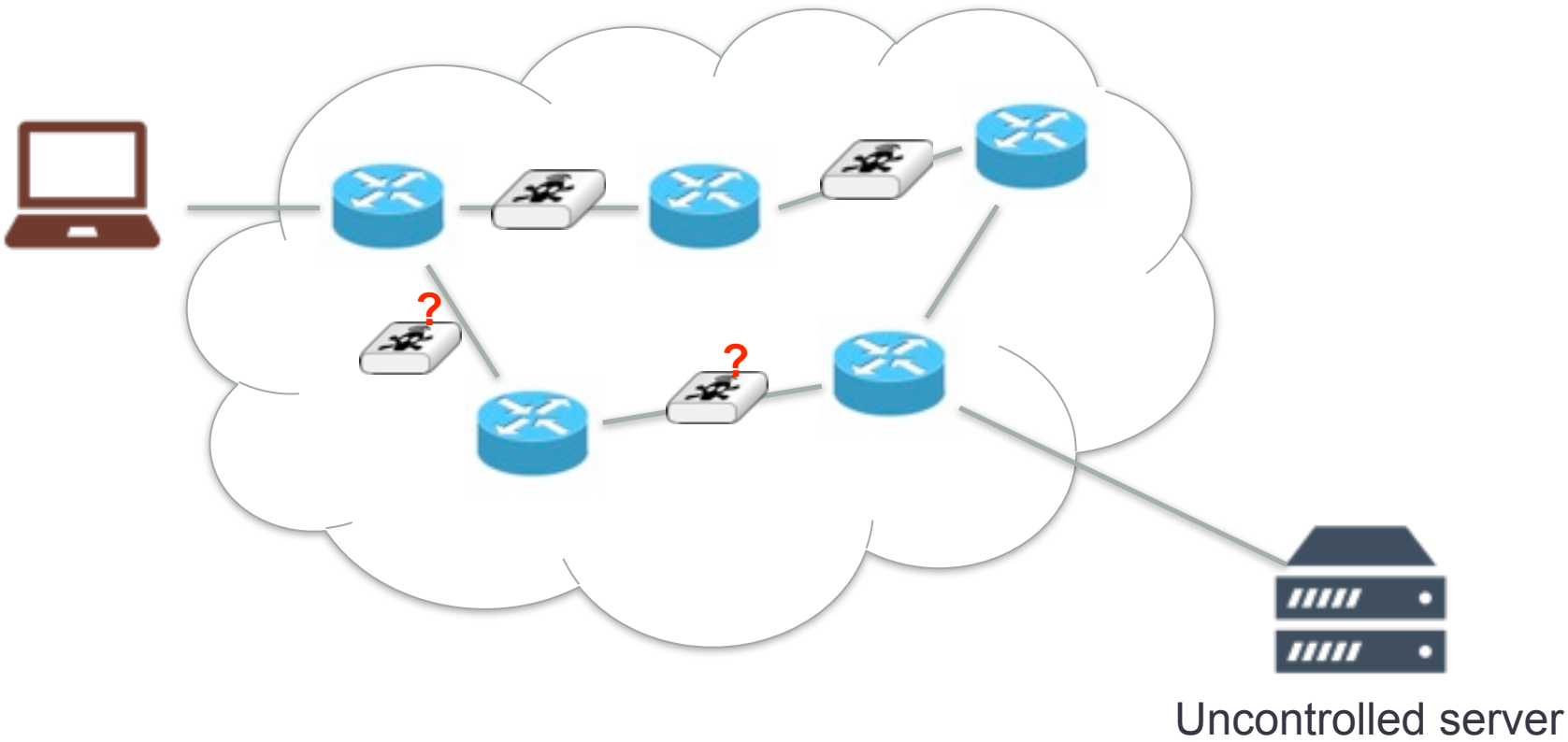
Controlling a server allows to detect middleboxes on one path



Controlling a server allows to detect middleboxes on one path



Potentially miss a lot of middleboxes



Motivation

Motivation

- Detecting middleboxes can help:
  - Understanding performances
  - Validate new protocols
  - …

Motivation

- Detecting middleboxes can help:
  - Understanding performances
  - Validate new protocols
  - …
- ➢ Debug the network !

Motivation

- Detecting middleboxes can help:
  - Understanding performances
  - Validate new protocols
  - …
- ➢ Debug the network !

Motivation

- Detecting middleboxes can help:
  - Understanding performances
  - Validate new protocols
  - …
- ➢ Debug the network !

- How can we detect middleboxes interference without server collaboration ?

Motivation

- Detecting middleboxes can help:
  - Understanding performances
  - Validate new protocols
  - …
- ➢ Debug the network !

- How can we detect middleboxes interference without server collaboration ?

Motivation

- Detecting middleboxes can help:
  - Understanding performances
  - Validate new protocols
  - …
- ➢ Debug the network !

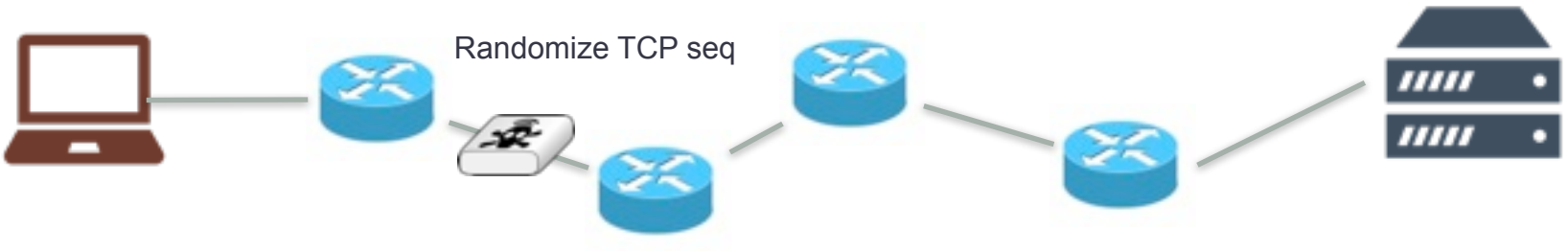
- How can we detect middleboxes interference without server collaboration ?

- How can we localize the middleboxes ?

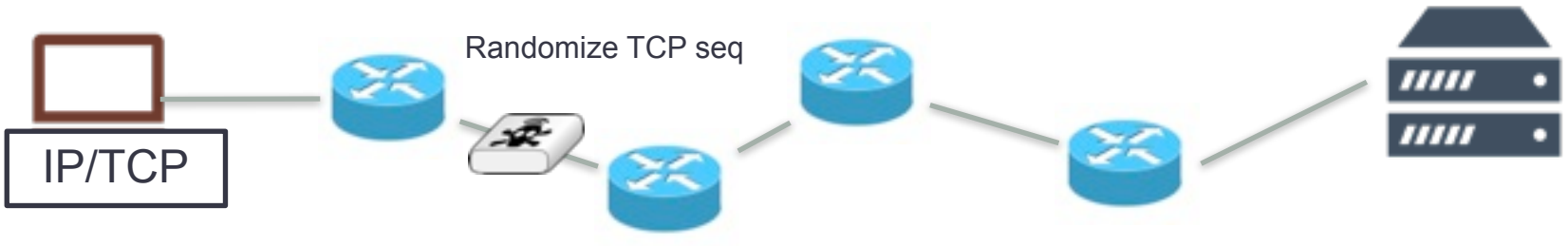
Outline

- Middleboxes interference
- **Detect packet modifications with ICMP**
- Measurements results
- Tracebox

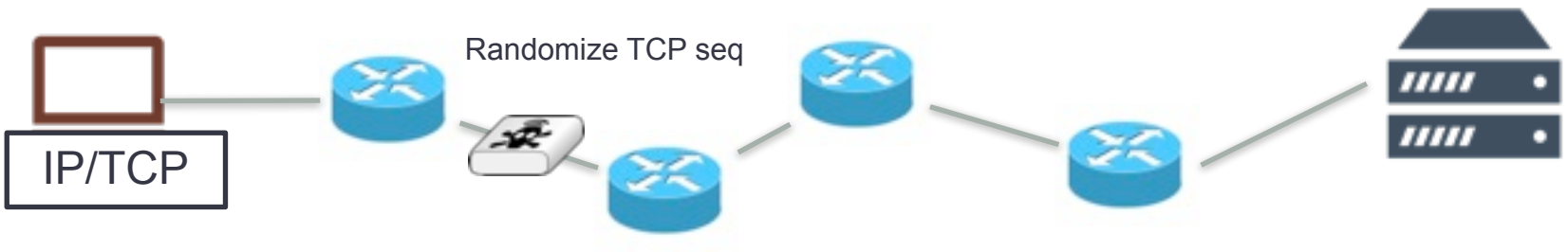
Middlebox detection using ICMP



Middlebox detection using ICMP

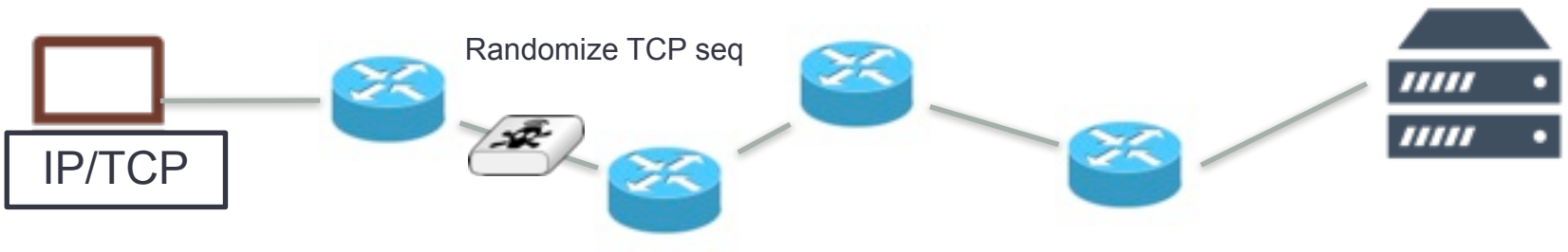


Middlebox detection using ICMP



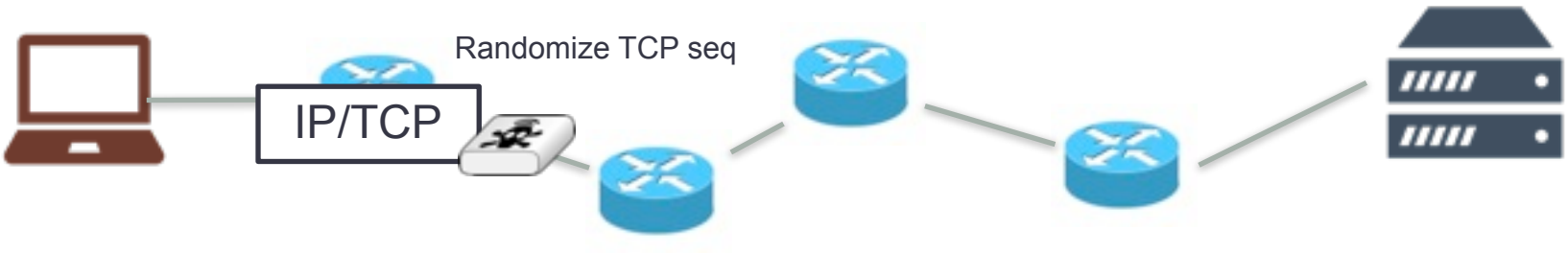
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



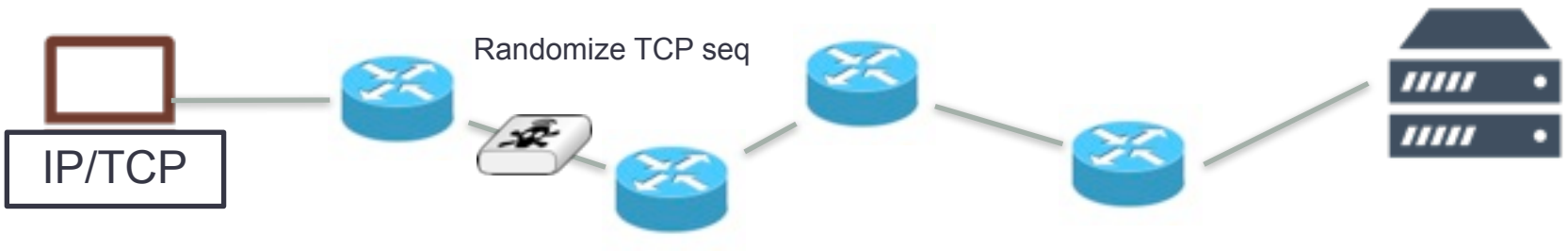
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



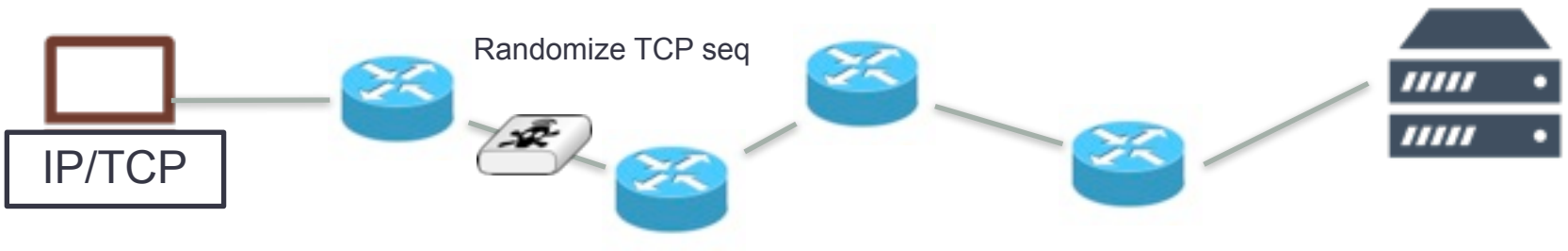
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



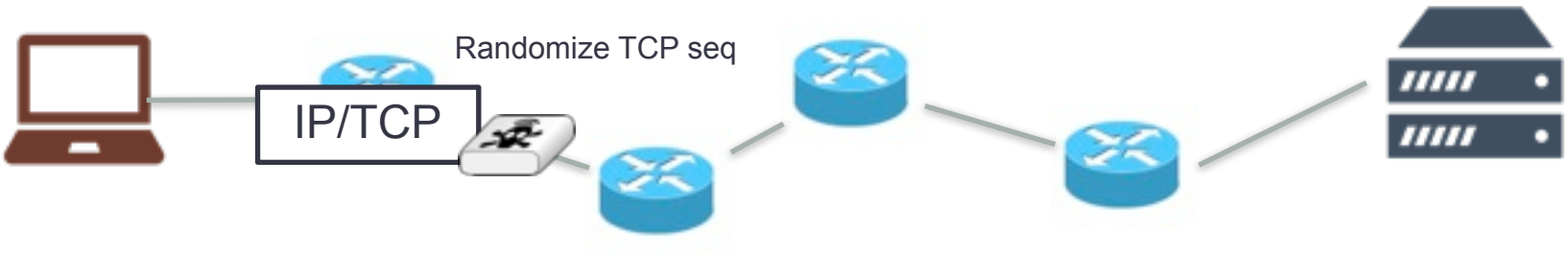
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



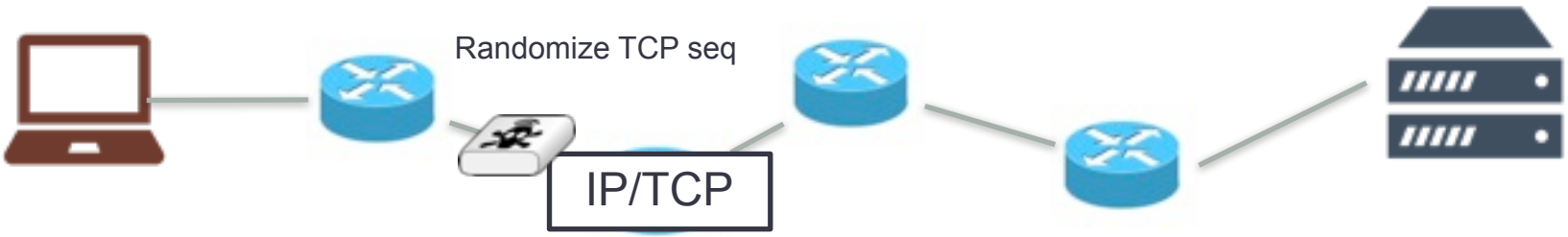
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



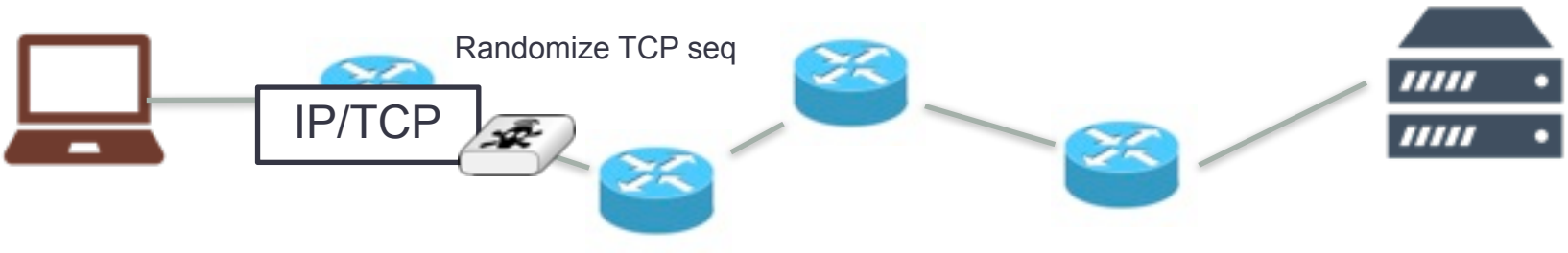
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



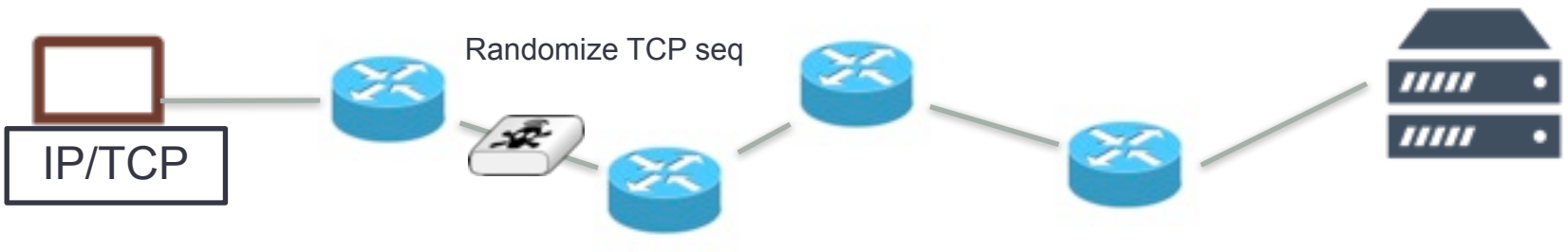
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



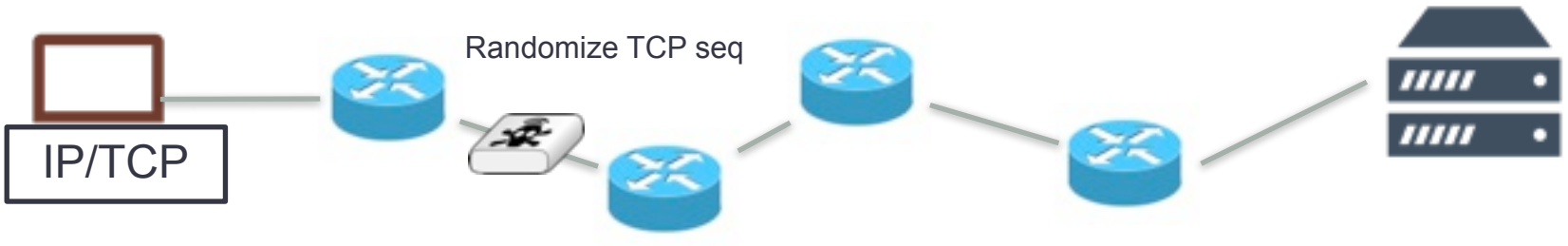
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP

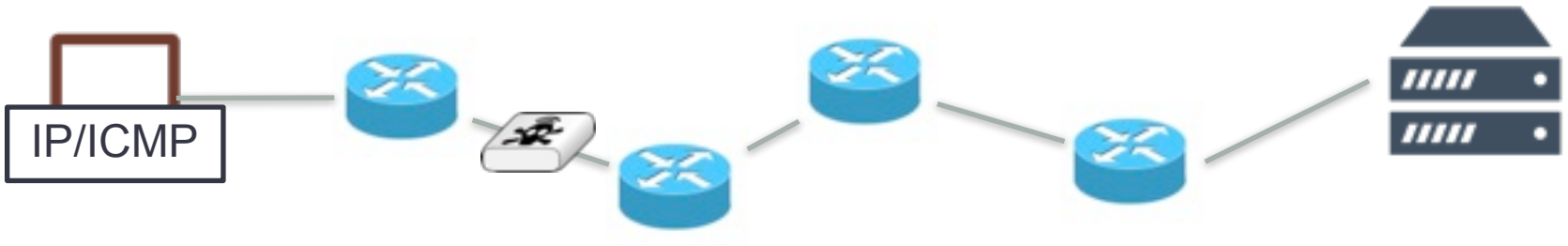


Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
TTL=2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

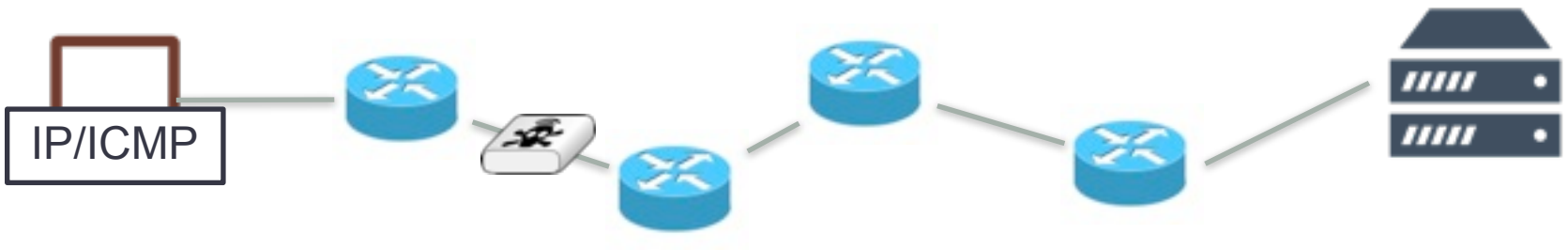
Middlebox detection using ICMP



Middlebox detection using ICMP

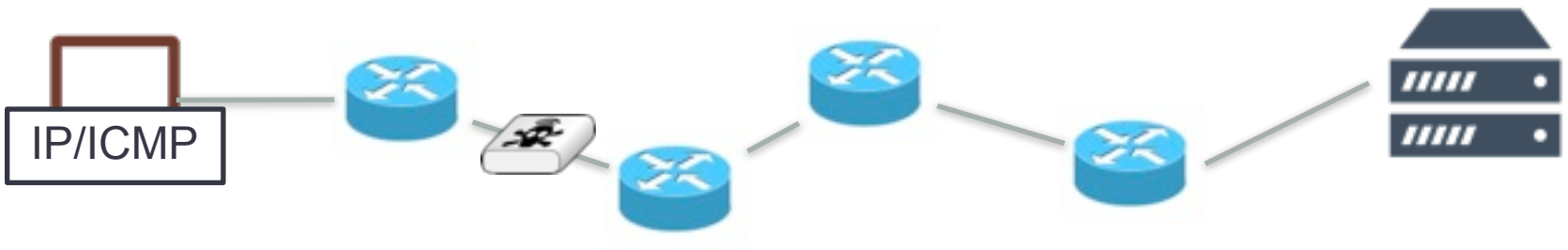


Middlebox detection using ICMP



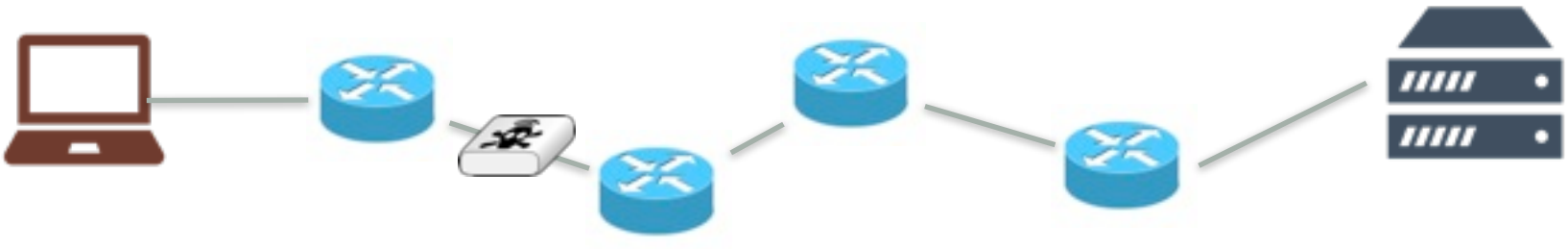
IP		
type = 11	code = 0	*checksum*
0 (unused)		

Middlebox detection using ICMP



IP			
type = 11	code = 0	*checksum*	
0 (unused)			
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

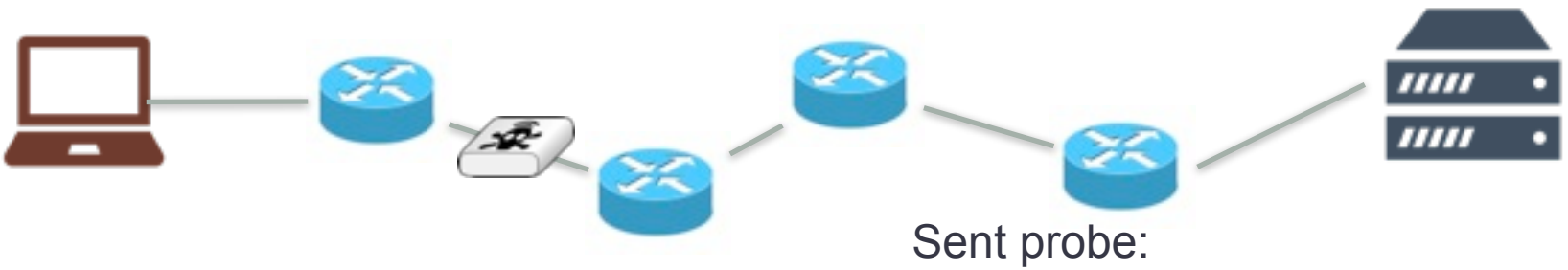
Middlebox detection using ICMP



Snapshot of probe at router:

Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Middlebox detection using ICMP



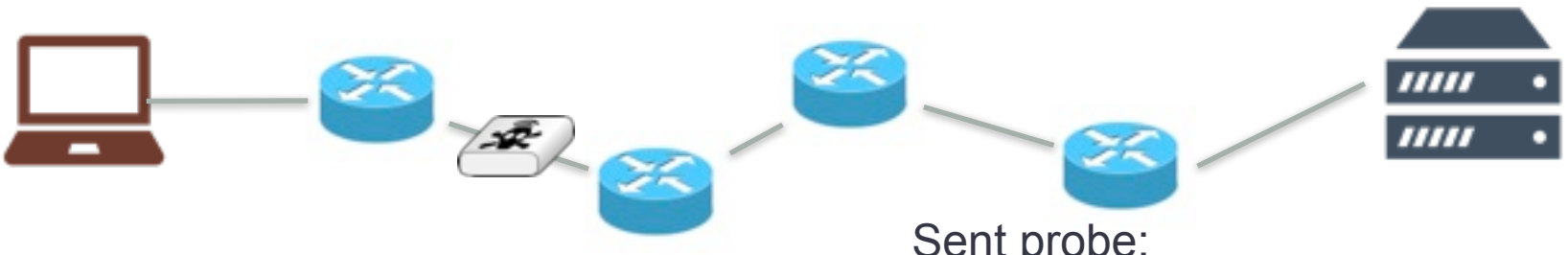
Snapshot of probe at router:

Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Sent probe:

Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum			Urgent pointer

Middlebox detection using ICMP



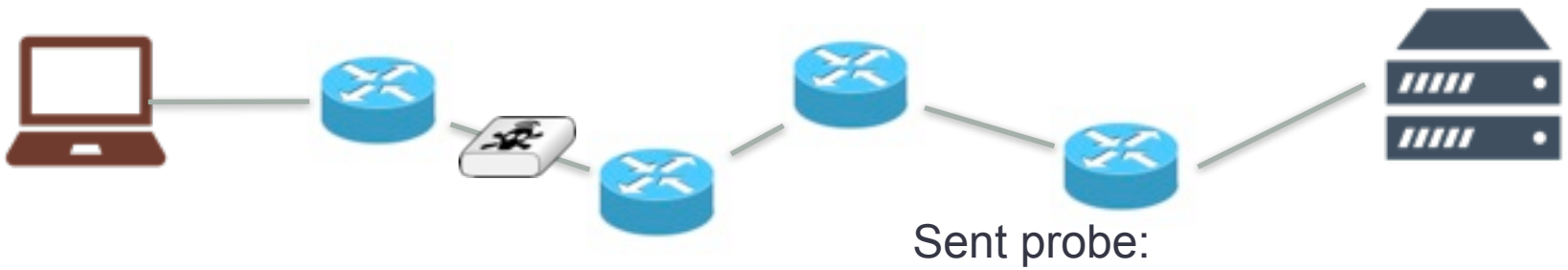
Snapshot of probe at router:

Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Sent probe:

Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Middlebox detection using ICMP



Sent probe:

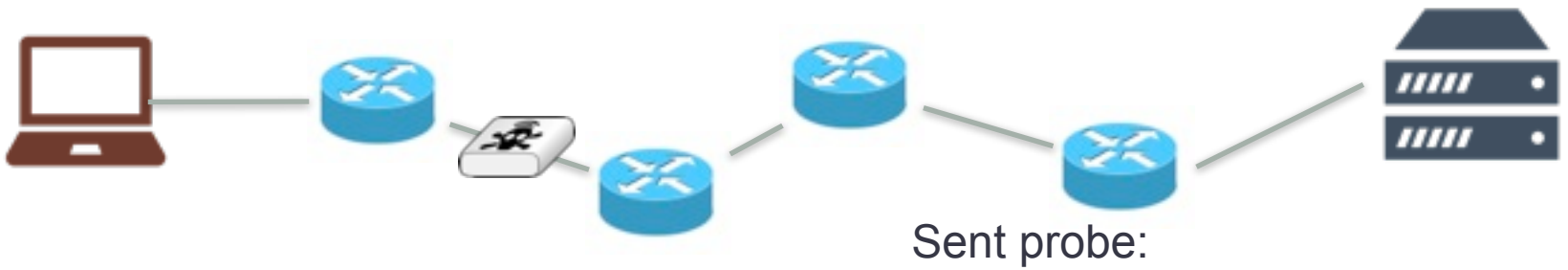
Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
2	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum		Urgent pointer	

Compare

Snapshot of probe at router:

Ver	IHL	ToS	Total length
Identification		Flags	Frag. Offset
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Middlebox detection using ICMP



Sent probe:

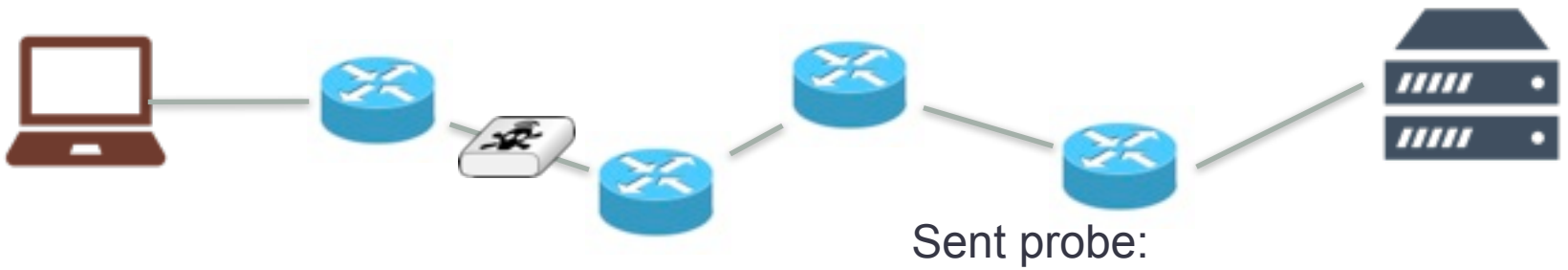
Ver	IHL	ToS	Total length
		Flags	Frag. Offset
		ol	Checksum
P address			
IP address			
Source port			Destination port
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum			Urgent pointer

There is a middlebox

Snapshot of probe at r

Ver	IHL	ToS	Tot
Identification		Flags	F
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Middlebox detection using ICMP



Sent probe:

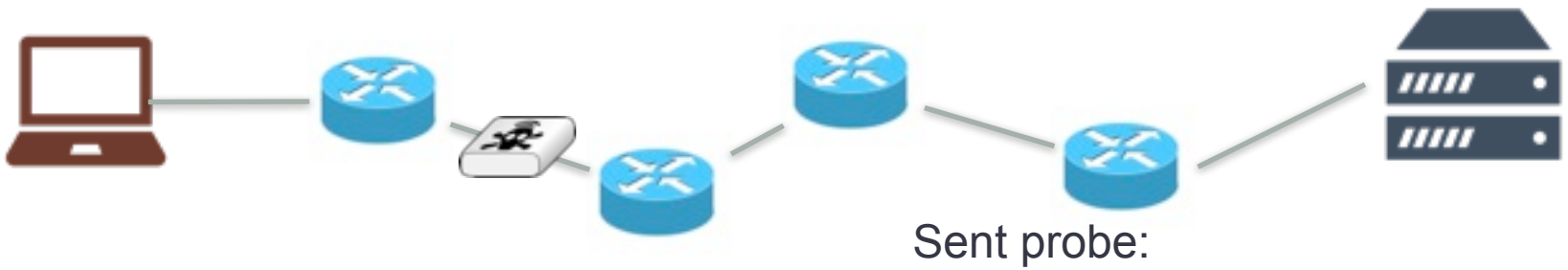
Ver	IHL	ToS	Total length
		Flags	Frag. Offset
			Checksum
P address			
IP address			
Source port			Destination port
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum			Urgent pointer

There is a middlebox that modifies the TCP

Snapshot of probe at r

Ver	IHL	ToS	Tot
Identification		Flags	F
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Middlebox detection using ICMP



Sent probe:

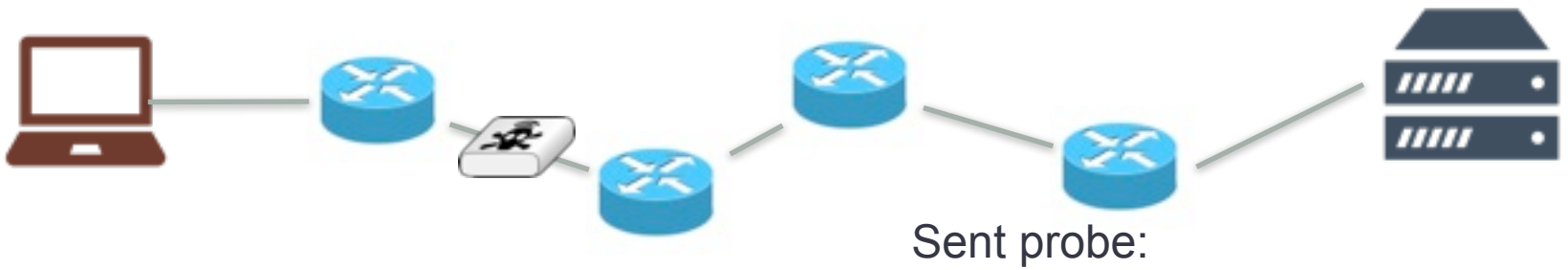
Ver	IHL	ToS	Total length
		Flags	Frag. Offset
			Checksum
			P address
			IP address
Source port			Destination port
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum			Urgent pointer

There is a middlebox that modifies the TCP sequence number before

Snapshot of probe at r

Ver	IHL	ToS	Tot
Identification		Flags	F
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

Middlebox detection using ICMP



Sent probe:

Ver	IHL	ToS	Total length
		Flags	Frag. Offset
			Checksum
P address			
IP address			
Source port			Destination port
Sequence number			
Acknowledgment number			
THL	Reserved	Flags	Window
Checksum			Urgent pointer

There is a middlebox that modifies the TCP sequence number before the second hop

Snapshot of probe at r

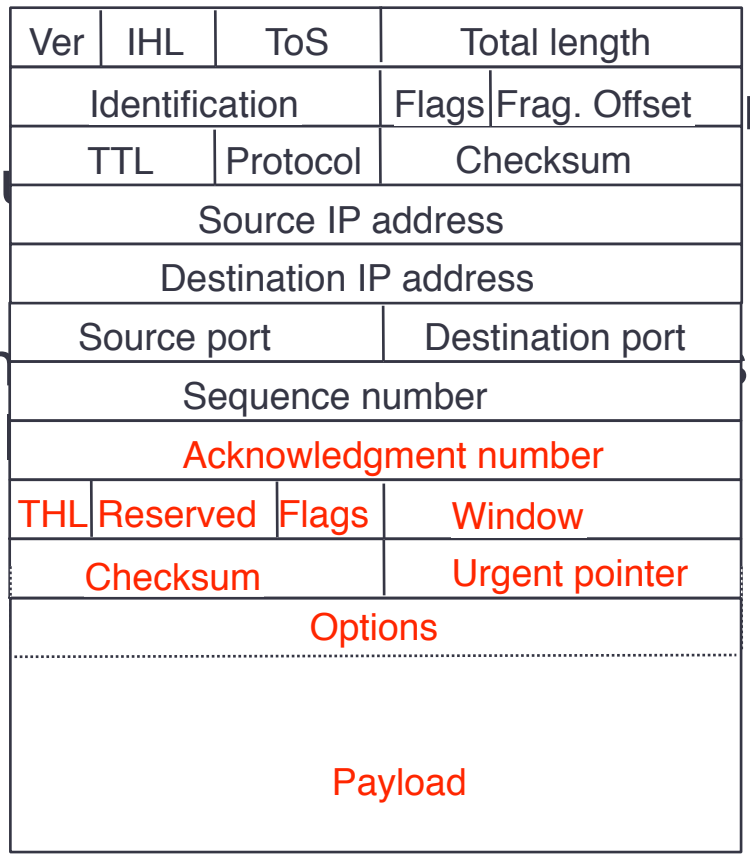
Ver	IHL	ToS	Tot
Identification		Flags	F
1	Protocol	Checksum	
Source IP address			
Destination IP address			
Source port		Destination port	
Sequence number			

ICMP-based modification detection

- RFC792 requires ICMP to include only the first 8 bytes of the transport header.
- In 1995 RFC1812 and in 2007 RFC4884 require that routers should quote the complete original packet.
- By default on Linux, Cisco IOX, HP routers, Alcatel routers, PaloAlto Firewall, etc.

ICMP-based modification detection

- RFC792 requires ICMP to include only the first 8 bytes of the transport header.
- In 1995 RFC181 require that routers should q packet.
- By default on Lin , Alcatel routers, PaloAlto Firewal



ICMP-based modification detection

- RFC792 requires ICMP to include only the first 8 bytes of the transport header.
- In 1995 RFC1812 and in 2007 RFC4884 require that routers should quote the complete original packet.
- By default on Linux, Cisco IOX, HP routers, Alcatel routers, PaloAlto Firewall, etc.

ICMP-based modification detection

- RFC792 requires ICMP to include only the first 8 bytes of the transport header.
- In 1995 RFC1812 and in 2007 RFC4884 require that routers should quote the complete original packet.

ICMP-based modification detection

- RFC792 requires ICMP to include only the first 8 bytes of the transport header.
- In 1995 RFC1812 and in 2007 RFC4884 require that routers should quote the complete original packet.

ICMP-based modification detection

- RFC792 requires ICMP to include only the first 8 bytes of the transport header.
- In 1995 RFC1812 and in 2007 RFC4884 require that routers should quote the complete original packet.
- By default on Linux, Cisco IOX, HP routers, Alcatel routers, PaloAlto Firewall, etc.

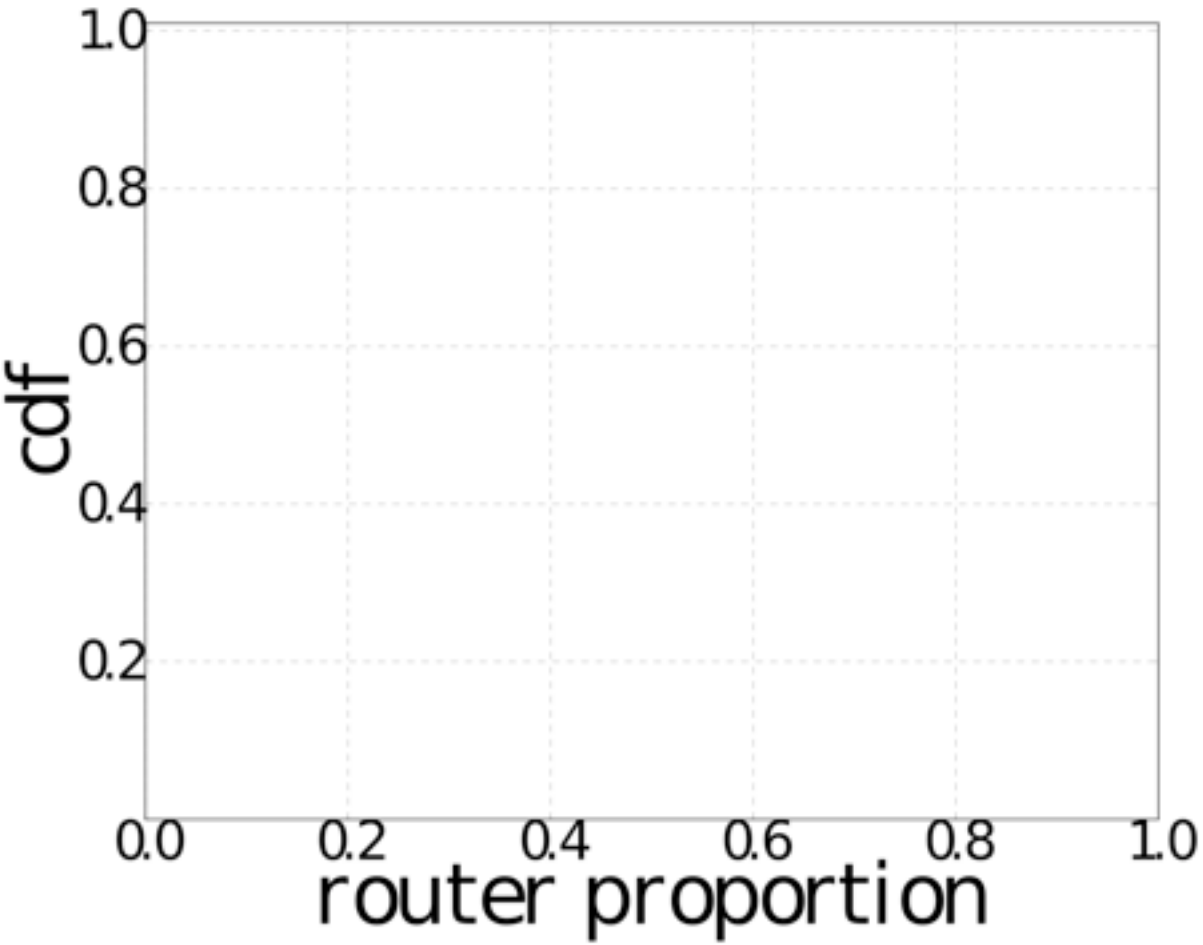
Outline

- Middleboxes interference
- Detect packet modifications with ICMP
- **Measurements results**
- Tracebox

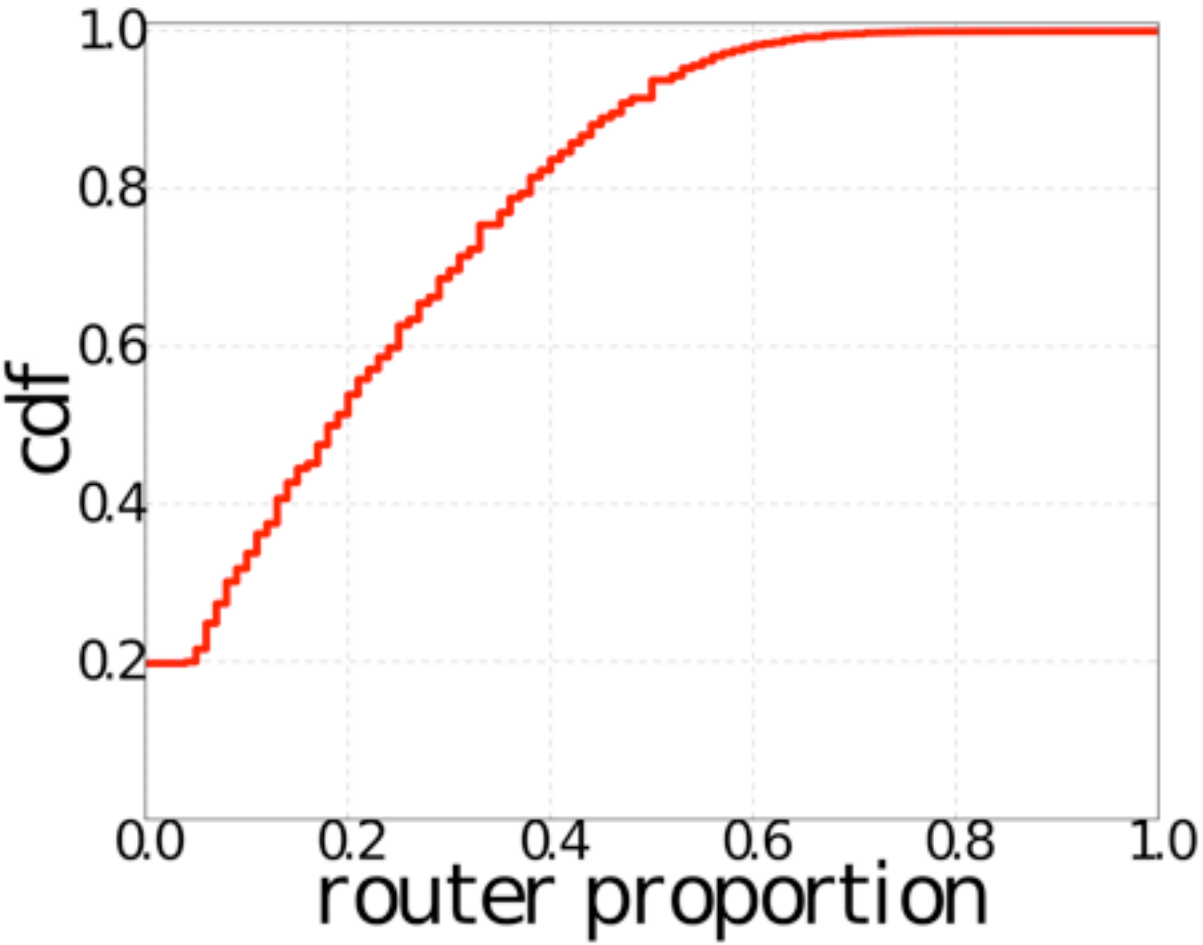
Measurements

- Used PlanetLab to perform experiments
- PlanetLab nodes are supposed to be directly connected to the Internet.
- Sources: 70 vantage points
- Destinations: Top 5000 Alexa

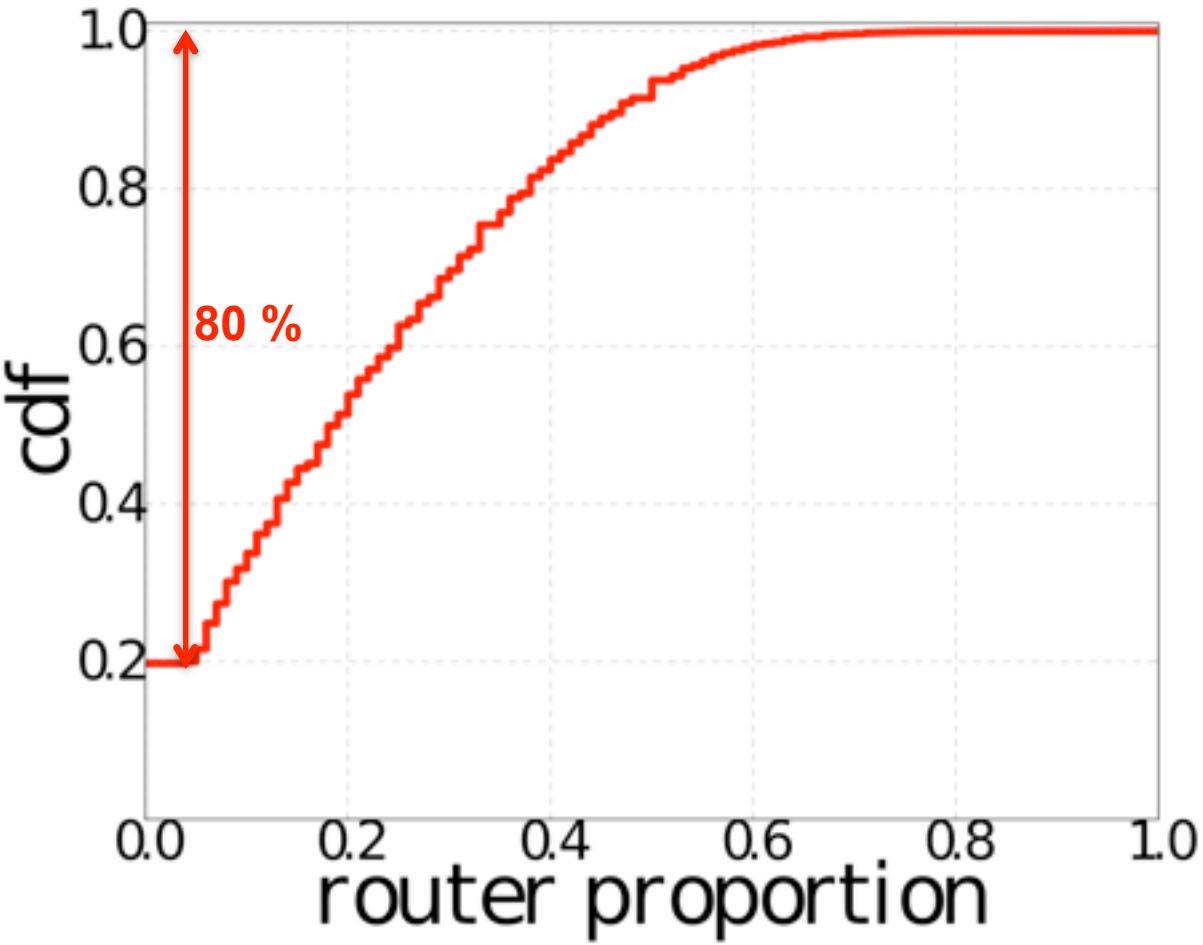
80 % of Internet paths contains at least one RFC1812-capable router



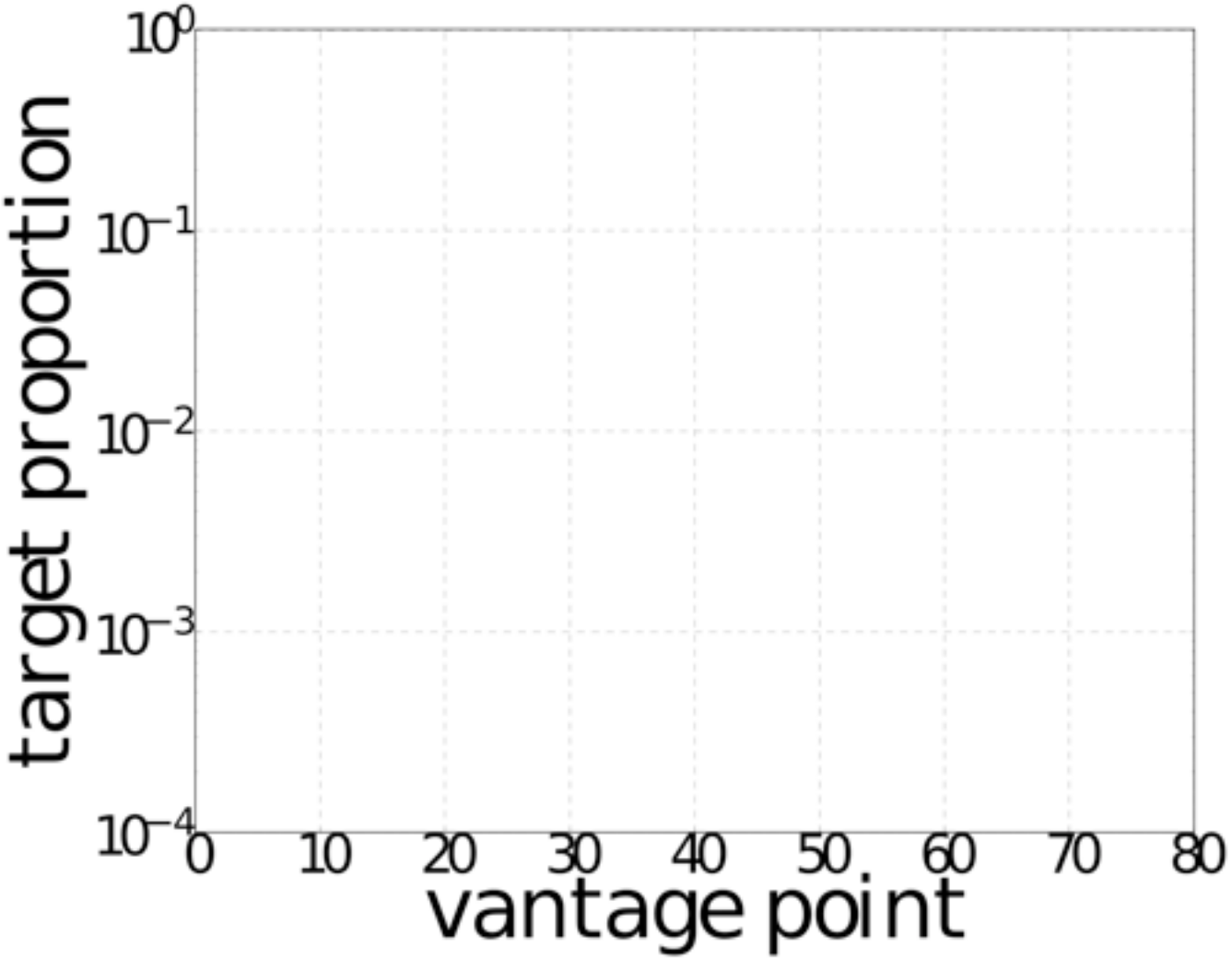
80 % of Internet paths contains at least one RFC1812-capable router



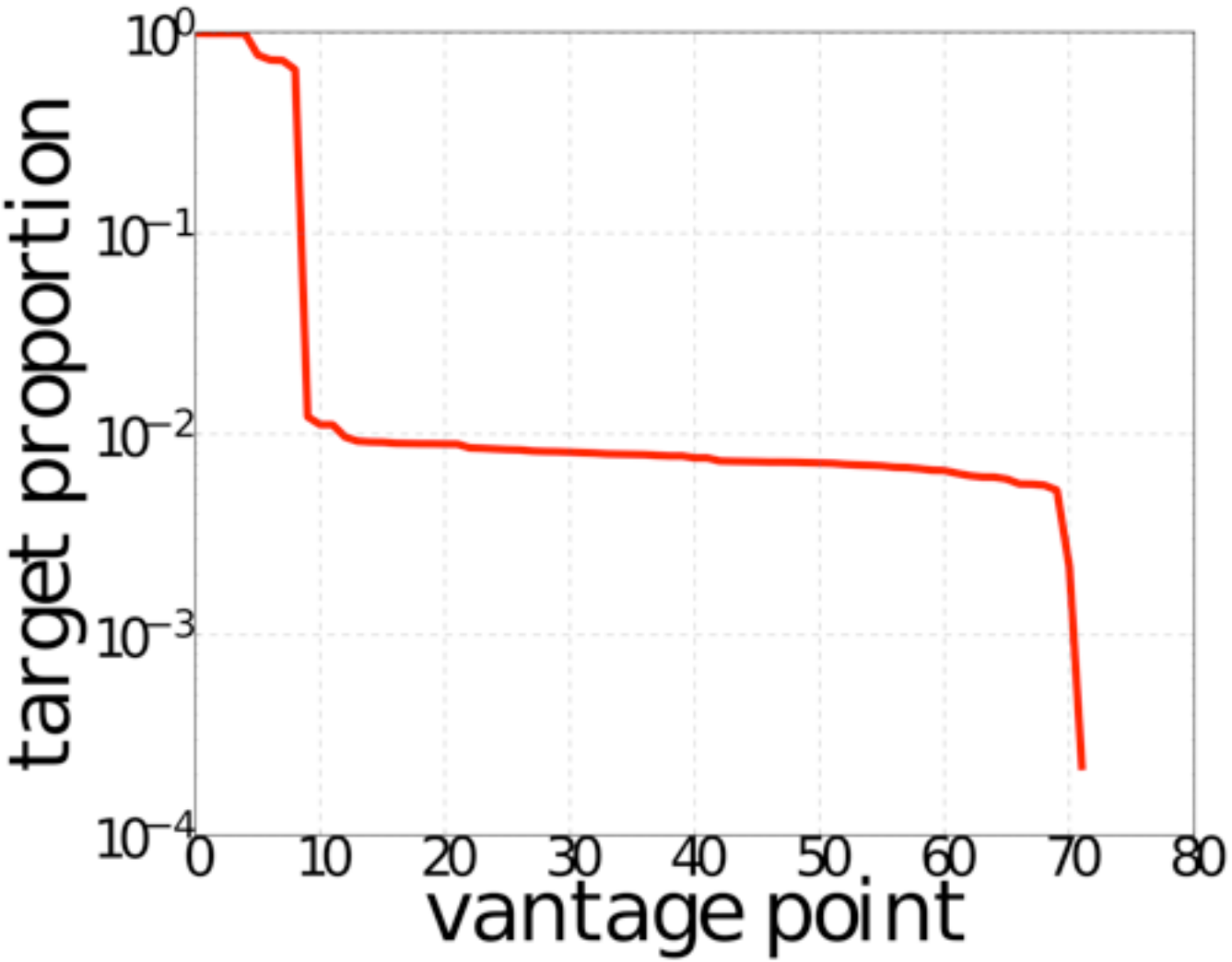
80 % of Internet paths contains at least one RFC1812-capable router



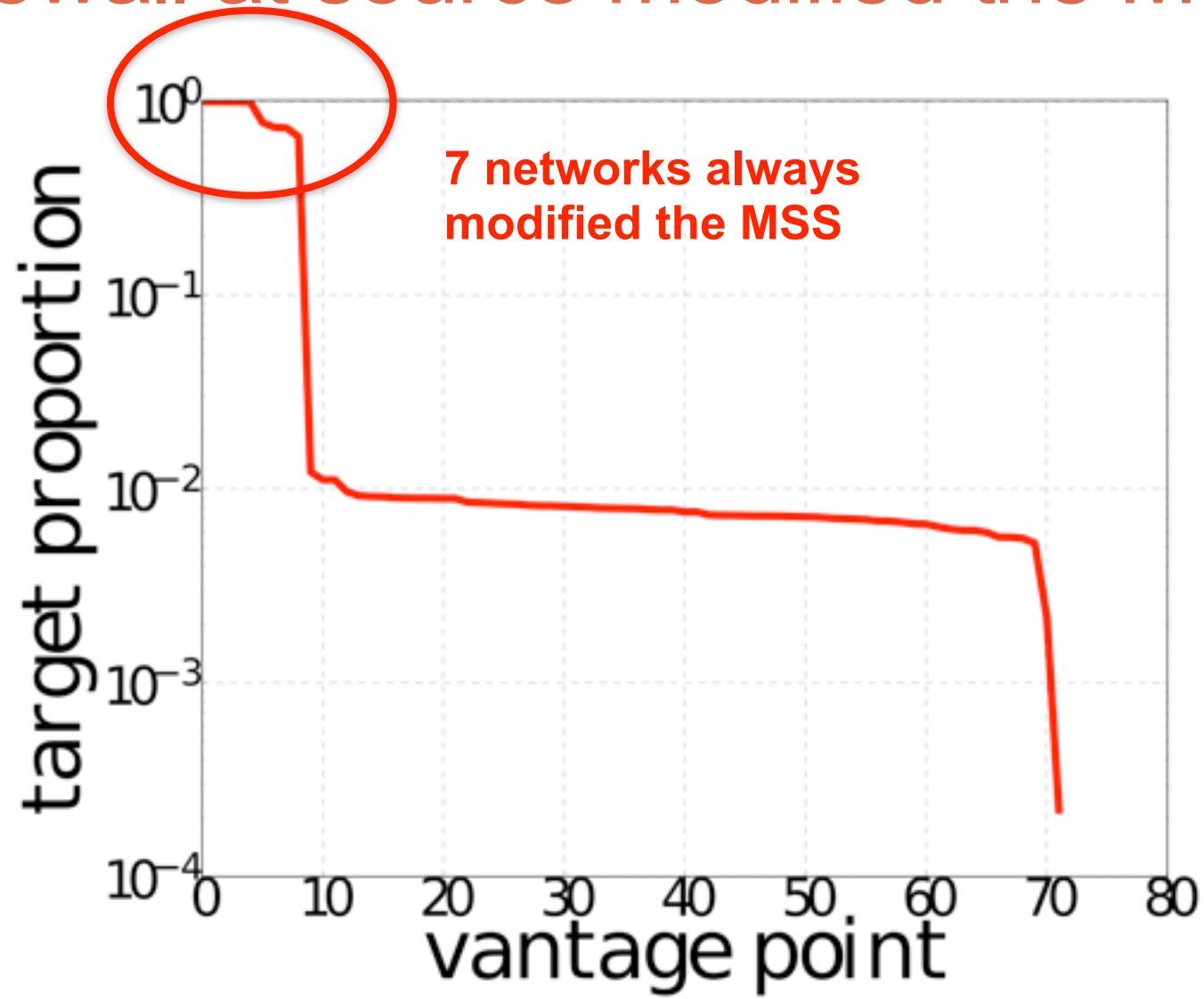
Firewall at source modified the MSS



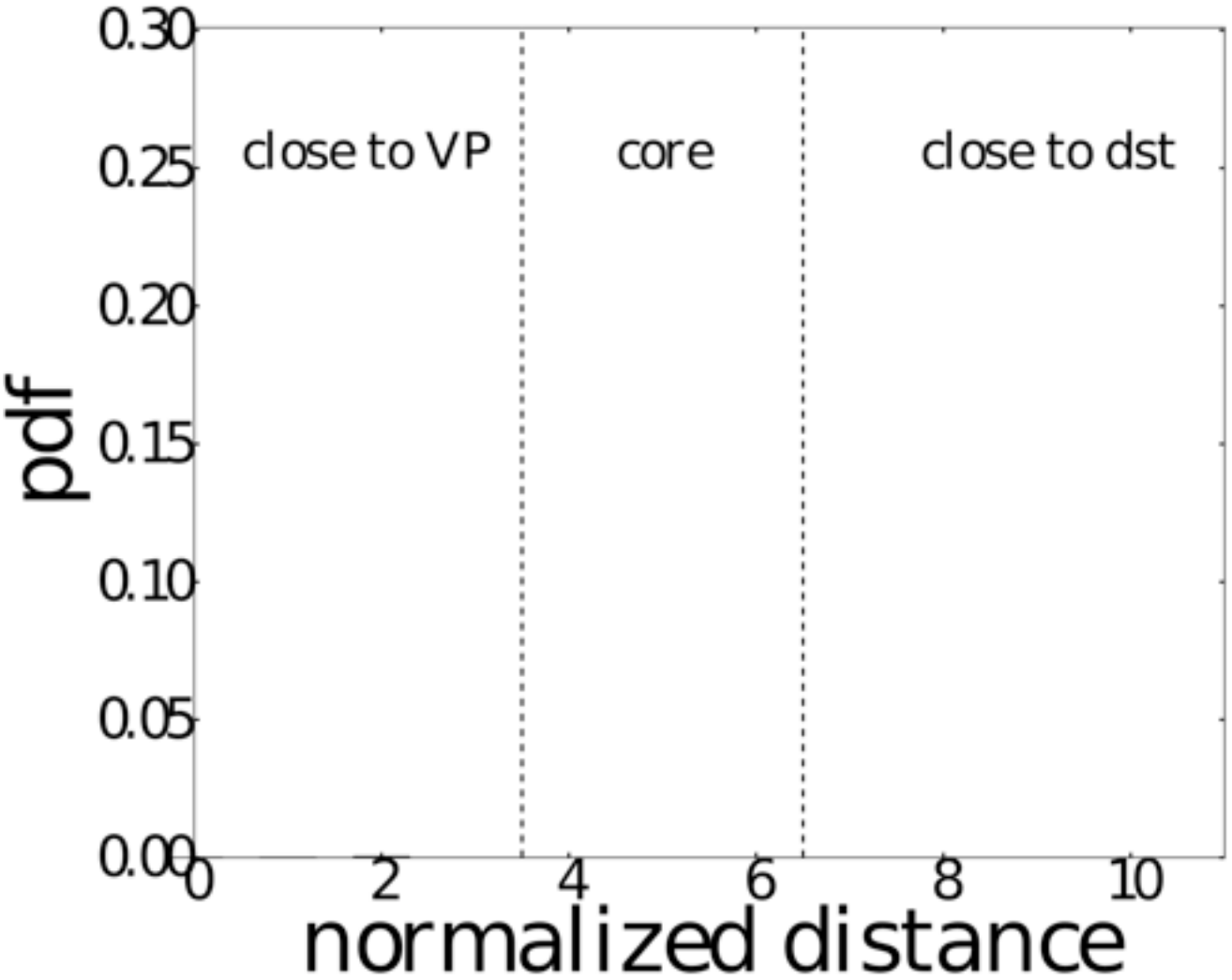
Firewall at source modified the MSS



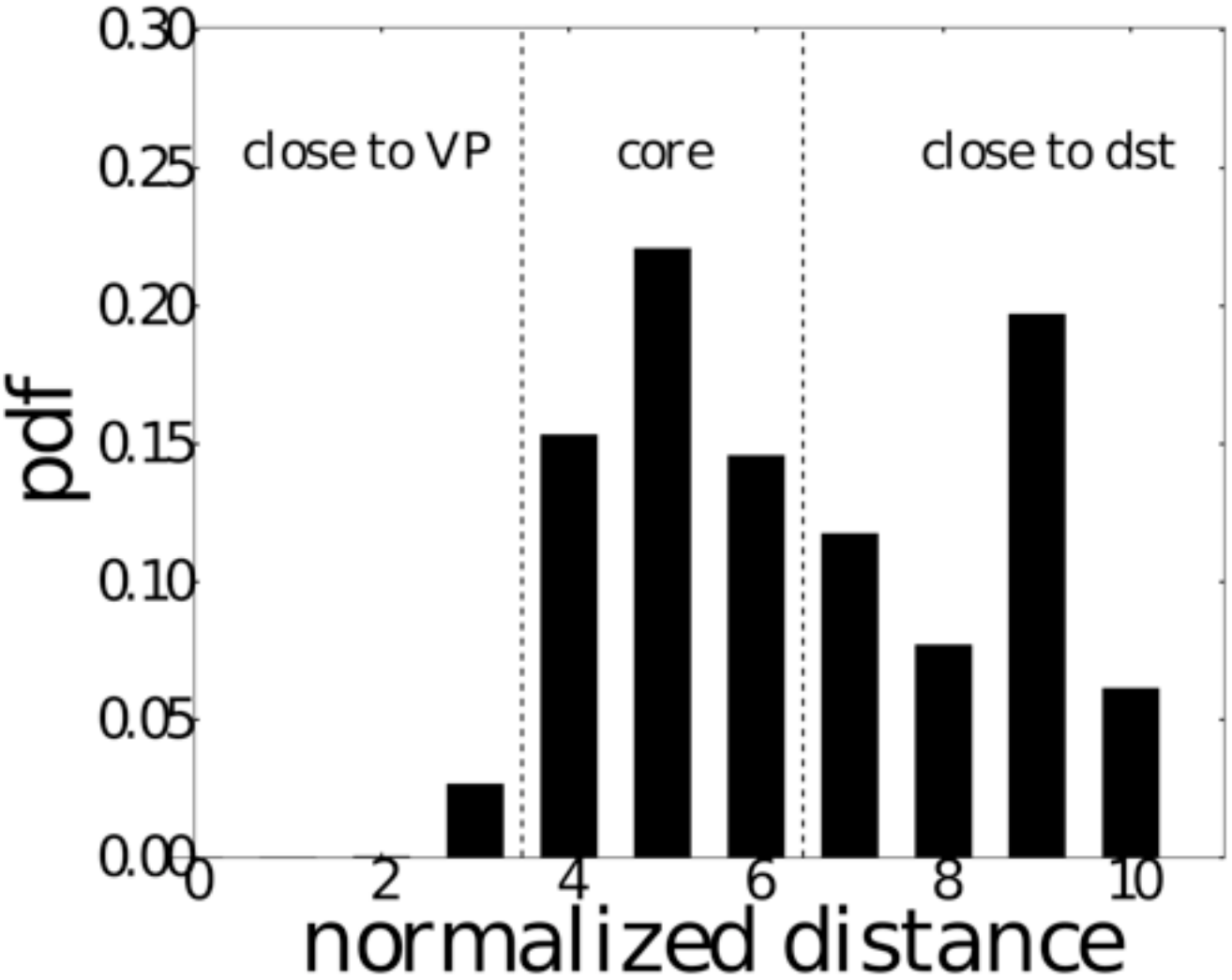
Firewall at source modified the MSS



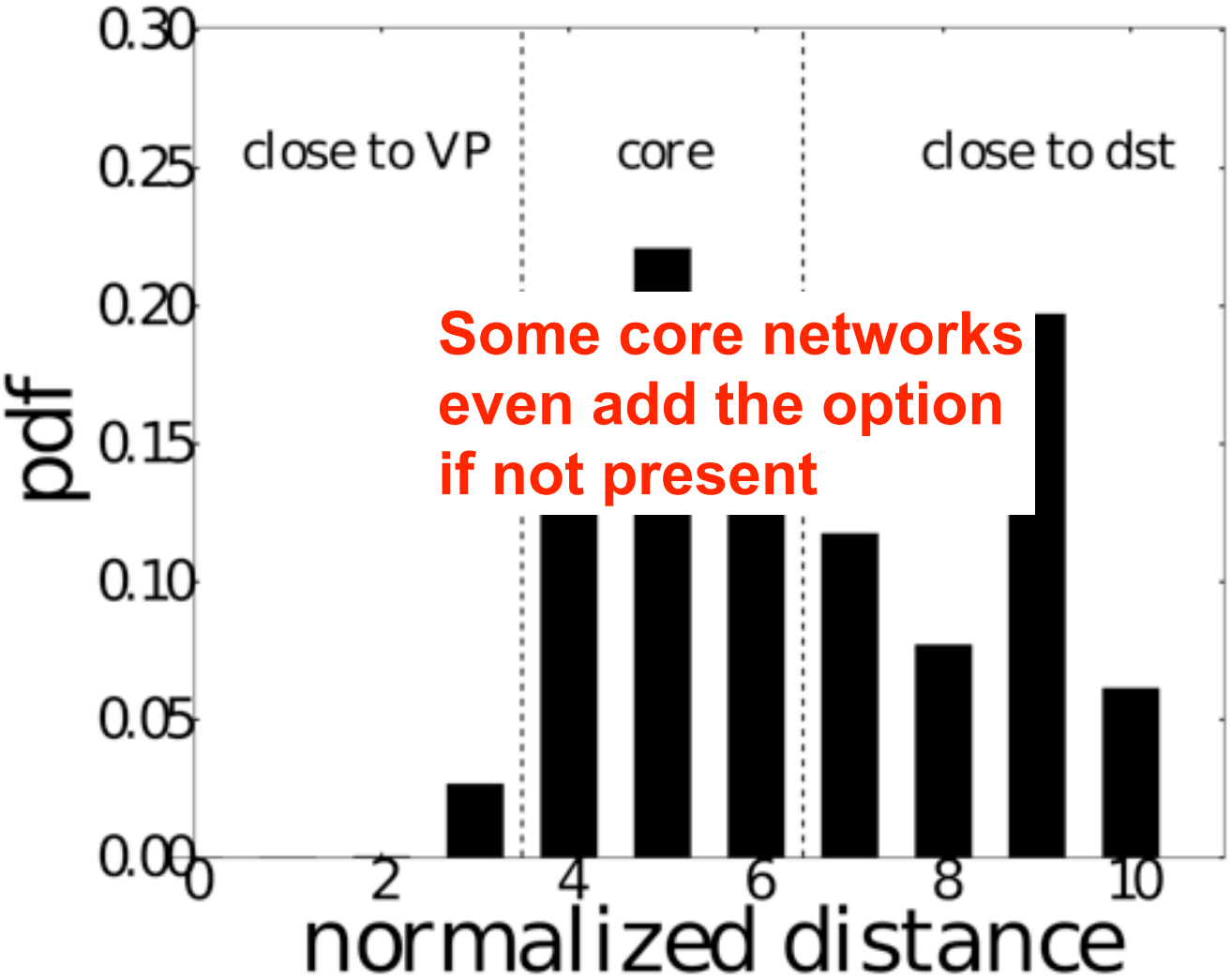
Core network also look at the MSS option and modifies it



Core network also look at the MSS option and modifies it



Core network also look at the MSS option and modifies it



Outline

- Middleboxes interference
- Detect packet modifications with ICMP
- Measurements results
- **Tracebox**

Tracebox

- Uses the previous mechanism to detect middleboxes.
- Implemented in C++ with Lua embedded.
- Libcrafter allows to generate probes as Scapy.
- Open source and available at http://www.tracebox.org
- Supports Linux and Mac OSX

Tracebox

Usage:

tracebox [OPTIONS] host

Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

Tracebox

Usage:

tracebox [OPTIONS] host

Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

Tracebox

Usage:

tracebox [OPTIONS] host

Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

Tracebox

Usage:
tracebox [OPTIONS] host
Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

Tracebox

Usage:

tracebox [OPTIONS] host

Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

Tracebox

Usage:
tracebox [OPTIONS] host
Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

Tracebox

Usage:

tracebox [OPTIONS] host

Options are:

-h	Display this help and exit
-n	Do not resolve IP addresses
-6	Use IPv6 for static probe generated
-u	Use UDP for static probe generated
-d port	Use the specified port for static probe generated. Default is 80.
-i device	Specify a network interface to operate with
-m hops_max	Set the max number of hops (max TTL to be reached). Default is 30
-p probe	Specify the probe to send.
-s script	Run a script.

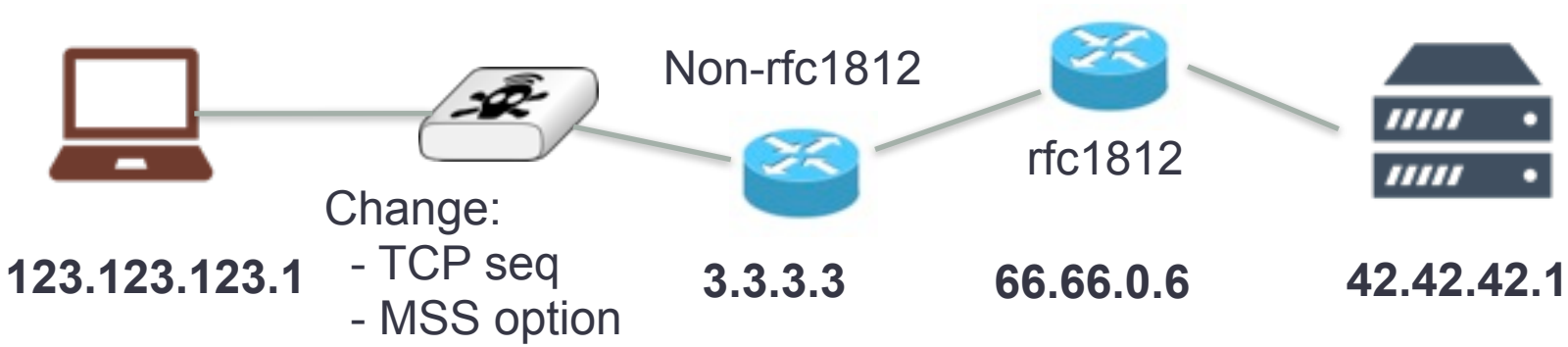
Probe definition

- SYN probe that contains the window scale option
  - ip{} / tcp{flags=0x2,dst=80} / WSCALE
  - IP / TCP / wscale(9) / NOP
- IPv6/UDP probe with payload
  - IPv6 / udp{dst=5678} / raw('this is a payload')
- Multiple options:
  - ip{} / RR(8) / tcp{dst=80} / mss(1400) / WSCALE / TS

Supported layers:

- IP, IPv6
- IP options:
  - Route Record (RR), Strict Source and Record Route(SSRR), Loose Source and Record Route (LSRR), Traceroute
- ICMP
- UDP
- TCP
- TCP options:
  - SACK Permited, SACK blocks, MSS, Timestamp, MPTCP options
- Payload (Raw layer)

Demo in a controlled environment



Lessons learned

- There exists middleboxes that affect performances and network operators are not always aware of them.

- Tracebox can detect some middleboxes.

What's next ?

- Improve scripts to detect ALG
- Deploy Tracebox

Thank you. Questions ?

fabien.duchene@uclouvain.be
http://www.tracebox.org

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
  8: 81.200.194.234 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
  8: 81.200.194.234 IP::TTL IP::CheckSum
  9: 81.200.197.9 IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
  8: 81.200.194.234 IP::TTL IP::CheckSum
  9: 81.200.197.9 IP::TTL IP::CheckSum
  10: 81.200.198.6 TCP::CheckSum IP::TTL IP::CheckSum
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
  8: 81.200.194.234 IP::TTL IP::CheckSum
  9: 81.200.197.9 IP::TTL IP::CheckSum
  10: 81.200.198.6 TCP::CheckSum IP::TTL IP::CheckSum
                   TCPOptionMaxSegSize::MaxSegSize
```

Output example

```
# tracebox -n -p “IP/TCP/MSS/MPCAPABLE/WSCALE” bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
  8: 81.200.194.234 IP::TTL IP::CheckSum
  9: 81.200.197.9 IP::TTL IP::CheckSum
  10: 81.200.198.6 TCP::CheckSum IP::TTL IP::CheckSum
                TCPOptionMaxSegSize::MaxSegSize
                –TCPOptionMPTCPCapable -TCPOptionWindowScale
```

Output example

```
# tracebox -n -p IP/TCP/MSS/MPCAPABLE/WSCALE bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
  1: 130.104.228.126 IP::CheckSum
  2: 130.104.254.229 IP::TTL IP::CheckSum
  3: 193.191.3.85 IP::TTL IP::CheckSum
  4: 193.191.16.21 IP::TTL IP::CheckSum
  5: 195.69.144.123 IP::TTL IP::CheckSum
  6: 145.254.5.158 IP::TTL IP::CheckSum
  7: 88.79.13.62 IP::TTL IP::CheckSum
  8: 81.200.194.234 IP::TTL IP::CheckSum
  9: 81.200.197.9 IP::TTL IP::CheckSum
  10: 81.200.198.6 TCP::CheckSum IP::TTL IP::CheckSum
                TCPOptionMaxSegSize::MaxSegSize
                –TCPOptionMPTCPCapable -TCPOptionWindowScale
```

Output example

```
# tracebox -n -p IP/TCP/MSS/MPCAPABLE/WSCALE bahn.de
  tracebox to 81.200.198.6 (bahn.de): 64 hops max
   1: 130.104.228.126 IP::CheckSum
   2: 130.104.254.229 IP::TTL IP::CheckSum
   3: 193.191.3.85 IP::TTL IP::CheckSum
   4: 193.191.16.21 IP::TTL IP::CheckSum
   5: 195.69.144.123 IP::TTL IP::CheckSum
   6: 145.254.5.158 IP::TTL IP::CheckSum
   7: 88.79.13.62 IP::TTL IP::CheckSum
   8: 81.200.194.234 IP::TTL IP::CheckSum
   9: 81.200.197.9 IP::TTL IP::CheckSum
   10: 81.200.198.6 TCP::CheckSum IP::TTL IP::CheckSum
            TCPOptionMaxSegSize::MaxSegSize
            –TCPOptionMPTCPCapable -TCPOptionWindowScale
```