
black hat®
USA 2019
AUGUST 3-8, 2019
MANDALAY BAY / LAS VEGAS
#BHUSA @BLACK HAT EVENTS

Critical Zero Days Remotely Compromise VxWorks The Most Popular RTOS

Ben Seri, VP Research
Dor Zusman, Researcher

ARMIS

Agenda

- What is VxWorks?
- TCP/IP Stack vulnerabilities
- What is URGENT/11?
- Technical deep dive
- Case study & Live demo – Patient Monitor

What is it and why should I care?

- Real-time Operating System
- Only 13 CVE's listed on MITRE
- Been around for 32 years, runs on over 2 Billion devices

VxWorks is **everywhere**

Healthcare

Manufacturing

Infrastructure/Network

Security

Auto

Aerospace

Defense

High Tech

VxWorks is used by **everybody**

Alcatel·Lucent

Rockwell

SIEMENS

SAMSUNG

RICOH

XEROX®

AIRBUS

Raytheon

AVAYA

SONICWALL

BOEING

ERICSSON

ARRIS

CISCO

NASA

Schneider Electric

PHILIPS

SPACEX

BELDEN


Why research TCP/IP stacks?

Remember WinNuke?

Windows

An exception 0E has occurred at 0028:C1064332 in VxD MSTCP(01) +
000041AE. This was called from 0028:C0050F8B in VxD NDIS(01) +
00006733. It may be possible to continue normally.

* Press any key to attempt to continue.
* Press CTRL+ALT+DEL to restart your computer. You will
   lose any unsaved information in all applications.

Press any key to continue

Kevin Backhouse
@kevin_backhouse

Follow

Video of my PoC for CVE-2018-4407. It crashes any macOS High Sierra or iOS 11 device that is on the same WiFi network. No user interaction required.

12:24 PM - 30 Oct 2018

2,953 Retweets **5,510** Likes

103 3.0K 5.5K

URGENT/11

11 Critical Zero Day Vulnerabilities in VxWork's TCP/IP Stack – IPnet

6 Remote Code Execution (RCEs)

5 Information Leaks, Denial of Service, Logical Flaws

Affects VxWorks versions for the last 13 years (v6.5 and up)

Affects hundreds of millions of devices

IPnet (VxWorks' TCP/IP Stack) is owned and maintained by Wind River

IPnet also used by other RTOSs in the past

INTEGRITY (GreenHills)
ThreadX (Today owned by Microsoft)
OSE (ENEA)

5 months discloser process

Security Advisories

Schneider Electric

Security Bulletin – Wind River VxWorks Vulnerabilities (URGENT/11)

2 August 2019

Overview

Schneider Electric is aware of recently disclosed vulnerabilities in Wind River's VxWorks TCP/IP Stack. These vulnerabilities have wide-ranging impact across multiple IT and industrial applications. We are working closely with Wind River to understand and assess how these vulnerabilities impact Schneider Electric offers and our customers' operations. We downloaded Wind River's patches as soon as they were made available to us, and we have quickly instituted a remediation plan to evolve all current and future products that rely on the Wind River platform to embed these fixes.

We will continue to monitor and will respond further if new information becomes available. In the meantime, customers should immediately make sure they have implemented cybersecurity best practices across their operations to protect themselves from these vulnerabilities. Where appropriate this includes locating your indu behind firewalls; installing physical controls mission-critical systems and devices from b

Please subscribe to the Schneider Electric updates to this disclosure, including details as other important security notifications:

https://www.schneider-electric.com/en/work

For additional information and support, plea representative or Schneider Electric's Custo

Details

Additional details on these specific vulnerab notification webpage:

https://www.windriver.com/security/announc

Of the 11 identified Wind River vulnerabilitie execution. Four of those six can be mitigate to the device is required. The other vulnerat information disclosure.

2-Aug-19 Document Reference

Rockwell

Industries Capabilities Products News & Events Sales & Partners Support

VxWorks Vulnerabilities affect
Programmable Automation Controllers, EtherNet/IP Communication Modules, I/O Modules, Kinetix 6500 Servo Drive, High-Frequency RFID Interface Block

1088561 | Date Created: 07/16/2019 | Last Updated: 08/01/2019
Access Level: Everyone

Email this page Print Subscribe to Updates

Version 1.0 – July 30, 2019

Armis, an IoT security firm, reported a total of eleven vulnerabilities to WindRiver that affect VxWorks, a real-time operating system (RTOS) utilized by many different technology vendors, including Rockwell Automation™. These vulnerabilities, if successfully exploited, may result in several impacts ranging from packet information disclosure to allowing a threat actor to execute arbitrary code on the targeted device.

Not every VxWorks vulnerability applies to every impacted product family. Please see the table under Affected Products for a full list of the potentially affected Rockwell Automation products and the corresponding VxWorks vulnerabilities, which are identified by their Common Vulnerabilities and Exposures (CVE) ID.

Currently, Rockwell Automation is working to address these available. Please subscribe to updates to this advisory and notified.

Customers using potentially affected products are encourag those listed below. Additional details relating to the discov

AFFECTED PRODUCTS

Product Family	Catalog	CVE-2019-12253	CVE-2019-12256	CVE-2019-12257
CompactLogix™ 5480 (EPIC controller)	5069-L4		x	
Compact 5000™ I/O EtherNet/IP Adapter	5069-AEN2TR		x	
ControlLogix® 5580 (+ GuardLogix®)	1756-L8		x	
CompactLogix Compact GuardLogix 5380	5069-L3		x	
CompactLogix 5370	1769-L3	x		x
CompactLogix GuardLogix 5370	1769-L35	x		x

XEROX®

xerox Search

Printers & Supplies | Solutions & Services | Customer Support | Partners

Security at Xerox

Wind River VXWorks IPnet TCP/IP STACK Vulnerabilities

Name	Wind River VXWorks IPnet TCP/IP STACK Vulnerabilities
Tracking Number	2019-001
First Publish Date	22 Jul 2019
Date of Current Status	22 Jul 2019
Next Planned Update	22 Aug 2019
Description	A number of vulnerabilities in Wind River's VXWorks IPnet TCP/IP Stack implementation have been reported. These vulnerabilities could allow attackers to hijack existing TCP sess Service attacks.
What You Need To Know?	Security researchers rep Stack implementation th existing TCP/IP connect or force transmittal of im middle, replay, and othe

Currently available infor conditions, and NULL-p or network connectivity information also sugges exploitation.

The 11 CVEs that were 2019-12265. Exploitabil

One of more of these 11

- All versions of VxWo SR610)
- Older, End-of-Life ve
- All versions of the dis
- IPnet when sold as a
- The VxWorks bootro

VXWorks 5.3 through 6.

PHILIPS

Security Advisory & Archive

VxWorks Urgent/11 Advisory (1 August 2019)

Publication Date: August 1, 2019
Update Date: August 2, 2019

Security researchers at Armis have disclosed 11 different zero-day vulnerabilities within Wind River used in over 2 billion embedded systems that include medical devices, routers, VOIP phones and The collection of vulnerabilities, which Armis refers to as "Urgent/11," could lead to remote code exe whole system without interacting with the user. Of the 11 flaws, six are deemed critical. Successful an unauthorized user to execute arbitrary code on the target system. An unauthorized user could data; or create new accounts with full user rights.

Philips is currently monitoring developments and updates related to the recent published advisor CVEs as referred to as Urgent/11. In the ac consideration for product evaluation and

As part of the company's product securit for potential impacts from these reported vulnerabilities and evaluating further acti remediate these vulnerabilities. Philips is

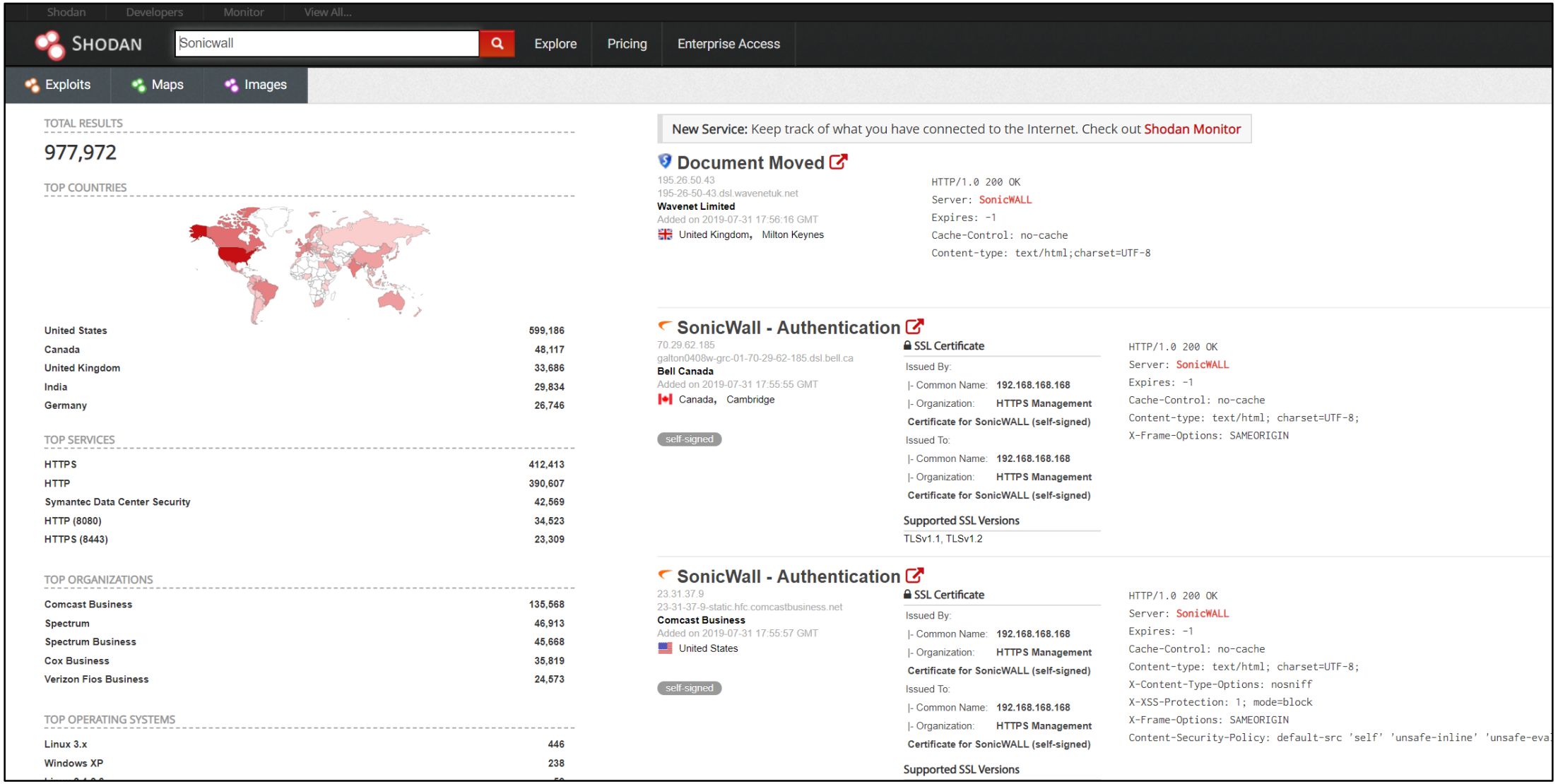
Philips is committed to ensuring the safe Philips approved product specifications. or software to Philips' products (including Philips product-specific, verified & validat

If a product does require operating syste Customer Services, product-specific servi delivery platforms such as the Philips InC contract-entitled customers, licensed rep

Contract-entitled customers may use Phi information posted. If customers still hav service support team or regional product

Begin Update A: August 2, 2019

SonicWall Firewalls exposed to the internet



IP Options parsing stack overflow
(CVE-2019-12256)

TCP Urgent Pointer TCP AO state confusion
(CVE-2019-12255)

TCP Urgent Pointer 0 integer underflow
(CVE-2019-12255)

TCP Urgent Pointer race condition
(CVE-2019-12263)

Reverse ARP logical flaw
(CVE-2019-12262)

TCP connection DoS via malformed options
(CVE-2019-12258)

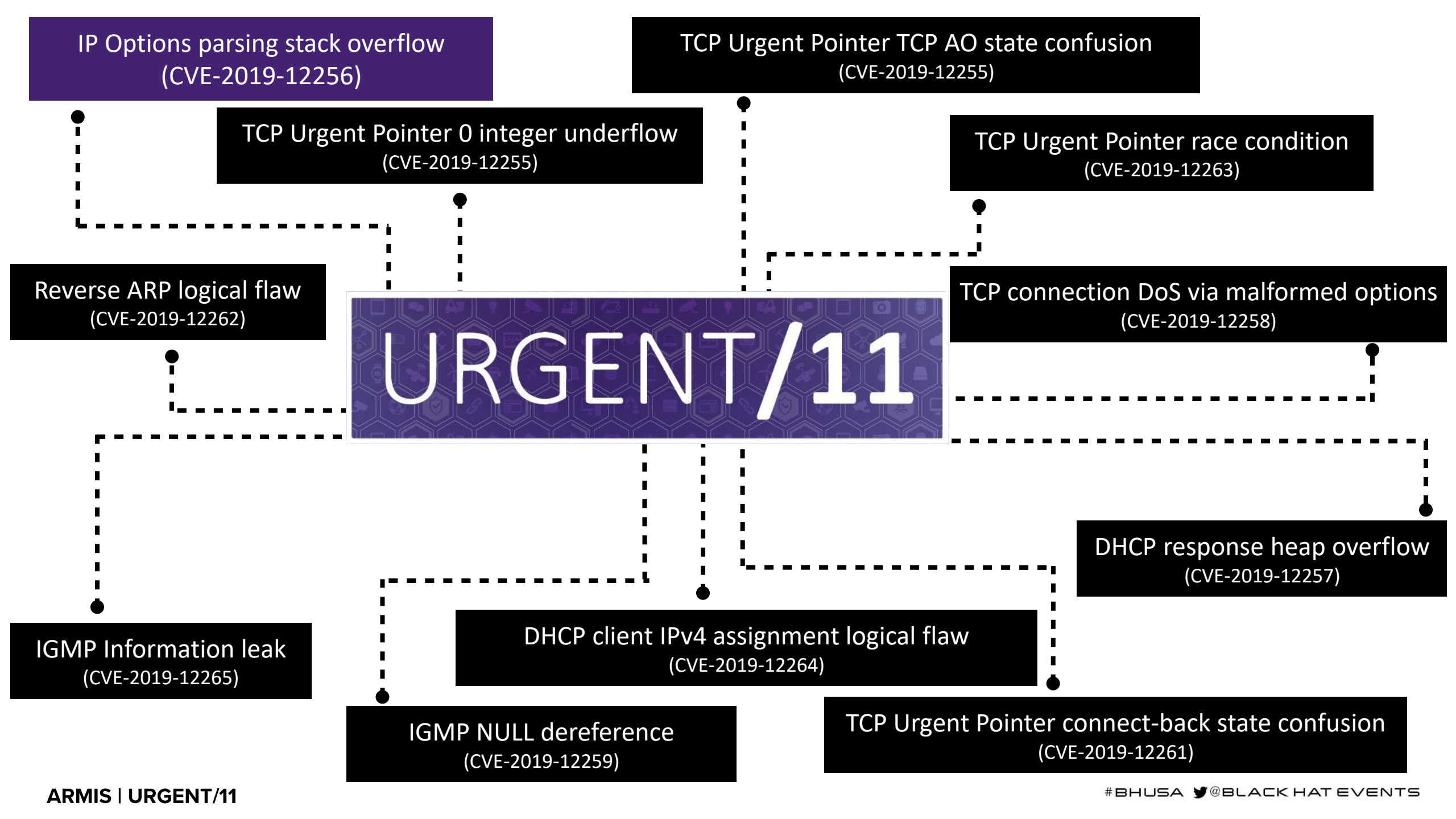
DHCP response heap overflow
(CVE-2019-12257)

IGMP Information leak
(CVE-2019-12265)

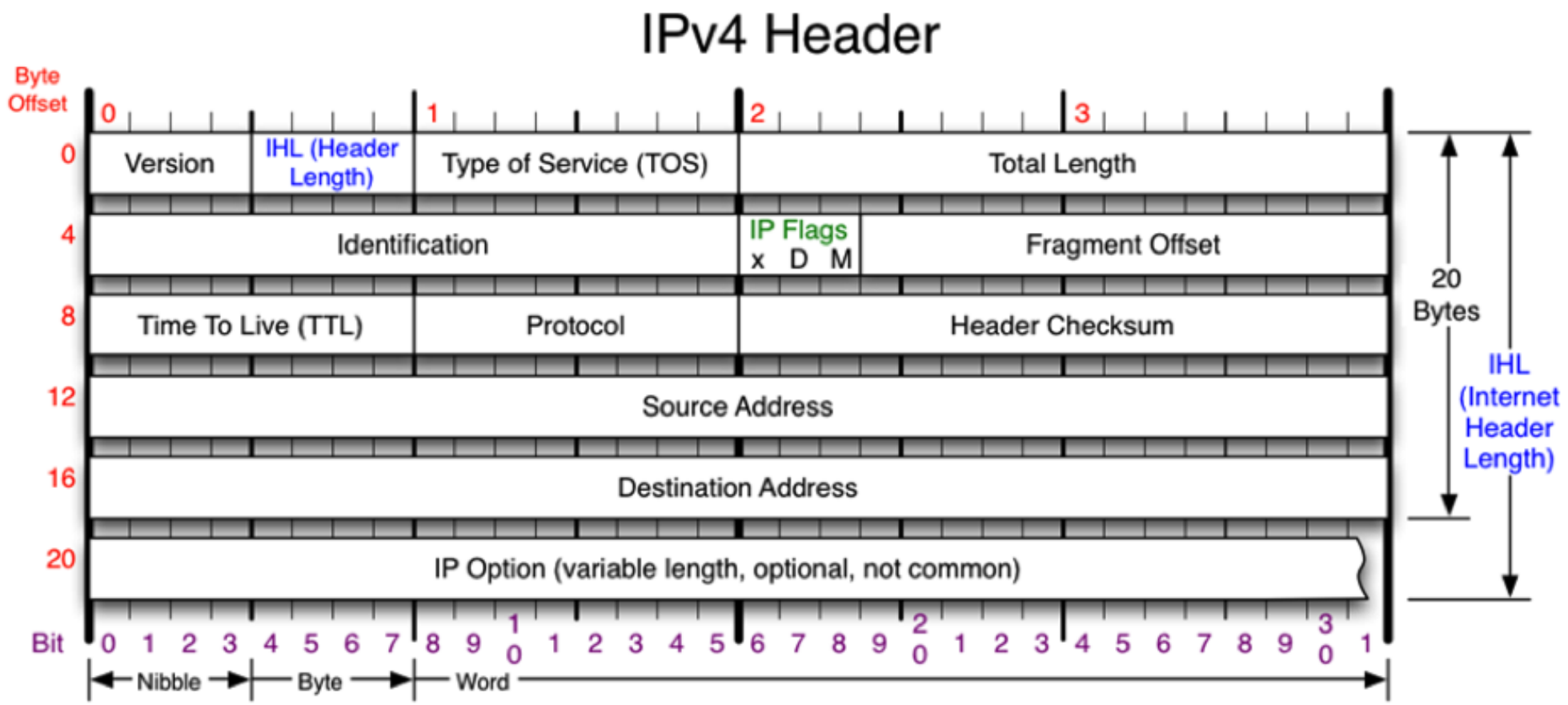
DHCP client IPv4 assignment logical flaw
(CVE-2019-12264)

IGMP NULL dereference
(CVE-2019-12259)

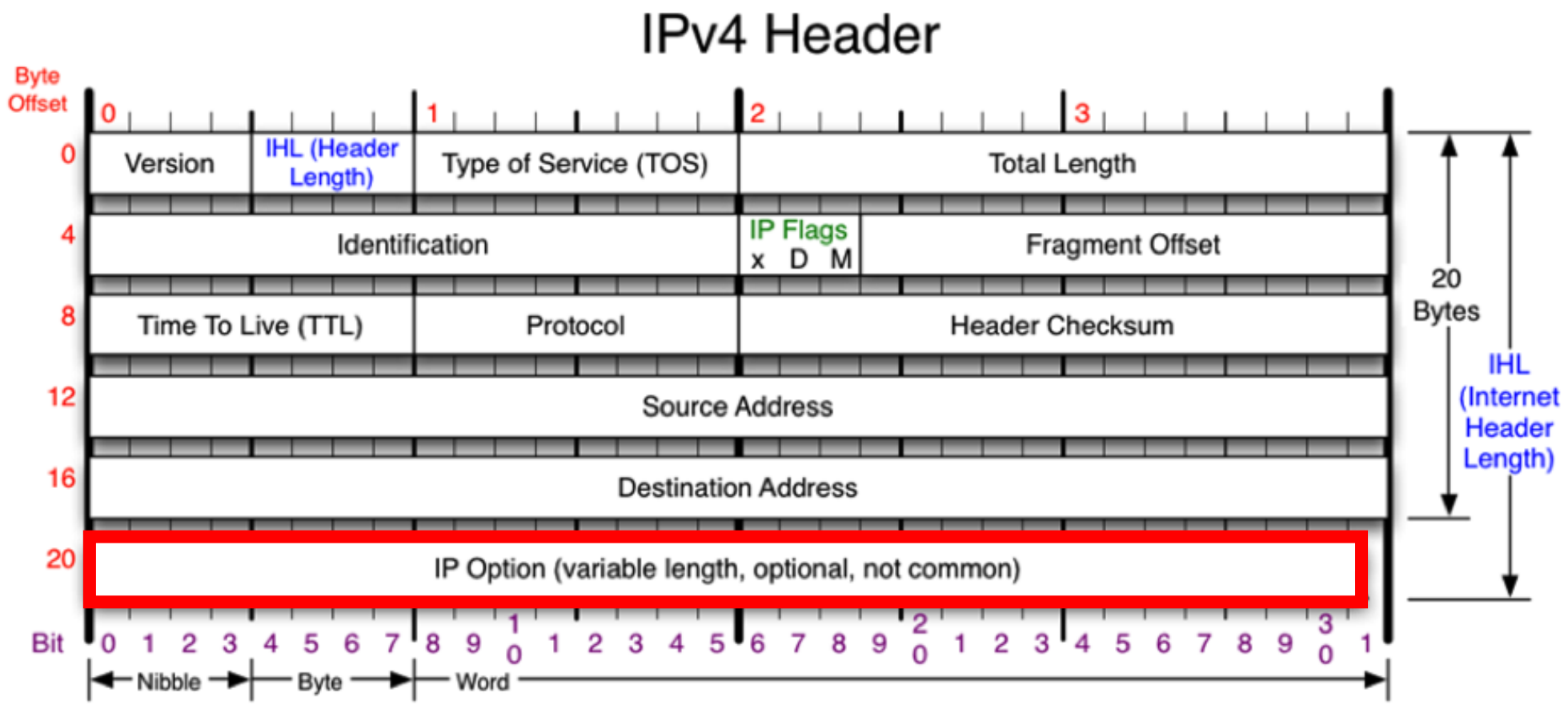
TCP Urgent Pointer connect-back state confusion
(CVE-2019-12261)


IP Options parsing stack overflow
(CVE-2019-12256)
TCP Urgent Pointer TCP AO state confusion
(CVE-2019-12255)
TCP Urgent Pointer 0 integer underflow
(CVE-2019-12255)
TCP Urgent Pointer race condition
(CVE-2019-12263)
Reverse ARP logical flaw
(CVE-2019-12262)
URGENT/11
TCP connection DoS via malformed options
(CVE-2019-12258)
DHCP response heap overflow
(CVE-2019-12257)
IGMP Information leak
(CVE-2019-12265)
DHCP client IPv4 assignment logical flaw
(CVE-2019-12264)
IGMP NULL dereference
(CVE-2019-12259)
TCP Urgent Pointer connect-back state confusion
(CVE-2019-12261)

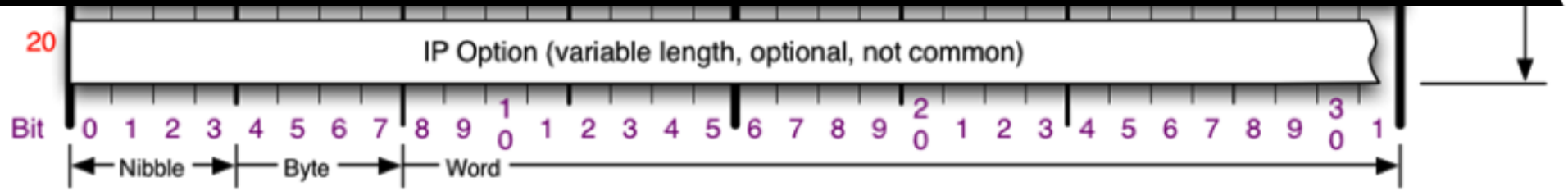
IP Options

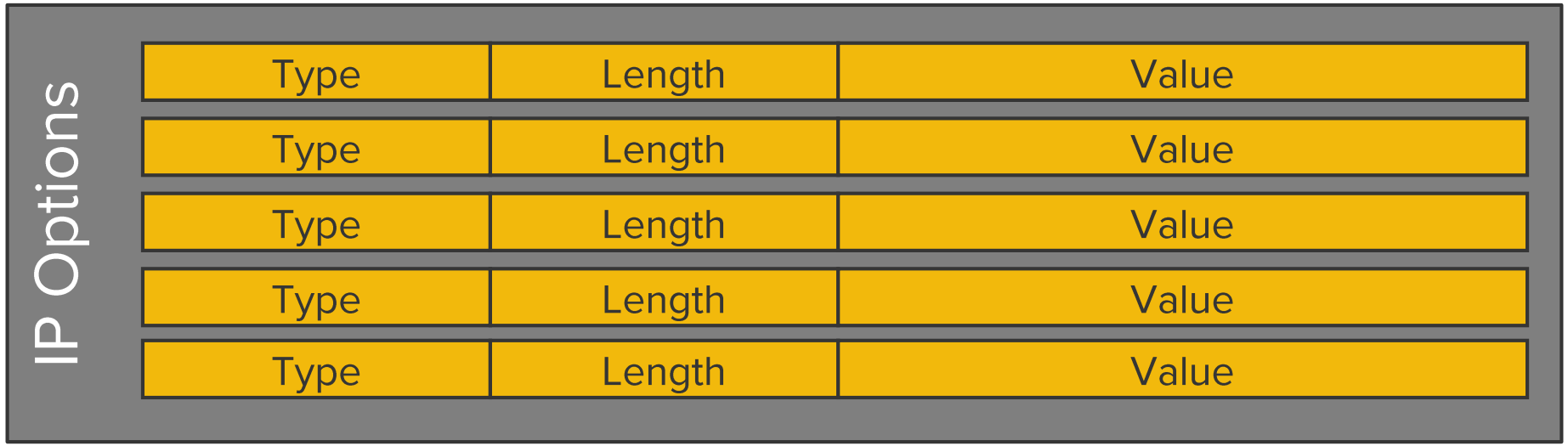

IPv4 Header
Byte Offset
0
4
8
12
16
20
0
1
2
3
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
20 Bytes
IHL (Internet Header Length)
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word

IP Options

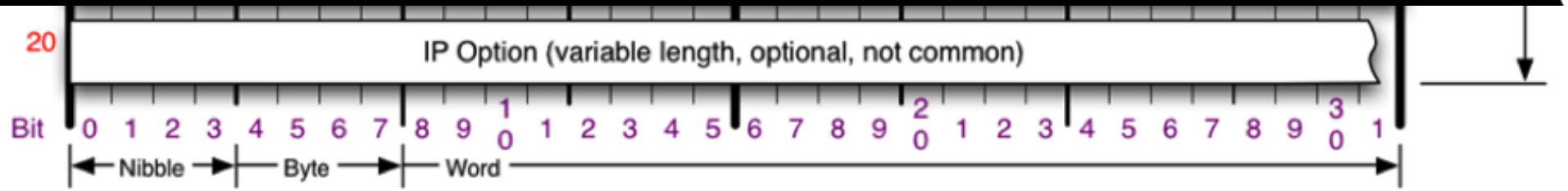

IPv4 Header
Byte Offset
0
4
8
12
16
20
0
1
2
3
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
20 Bytes
IHL (Internet Header Length)
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word

IP Options


20
IP Option (variable length, optional, not common)
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word


IP Options
Type
Length
Value
Type
Length
Value
Type
Length
Value
Type
Length
Value
Type
Length
Value

IP Source Routing



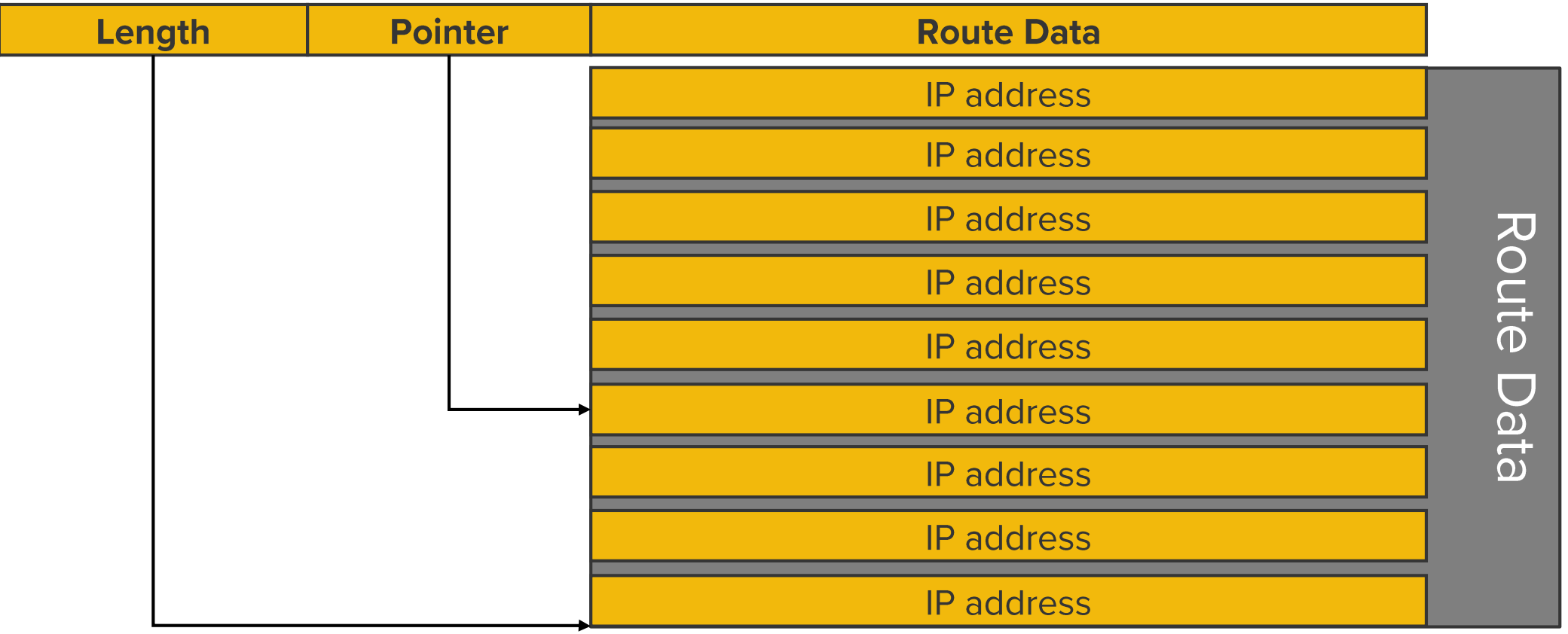
IP Options

SSRR	Length	Pointer	Route Data

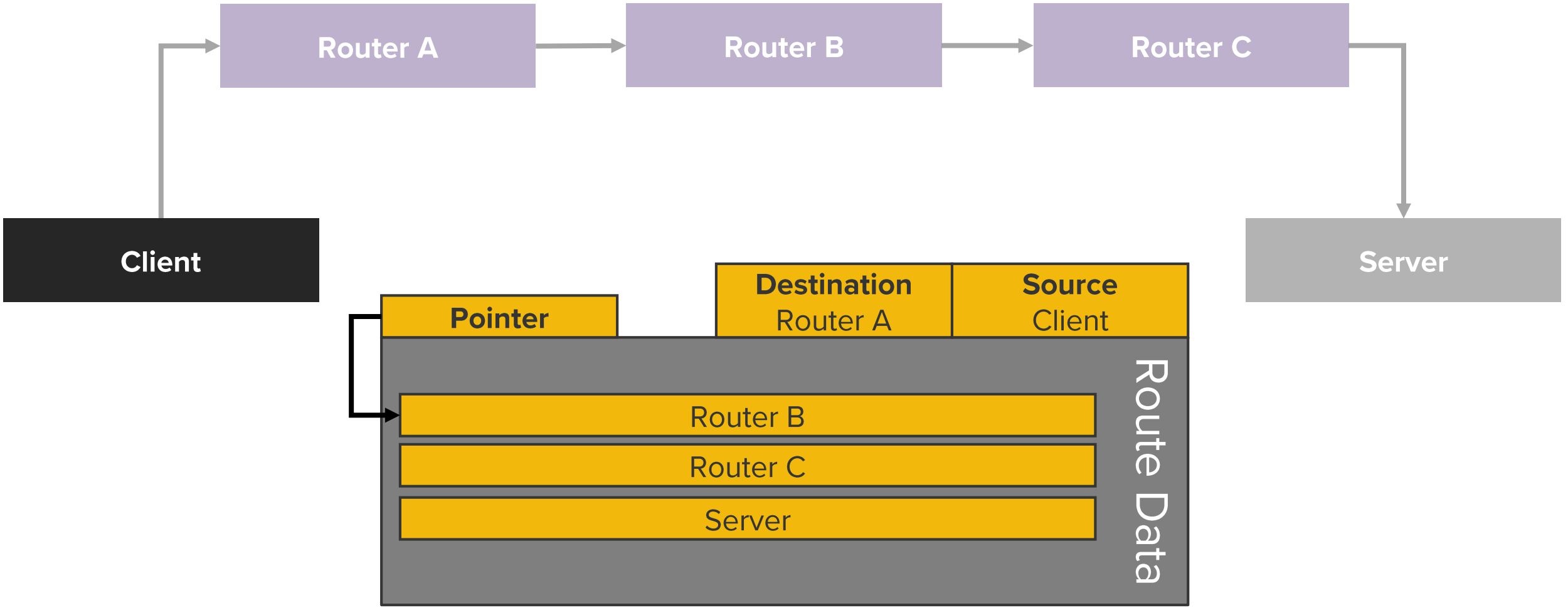
Strict Source and Record Route

This option lets the originating system specify a number of intermediate systems a packet must pass through to get to the destination host

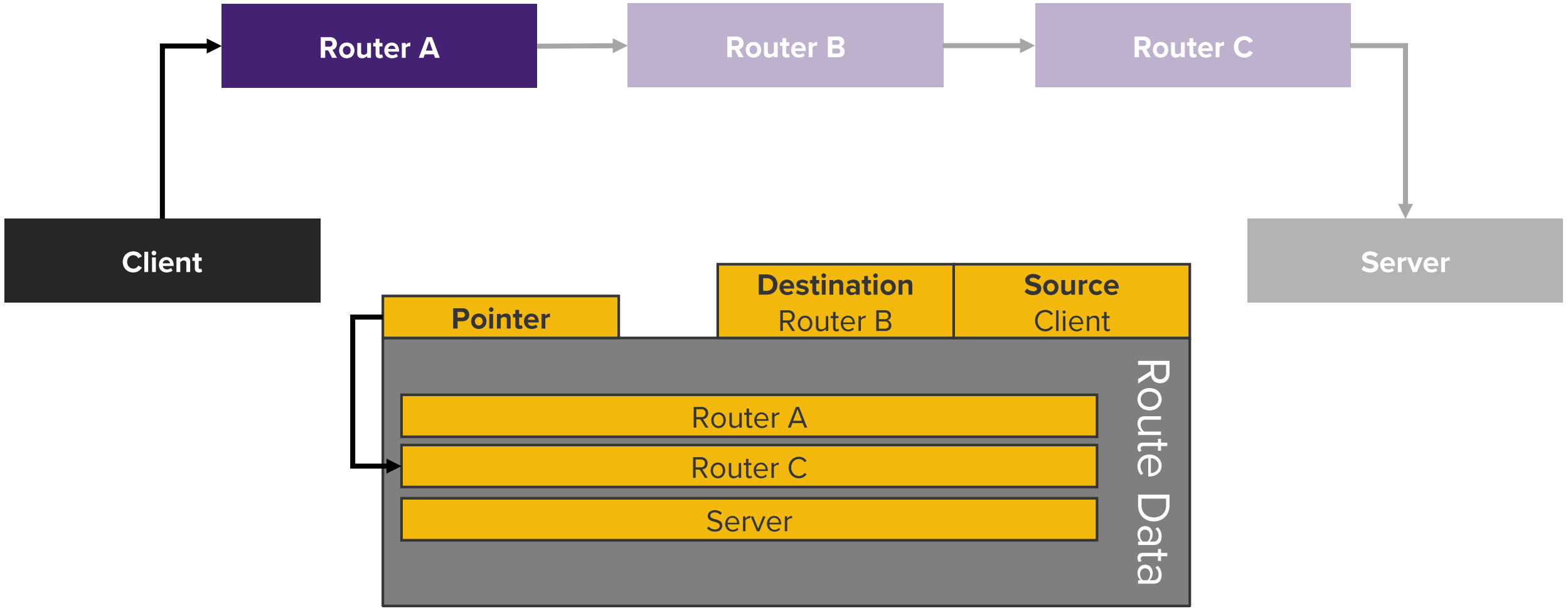
SSRR IP Option



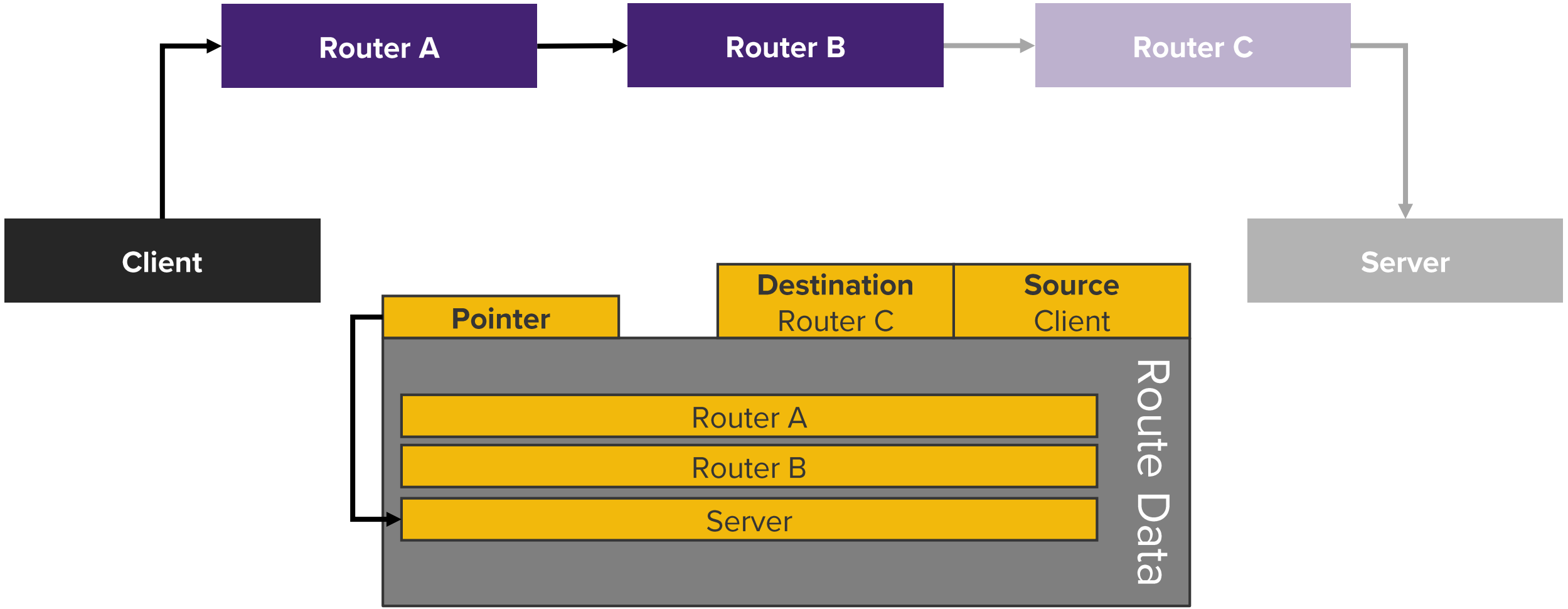
SSRR


Router A
Router B
Router C
Client
Server
Pointer
Destination
Router A
Source
Client
Router B
Router C
Server
Route Data

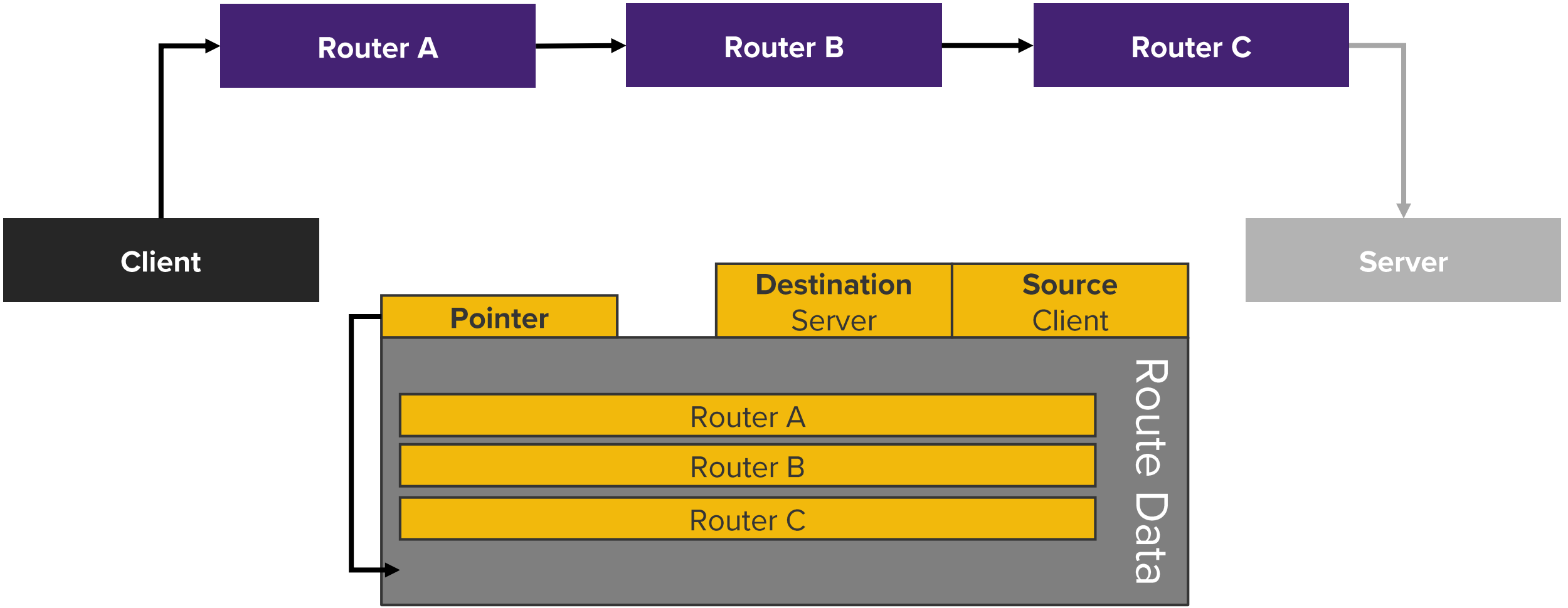
SSRR


Router A
Router B
Router C
Client
Server
Pointer
Destination
Router B
Source
Client
Router A
Router C
Server
Route Data

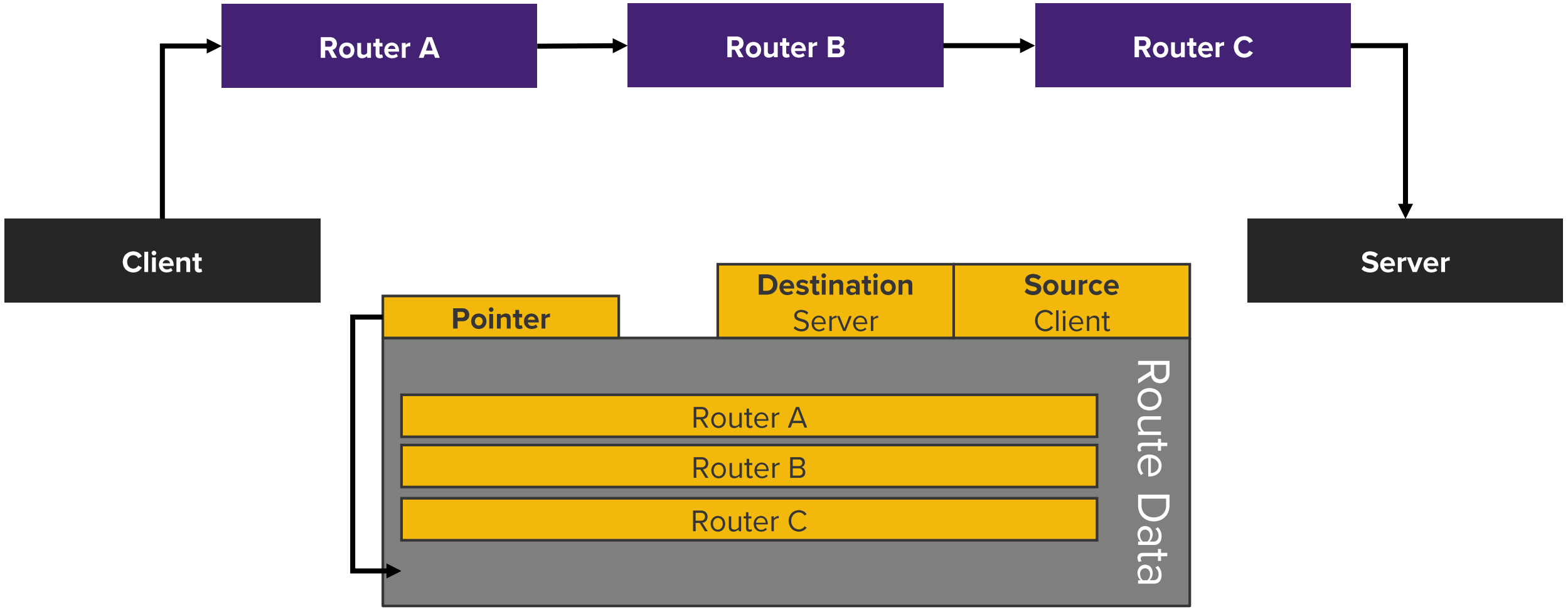
SSRR


Router A
Router B
Router C
Client
Server
Pointer
Destination
Router C
Source
Client
Router A
Router B
Server
Route Data

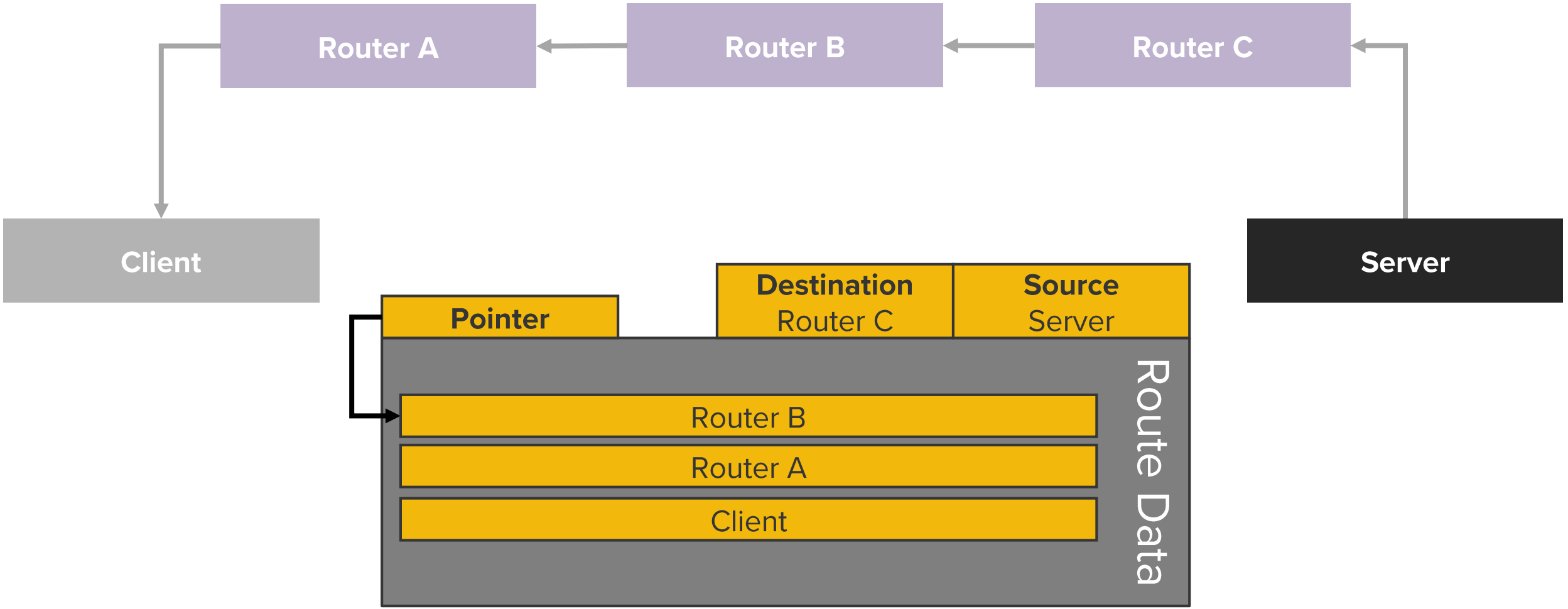
SSRR


Router A
Router B
Router C
Client
Server
Pointer
Destination
Server
Source
Client
Router A
Router B
Router C
Route Data

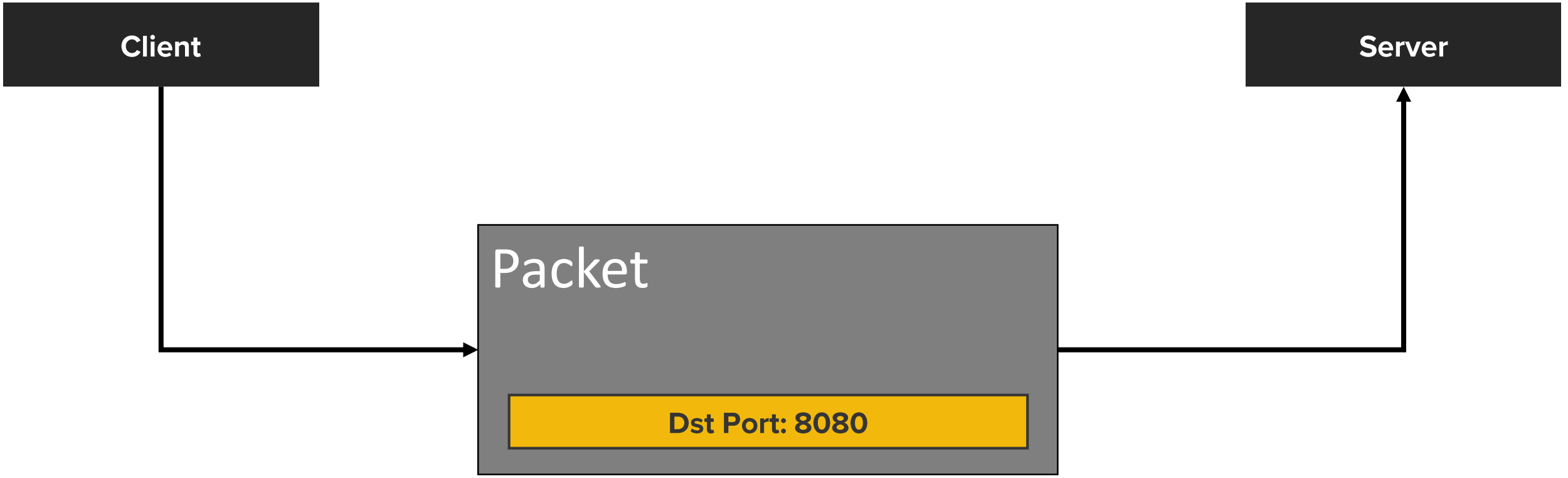
SSRR


Router A
Router B
Router C
Client
Server
Pointer
Destination
Server
Source
Client
Router A
Router B
Router C
Route Data

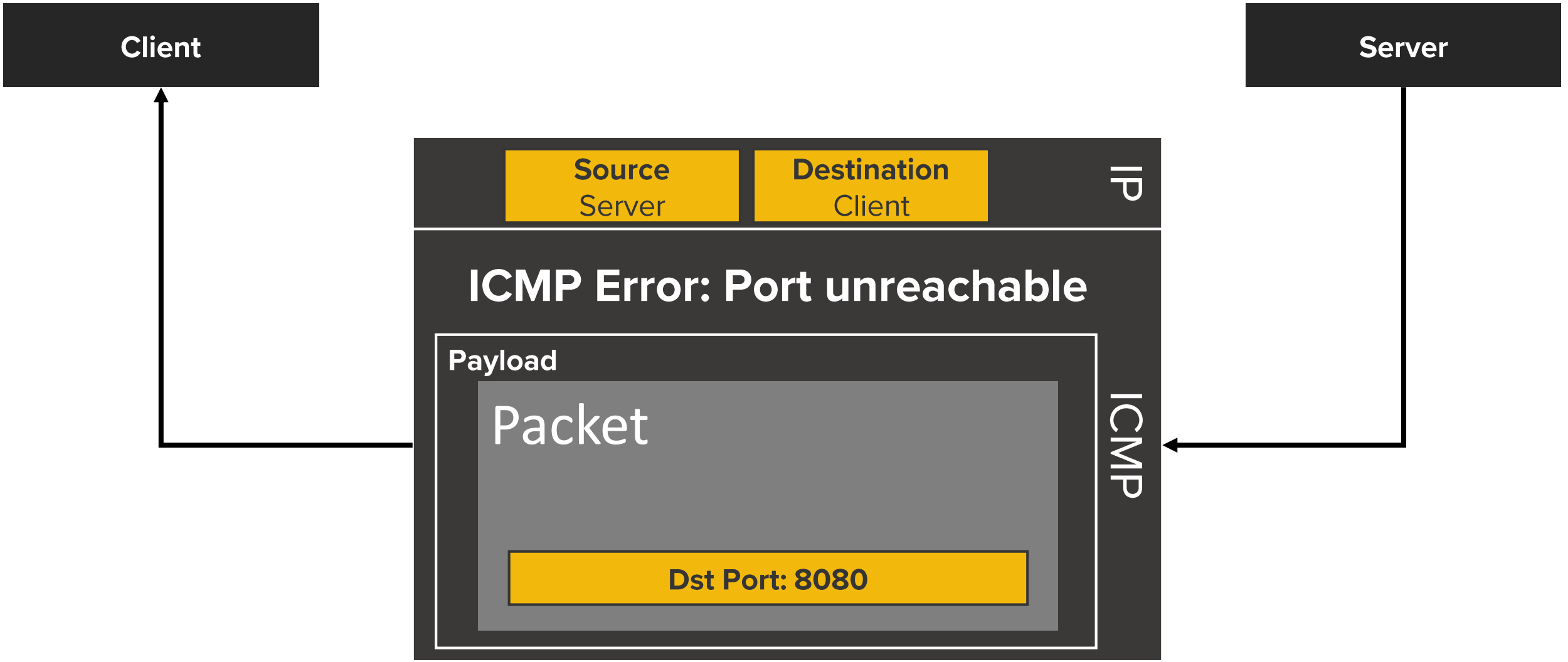
SSRR


Router A
Router B
Router C
Client
Server
Pointer
Destination
Router C
Source
Server
Router B
Router A
Client
Route Data

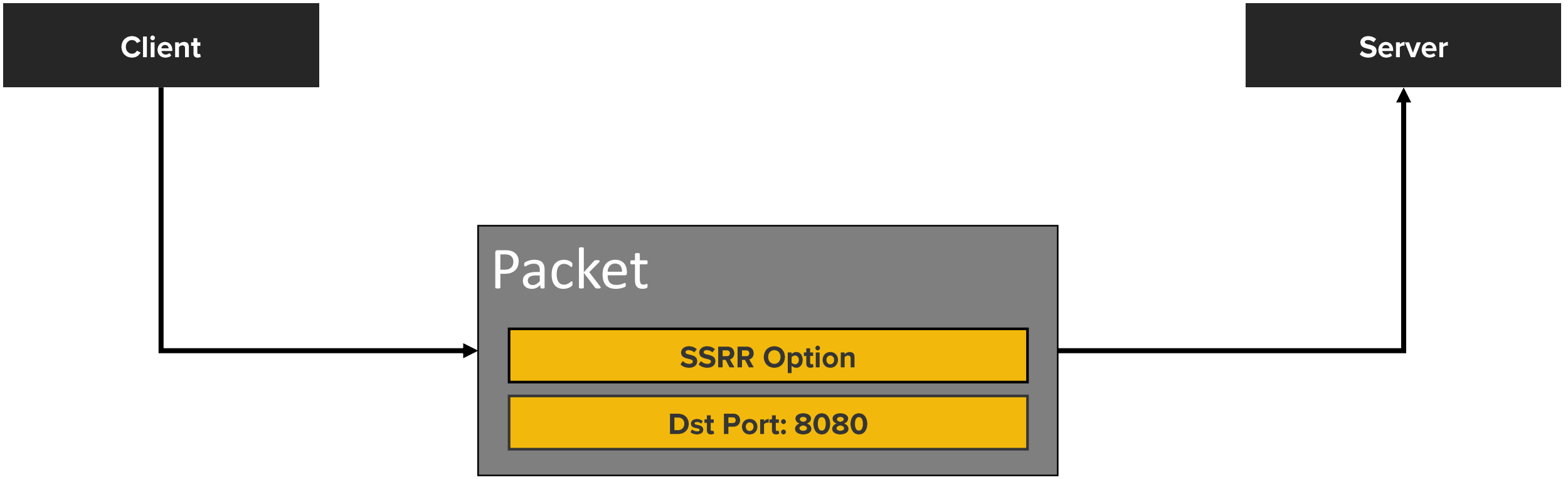
ICMP Errors



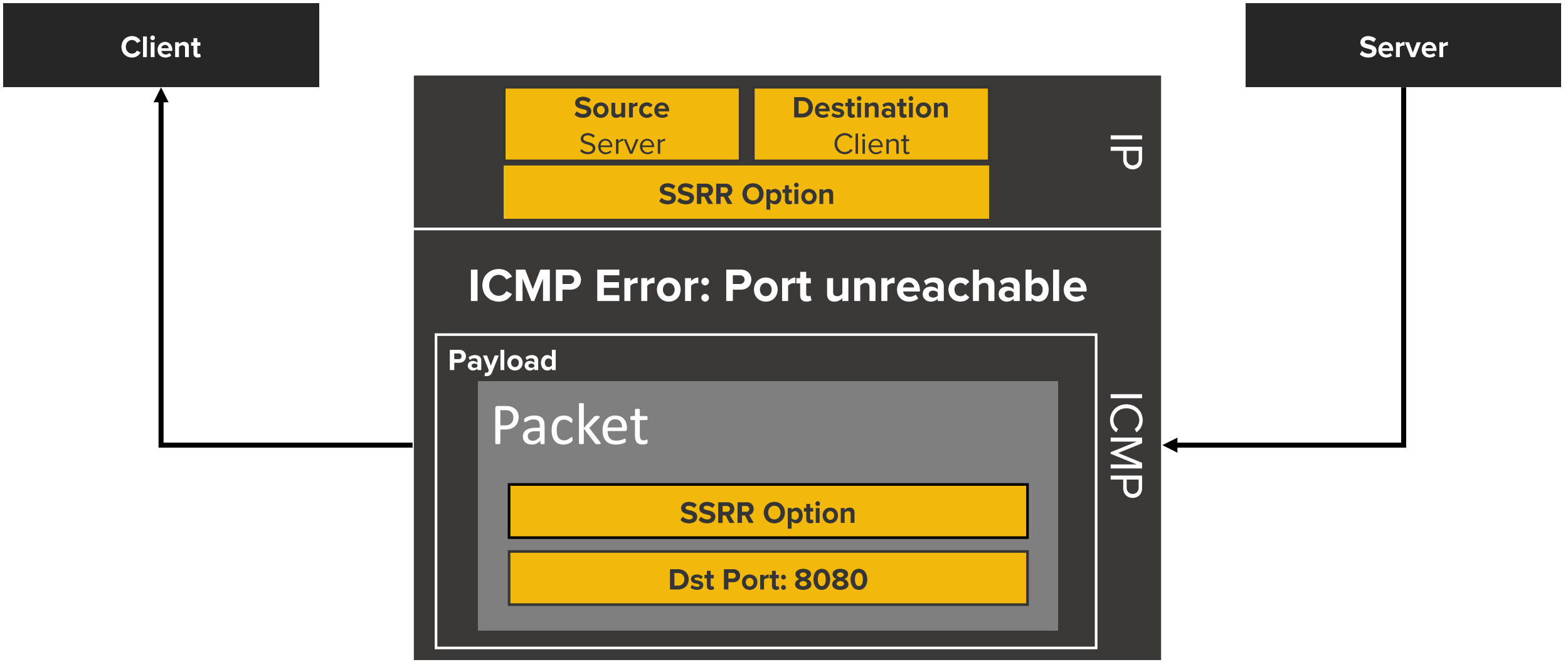
ICMP Errors


Client
Server
Source
Server
Destination
Client
IP
ICMP Error: Port unreachable
Payload
Packet
Dst Port: 8080
ICMP

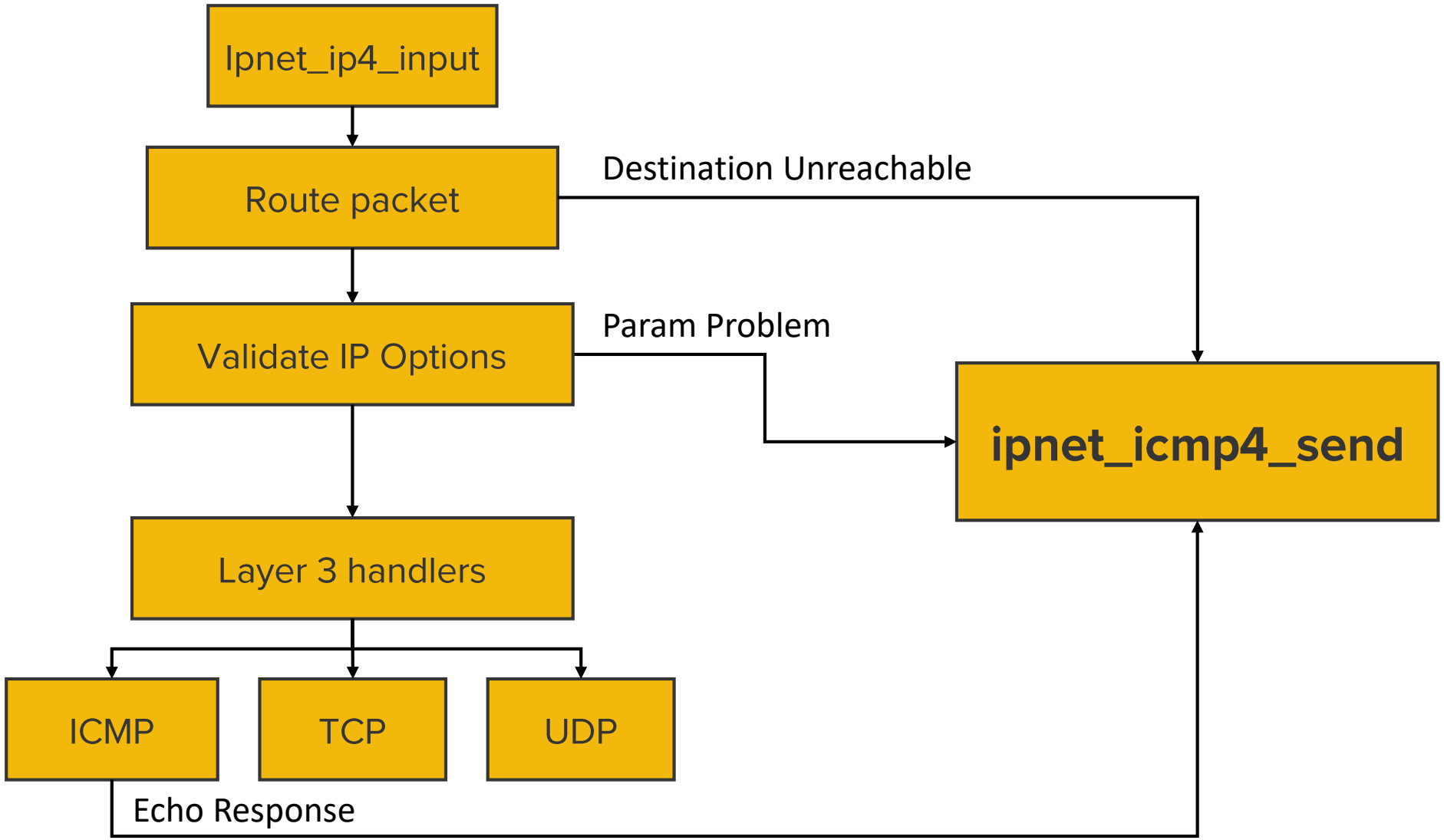
ICMP Errors



ICMP Errors


Client
Server
Source
Server
Destination
Client
SSRR Option
IP
ICMP Error: Port unreachable
Payload
Packet
SSRR Option
Dst Port: 8080
ICMP

IP Packet flow



ipnet_icmp4_send

```
while ( 1 ) {
    current_opt = ipnet_ip4_get_ip_opt_next(current_opt, copyopts_param->options_ptr,
                                            copyopts_param->total_opt_size);
    ...
    opt_type = *current_opt;
    if ( opt_type == IP_IPOPT_SSRR ) {
      srr_opt = (srr_opt_t *)&opts->opts[opts->len];
      // Doesn't validate this offset is within the current option
      offset_to_current_route_entry = srr_opt->ptr - 5;
      ...
      current_route_entry = &current_opt[offset_to_current_route_entry];
      while ( offset_to_current_route_entry > 0 ) {
        memcpy((char *)srr_opt + srr_opt->length, current_route_entry, 4);
        current_route_entry -= 4;
        offset_to_current_route_entry -= 4;
        srr_opt->length += 4;
      }
   }
```

IP Options parsing stack overflow
(CVE-2019-12256)

TCP Urgent Pointer TCP AO state confusion
(CVE-2019-12255)

TCP Urgent Pointer 0 integer underflow
(CVE-2019-12255)

TCP Urgent Pointer race condition
(CVE-2019-12263)

Reverse ARP logical flaw
(CVE-2019-12262)

TCP connection DoS via malformed options
(CVE-2019-12258)

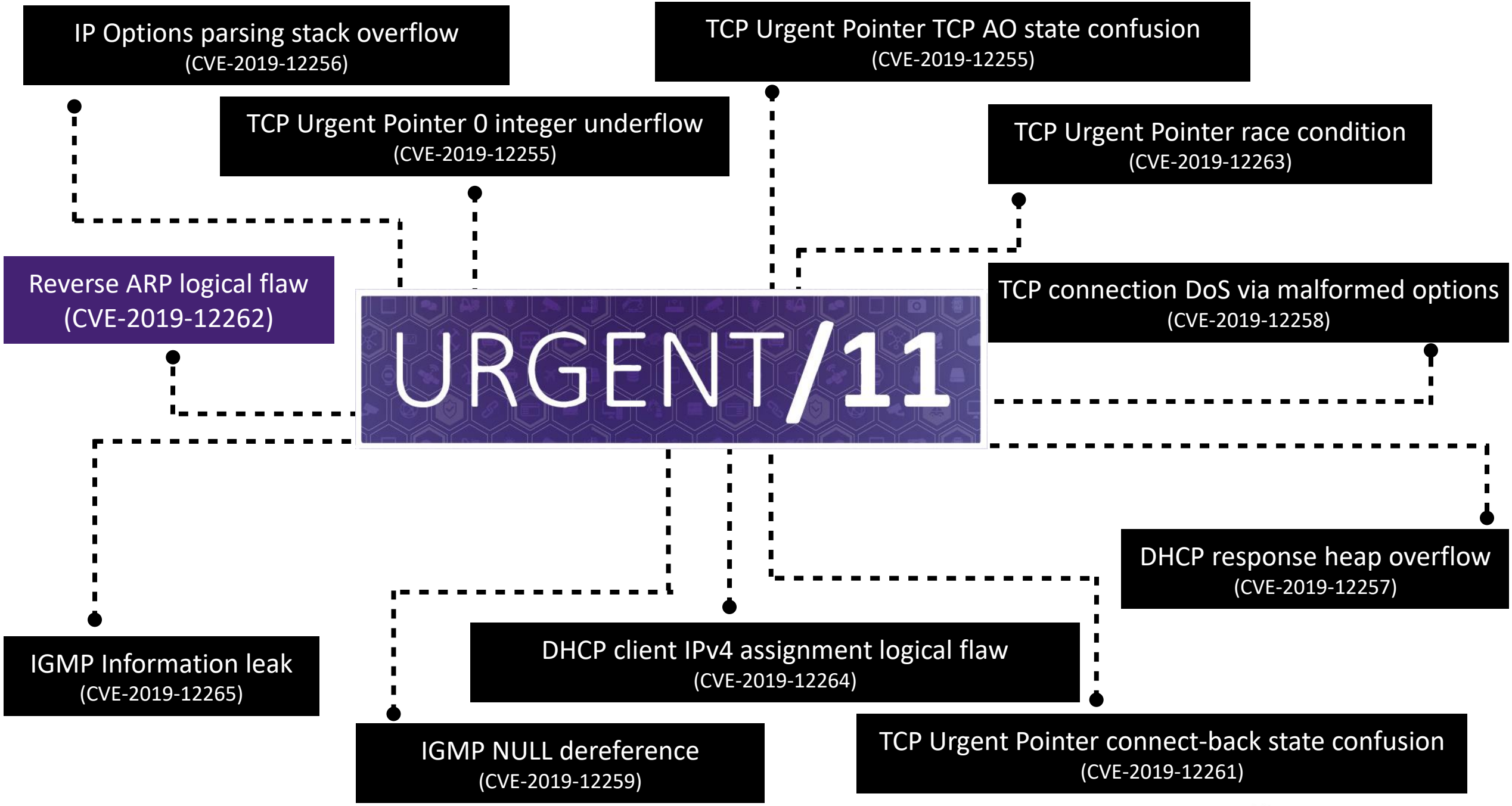
DHCP response heap overflow
(CVE-2019-12257)

IGMP Information leak
(CVE-2019-12265)

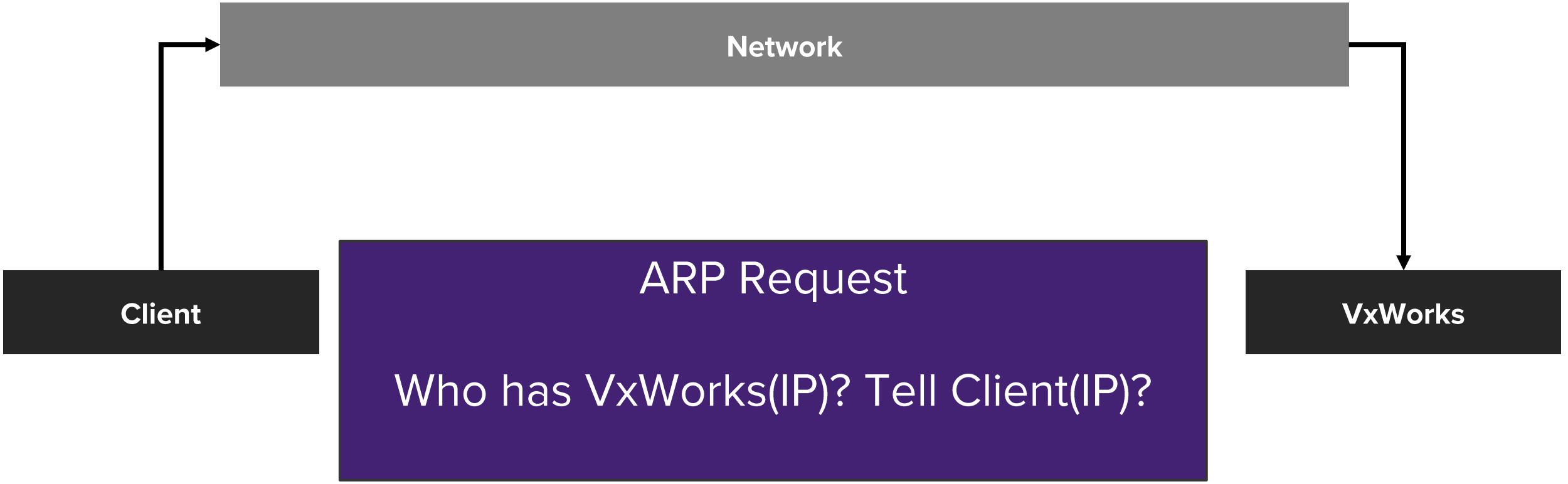
DHCP client IPv4 assignment logical flaw
(CVE-2019-12264)

IGMP NULL dereference
(CVE-2019-12259)

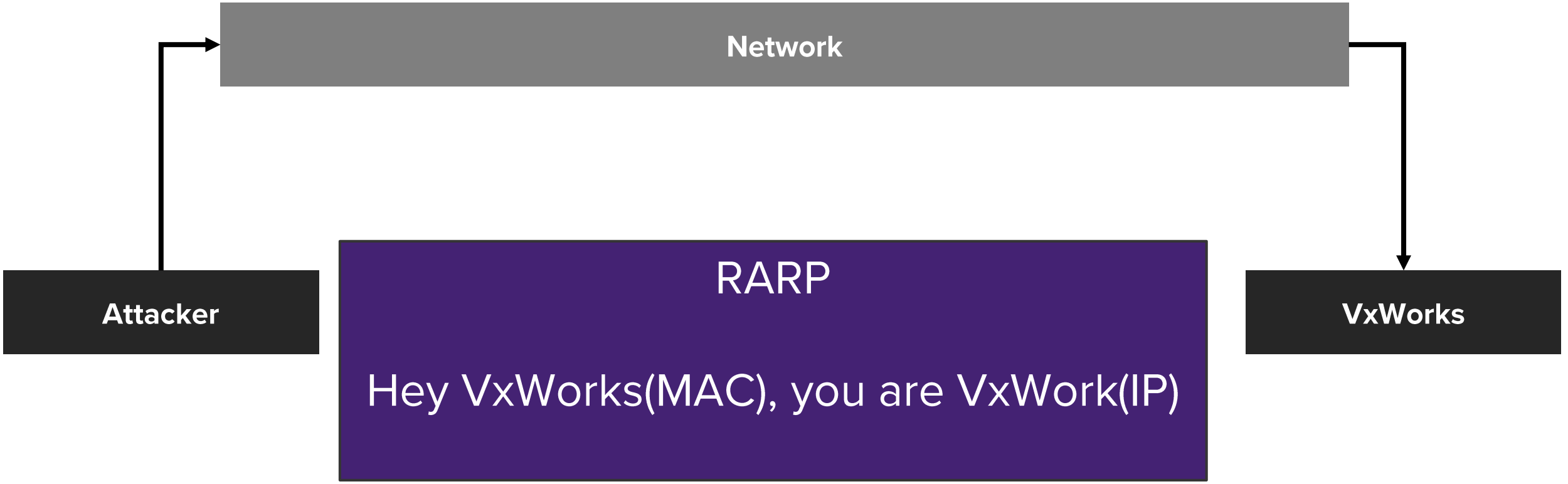
TCP Urgent Pointer connect-back state confusion
(CVE-2019-12261)


IP Options parsing stack overflow
(CVE-2019-12256)
TCP Urgent Pointer TCP AO state confusion
(CVE-2019-12255)
TCP Urgent Pointer 0 integer underflow
(CVE-2019-12255)
TCP Urgent Pointer race condition
(CVE-2019-12263)
Reverse ARP logical flaw
(CVE-2019-12262)
URGENT/11
TCP connection DoS via malformed options
(CVE-2019-12258)
DHCP response heap overflow
(CVE-2019-12257)
IGMP Information leak
(CVE-2019-12265)
DHCP client IPv4 assignment logical flaw
(CVE-2019-12264)
IGMP NULL dereference
(CVE-2019-12259)
TCP Urgent Pointer connect-back state confusion
(CVE-2019-12261)

Your run-of-the-mill ARP



RARP!


Network
Attacker
RARP
Hey VxWorks(MAC), you are VxWork(IP)
VxWorks

RARP!

RARP!

```
...
if ( *(_WORD *)rarp_pkt == 256
  && *(_WORD *)(rarp_pkt + 2) == 8
  && *(_BYTE *)(rarp_pkt + 4) == 6
  && *(_BYTE *)(rarp_pkt + 5) == 4
  && *(_WORD *)(rarp_pkt + 6) == 1024 )
{
  interface = ipnet_eth_is_valid_node_mac(interface, rarp_pkt + 8);
  if ( interface )
  {
    // Validate the IP address isn't class D or E
    if ( !(*(_WORD *)(rarp_pkt + 24) & 0x80)
      || (*(_WORD *)(rarp_pkt + 24) & 0xC0) == 128
      || (interface = *(_WORD *)(rarp_pkt + 24) & 0xE0, interface == 192) )
    {
      ret = ipnet_ip4_add_addr(interface);
...
```

IP Options parsing stack overflow
(CVE-2019-12256)

TCP Urgent Pointer TCP AO state confusion
(CVE-2019-12255)

TCP Urgent Pointer 0 integer underflow
(CVE-2019-12255)

TCP Urgent Pointer race condition
(CVE-2019-12263)

Reverse ARP logical flaw
(CVE-2019-12262)

TCP connection DoS via malformed options
(CVE-2019-12258)

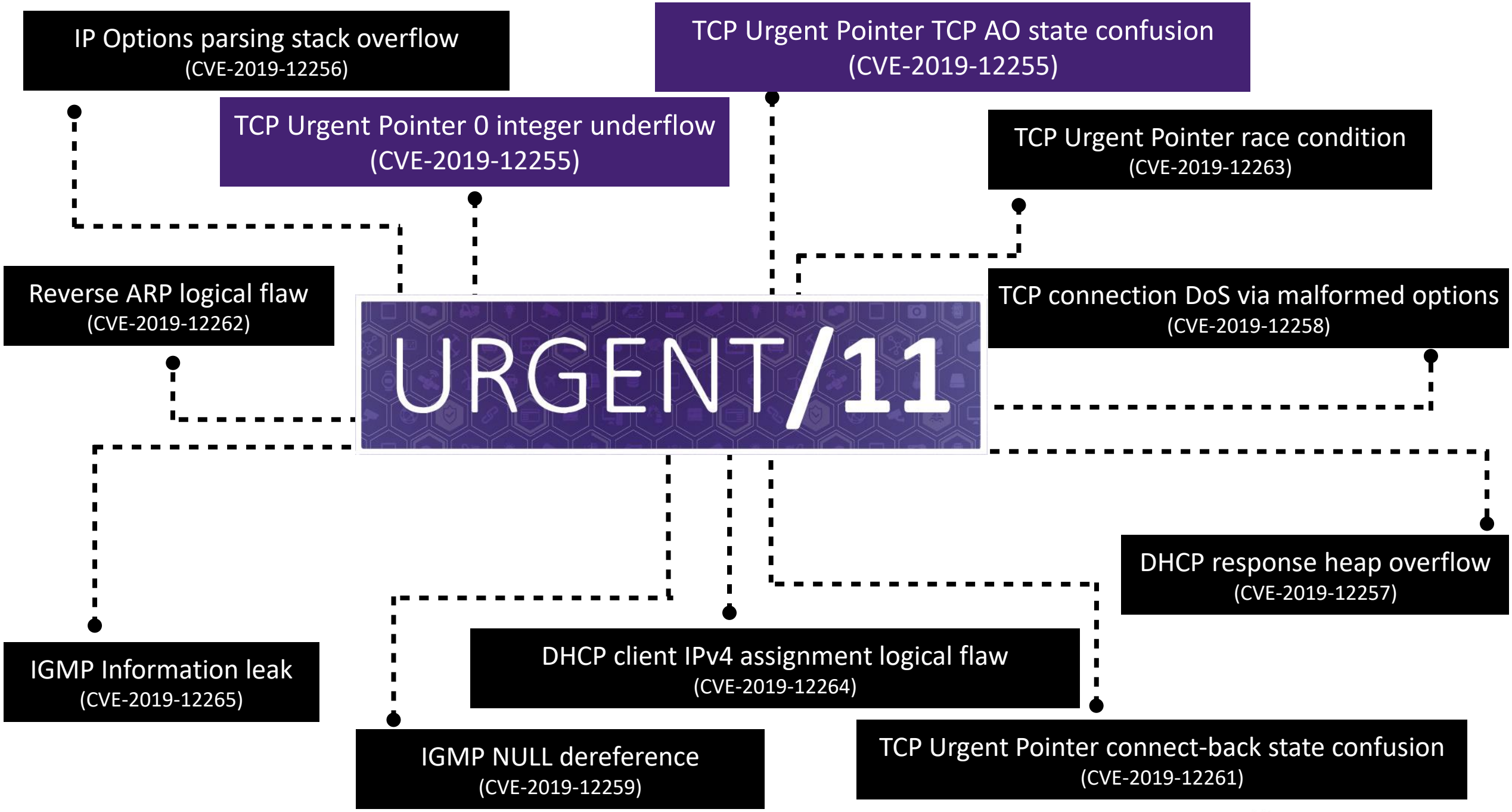
DHCP response heap overflow
(CVE-2019-12257)

IGMP Information leak
(CVE-2019-12265)

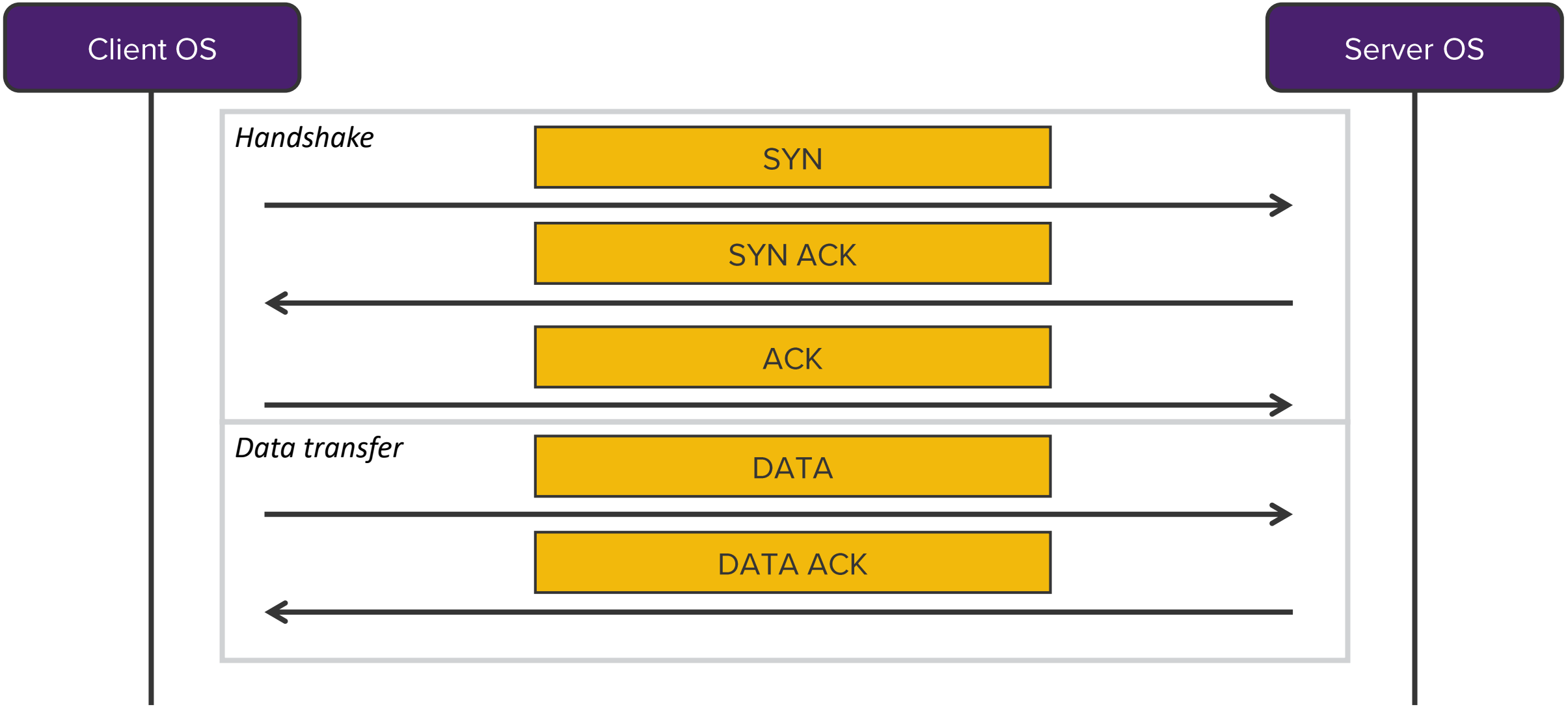
DHCP client IPv4 assignment logical flaw
(CVE-2019-12264)

IGMP NULL dereference
(CVE-2019-12259)

TCP Urgent Pointer connect-back state confusion
(CVE-2019-12261)


IP Options parsing stack overflow
(CVE-2019-12256)
TCP Urgent Pointer TCP AO state confusion
(CVE-2019-12255)
TCP Urgent Pointer 0 integer underflow
(CVE-2019-12255)
TCP Urgent Pointer race condition
(CVE-2019-12263)
Reverse ARP logical flaw
(CVE-2019-12262)
URGENT/11
TCP connection DoS via malformed options
(CVE-2019-12258)
DHCP response heap overflow
(CVE-2019-12257)
IGMP Information leak
(CVE-2019-12265)
DHCP client IPv4 assignment logical flaw
(CVE-2019-12264)
IGMP NULL dereference
(CVE-2019-12259)
TCP Urgent Pointer connect-back state confusion
(CVE-2019-12261)

TCP Connection



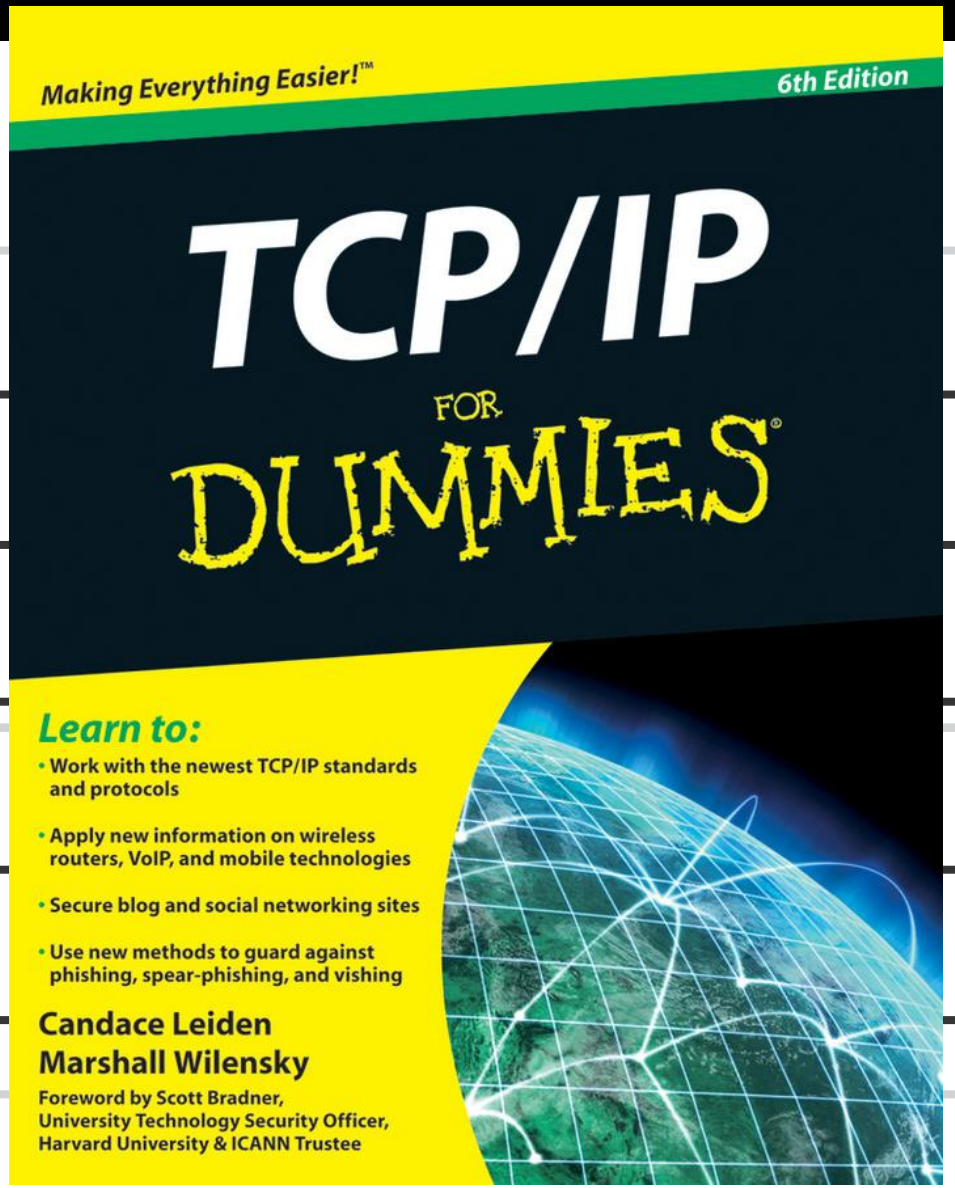
TCP Connection

Client OS

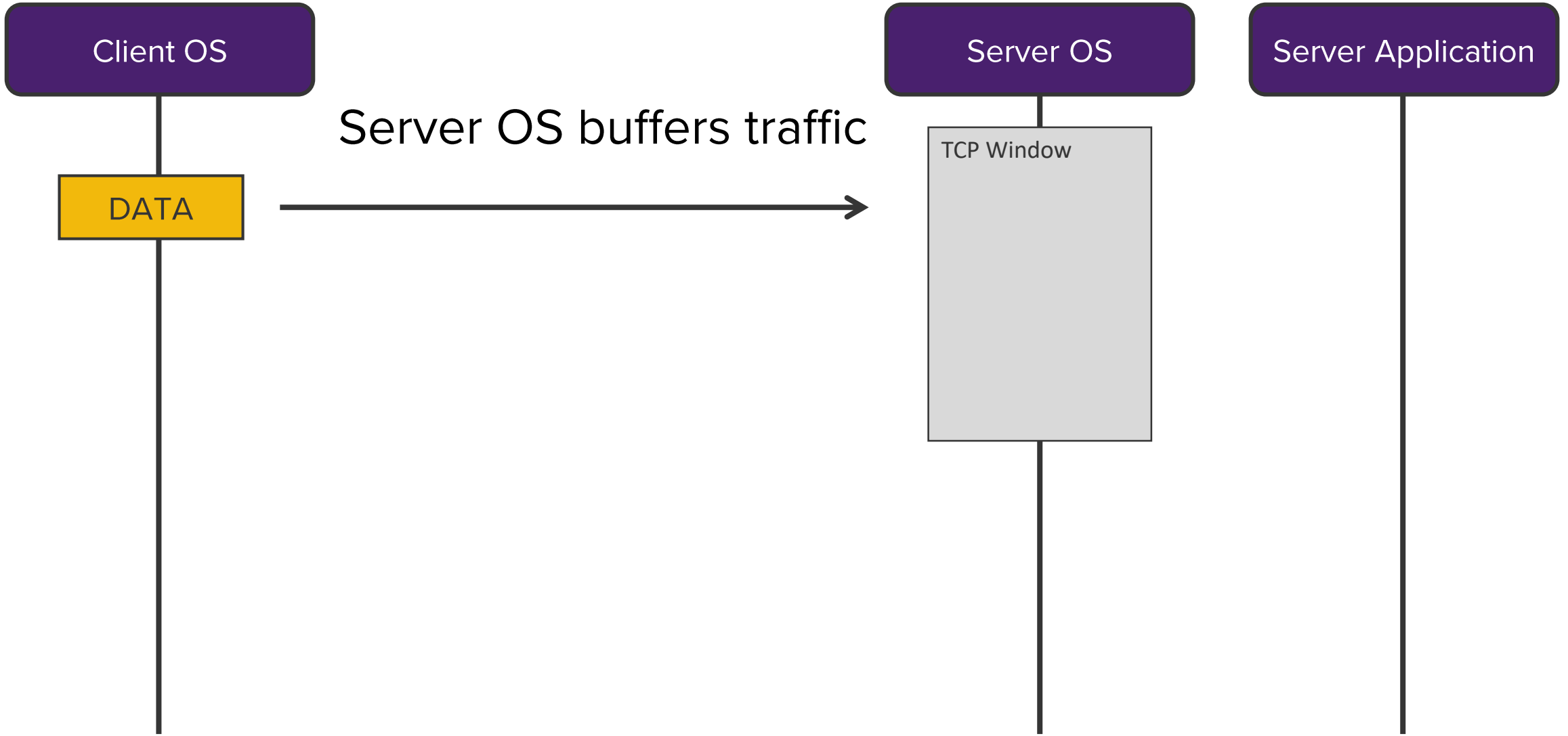
Server OS

Handshake

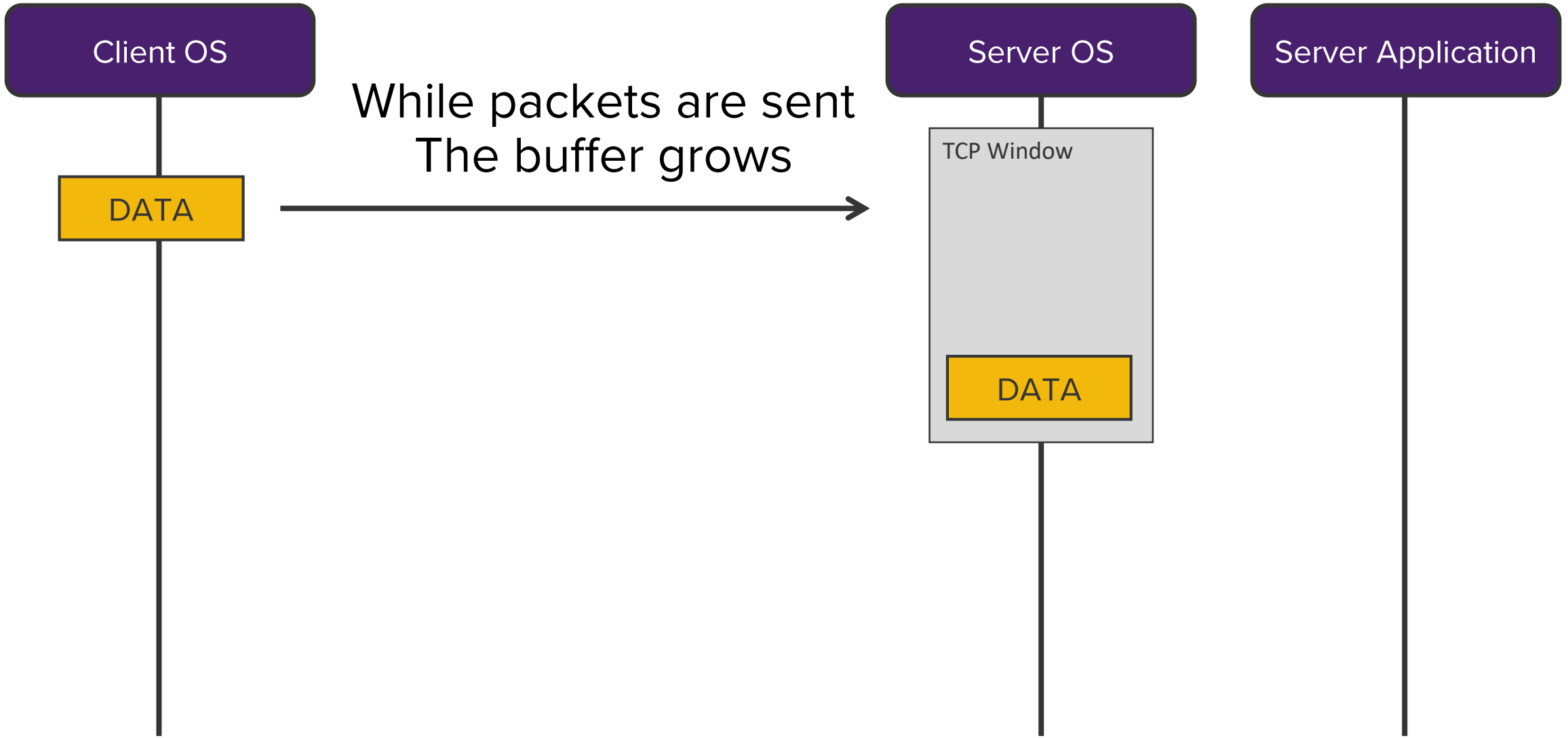
Data transfer



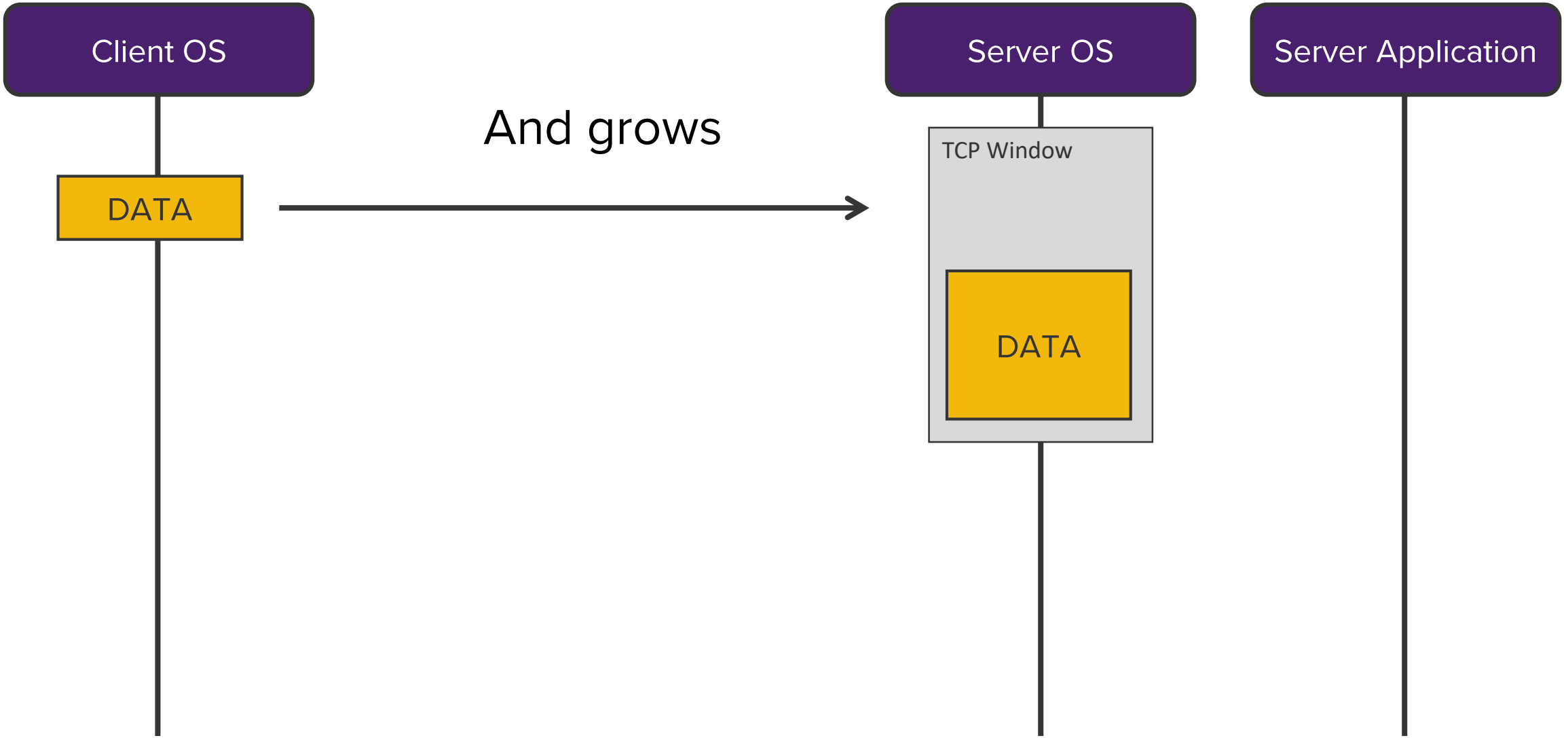
TCP Window


Client OS
Server OS
Server Application
Server OS buffers traffic
DATA
TCP Window

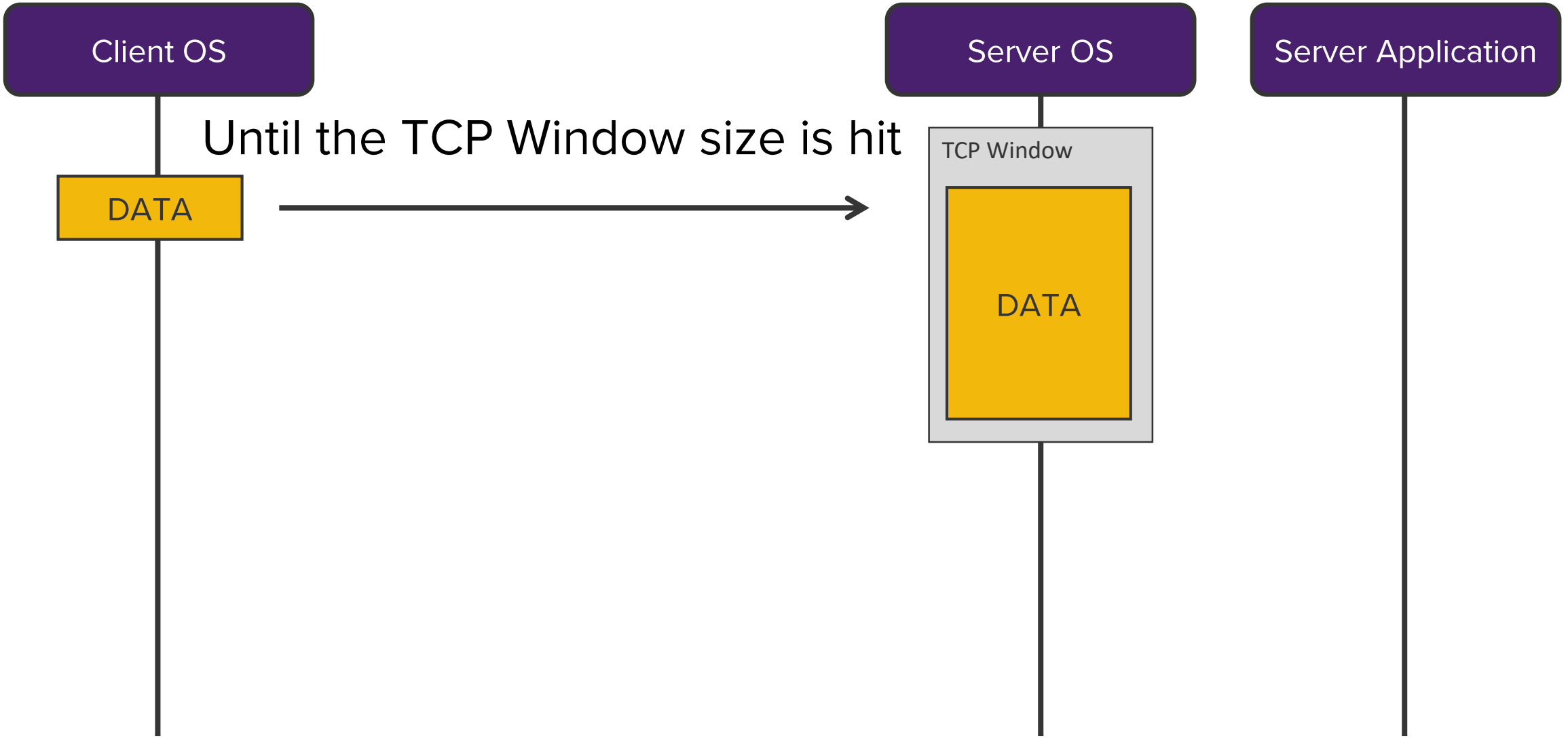
TCP Window


Client OS
Server OS
Server Application
While packets are sent
The buffer grows
DATA
TCP Window
DATA

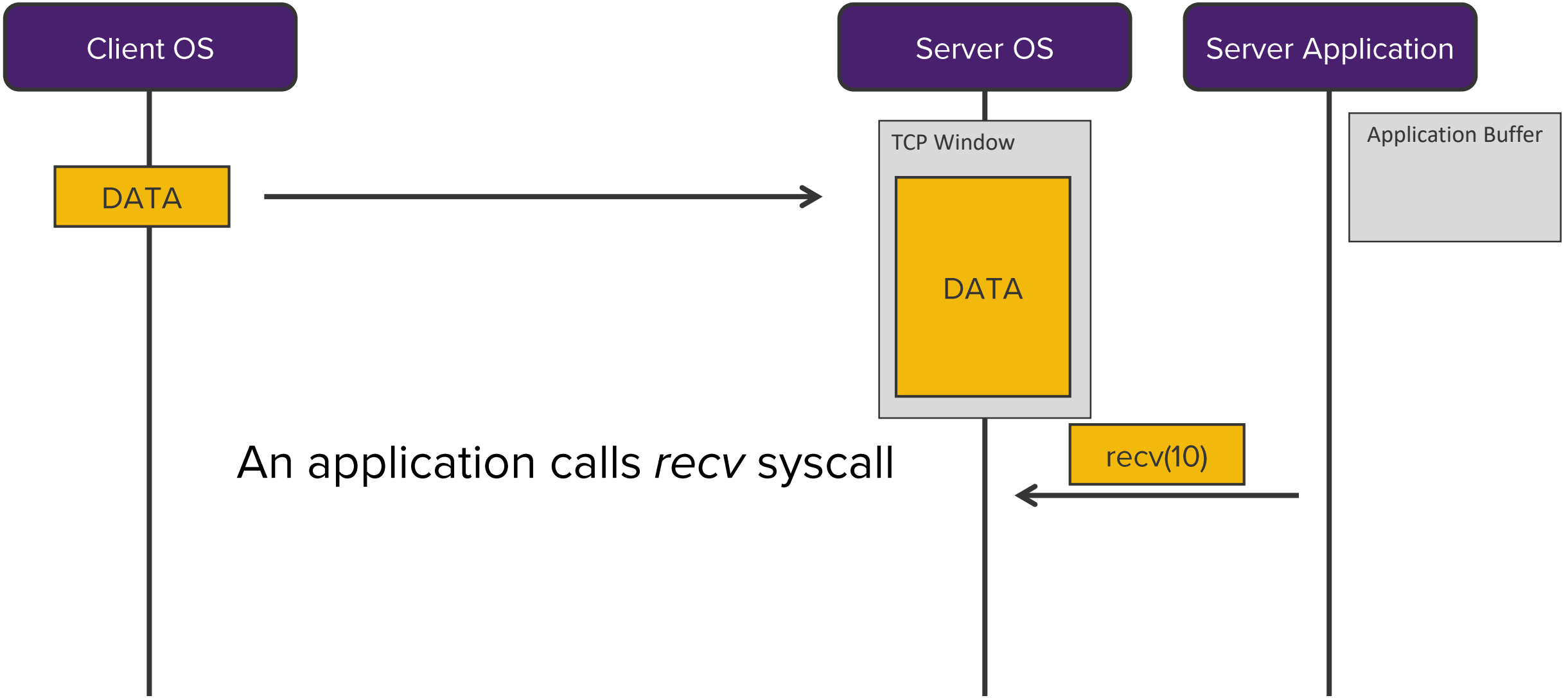
TCP Window


Client OS
Server OS
Server Application
DATA
And grows
TCP Window
DATA

TCP Window

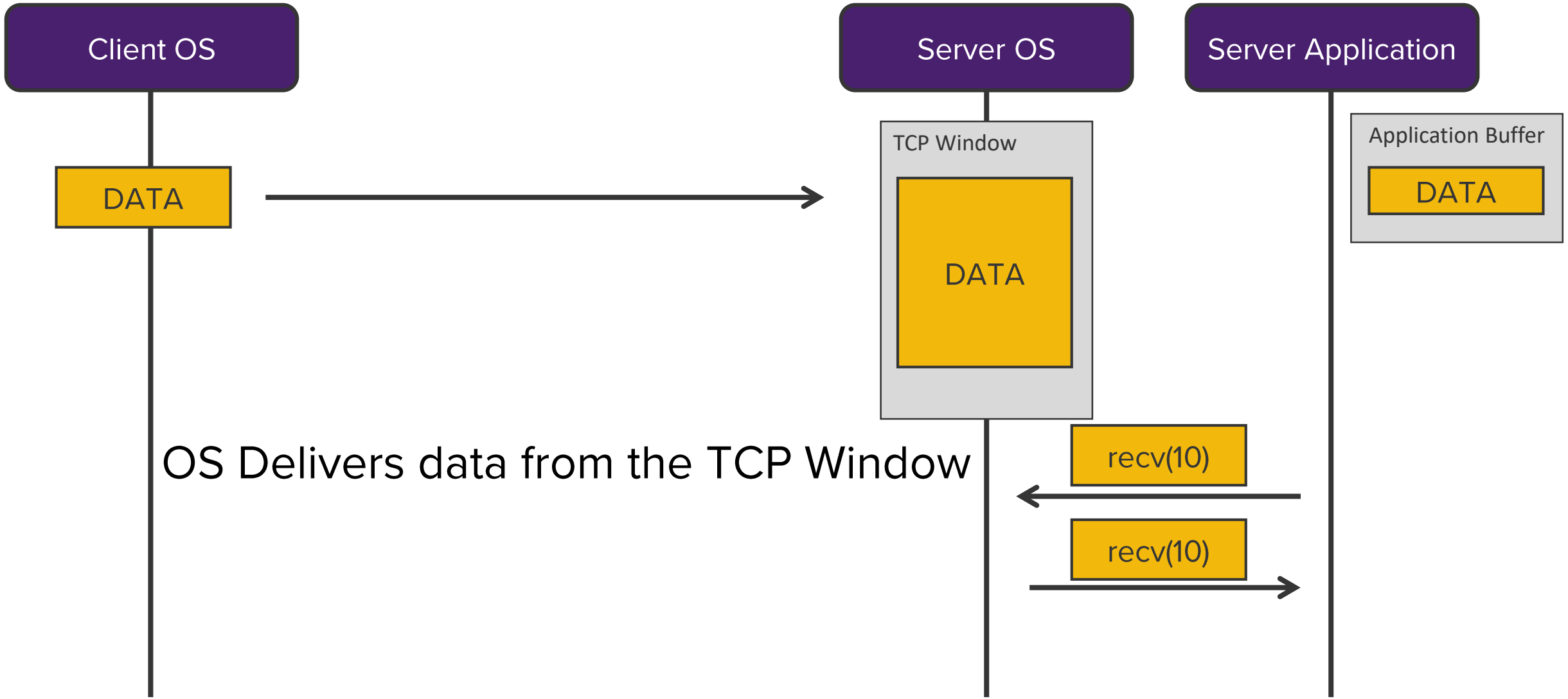

Client OS
Server OS
Server Application
Until the TCP Window size is hit
DATA
TCP Window
DATA

TCP Window



An application calls *recv* syscall

TCP Window

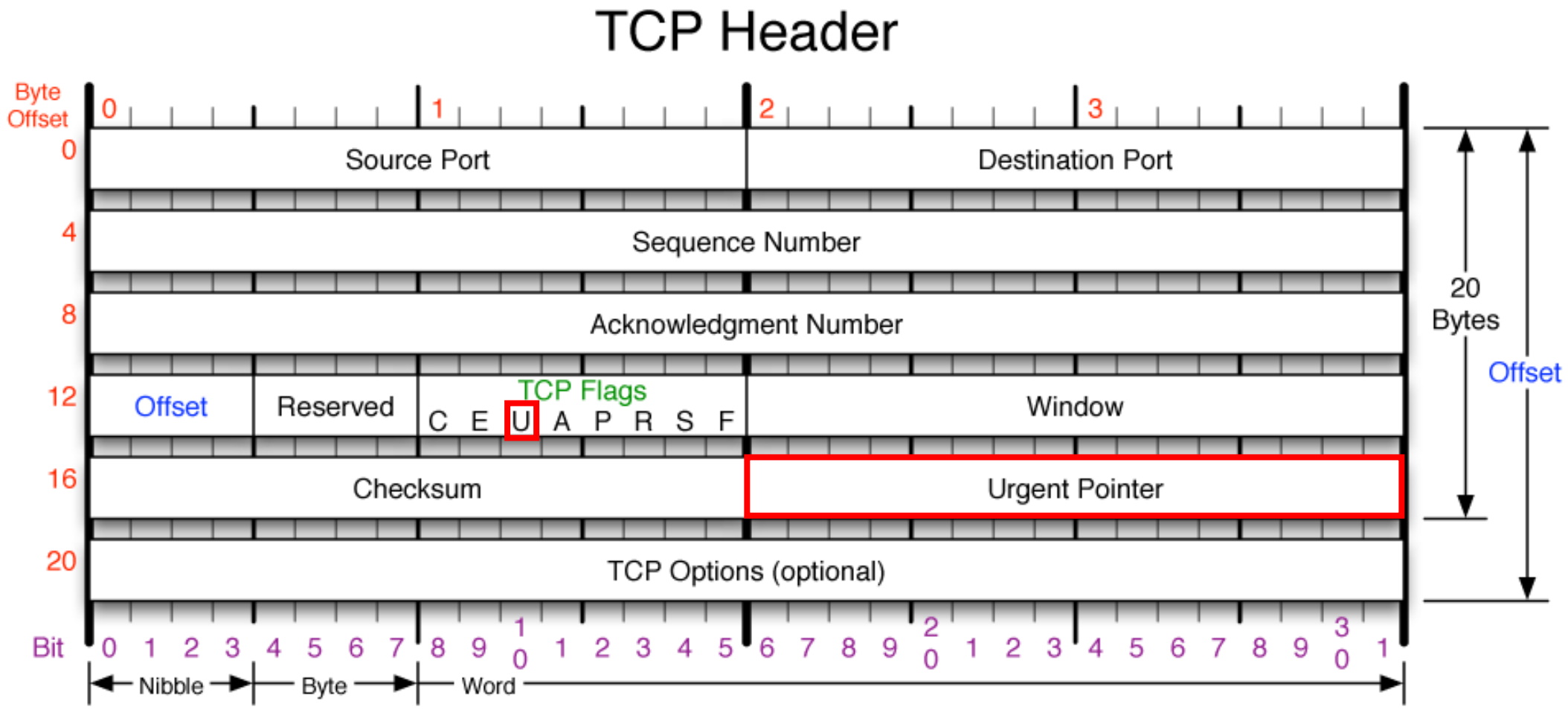


OS Delivers data from the TCP Window

TCP Urgent data

“.. If all this sounds complicated just ***don't use urgent data****”*

cnicutar, stackoverflow.com
(first result on google “tcp urgent data”)


TCP Header
Byte Offset
0
1
2
3
Source Port
Destination Port
4
Sequence Number
8
Acknowledgment Number
12
Offset
Reserved
TCP Flags
C E U A P R S F
Window
16
Checksum
Urgent Pointer
20
TCP Options (optional)
20 Bytes
Offset
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word

RFC 793 (page 17)

The urgent pointer points to the sequence number of the octet following the urgent data.

1981

IETF Group Chat

RFC 793 (page 17)

The urgent pointer points to the sequence number of the octet following the urgent data.

1981

RFC 1011 (page 8)

Page 17 is wrong. The urgent pointer points to the last octet of urgent data (not to the first octet of non-urgent data).

1987

Type a message

IETF Group Chat

RFC 793 (page 17)

The urgent pointer points to the sequence number of the octet following the urgent data.

1981

RFC 1011 (page 8)

Page 17 is wrong. The urgent pointer points to the last octet of urgent data (not to the first octet of non-urgent data).

1987

RFC 1122 (page 84)

..the urgent pointer points to the sequence number of the LAST octet (not LAST+1) in a sequence of urgent data.

1989

Type a message

IETF Group Chat

RFC 1011 (page 8)

Page 17 is wrong. The urgent pointer points to the last octet of urgent data (not to the first octet of non-urgent data).

1987

RFC 1122 (page 84)

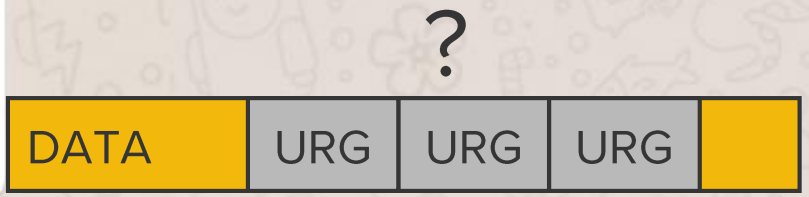
..the urgent pointer points to the sequence number of the LAST octet (not LAST+1) in a sequence of urgent data.

1989

RFC 6093 (pages 6-7)

Considering that as long as both the TCP sender and the TCP receiver implement the same semantics for the Urgent Pointer there is no functional difference in having the Urgent Pointer point to 'the sequence number of the octet following the urgent data' vs. 'the last octet of urgent data', and that all known implementations interpret the semantics of the Urgent Pointer as pointing to 'the sequence number of the octet following the urgent data'.

2011



Type a message

IETF Group Chat

1987

RFC 1122 (page 84)

..the urgent pointer points to the sequence number of the LAST octet (not LAST+1) in a sequence of urgent data.

1989

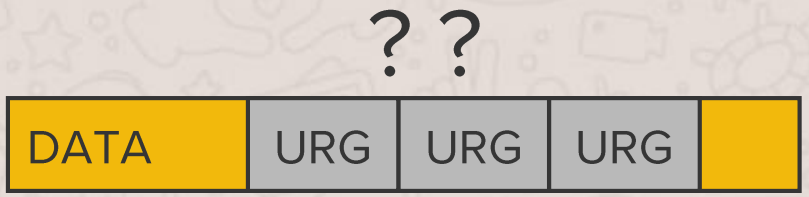
RFC 6093 (pages 6-7)

Considering that as long as both the TCP sender and the TCP receiver implement the same semantics for the Urgent Pointer there is no functional difference in having the Urgent Pointer point to 'the sequence number of the octet following the urgent data' vs. 'the last octet of urgent data', and that all known implementations interpret the semantics of the Urgent Pointer as pointing to 'the sequence number of the octet following the urgent data'.

2011

TCP/IP Stack developer

Wait, what? how do i specify the length of the urgent data?



Type a message

IETF Group Chat

RFC 6093 (pages 6-7)

Considering that as long as both the TCP sender and the TCP receiver implement the same semantics for the Urgent Pointer there is no functional difference in having the Urgent Pointer point to 'the sequence number of the octet following the urgent data' vs. 'the last octet of urgent data', and that all known implementations interpret the semantics of the Urgent Pointer as pointing to 'the sequence number of the octet following the urgent data'.

2011

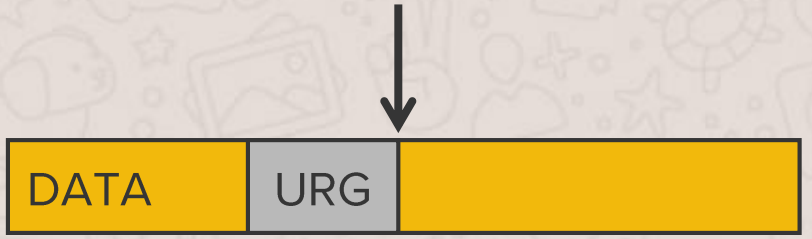
TCP/IP Stack developer

Wait, what? how do i specify the length of the urgent data?

RFC 6093 (page 5)

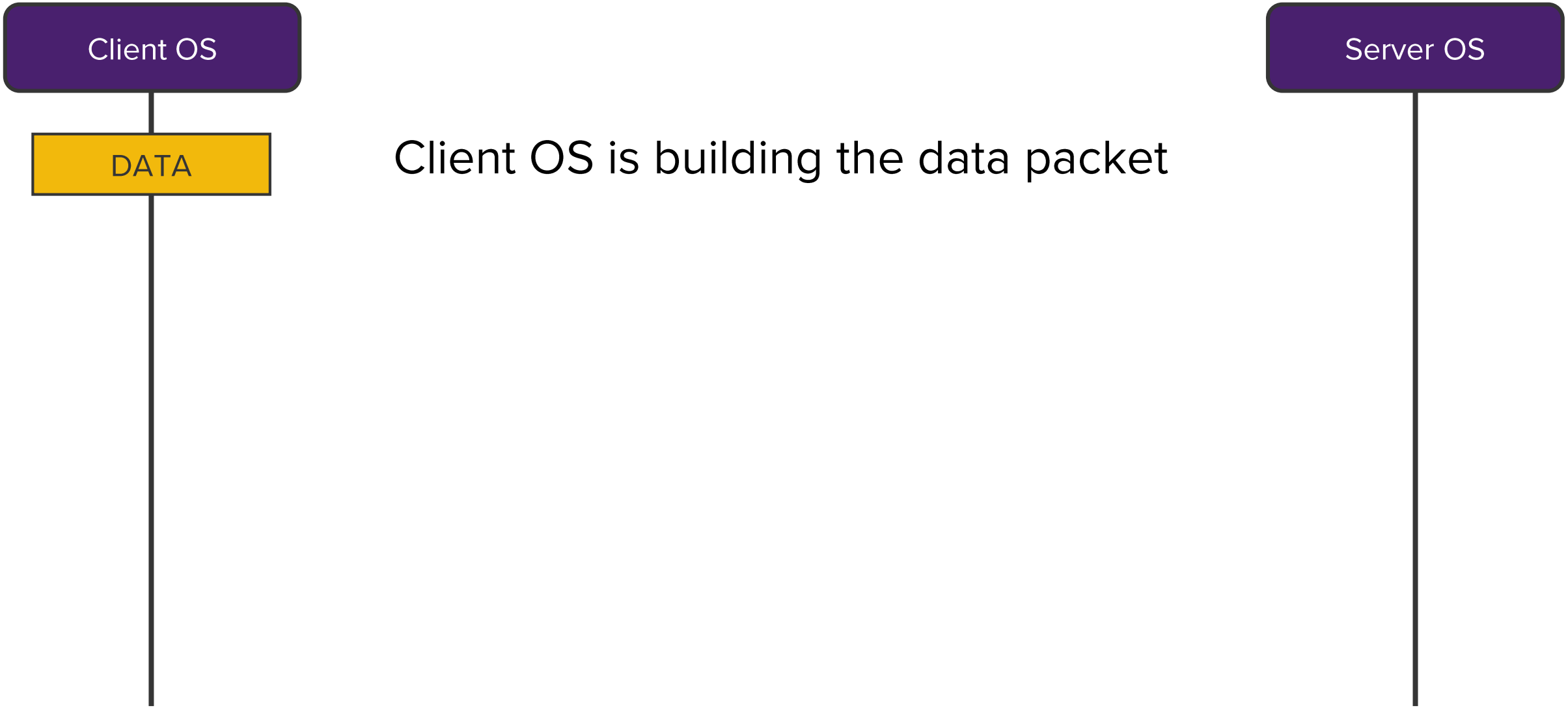
If successive indications of 'urgent data' are received before the application reads the pending 'out-of-band' byte, that pending byte will be discarded (i.e., overwritten by the new byte of 'urgent data').

2011



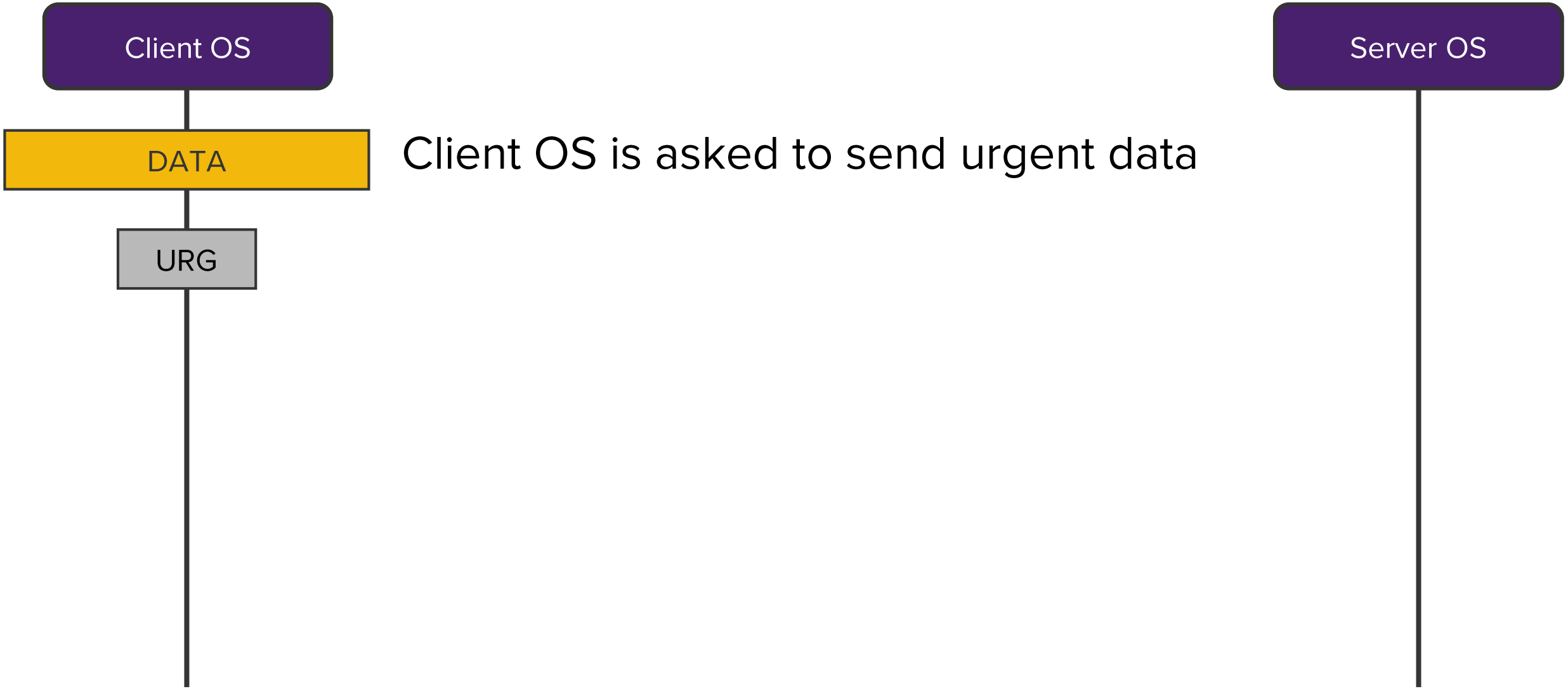
Type a message

TCP Urgent data



Client OS is building the data packet

TCP Urgent data



Client OS is asked to send urgent data

TCP Urgent data

Client OS

Server OS

OS embeds the urgent data inside normal data packet

TCP Urgent data

Server OS

Server's OS recvies data containing urgent data

TCP Urgent data

Server OS

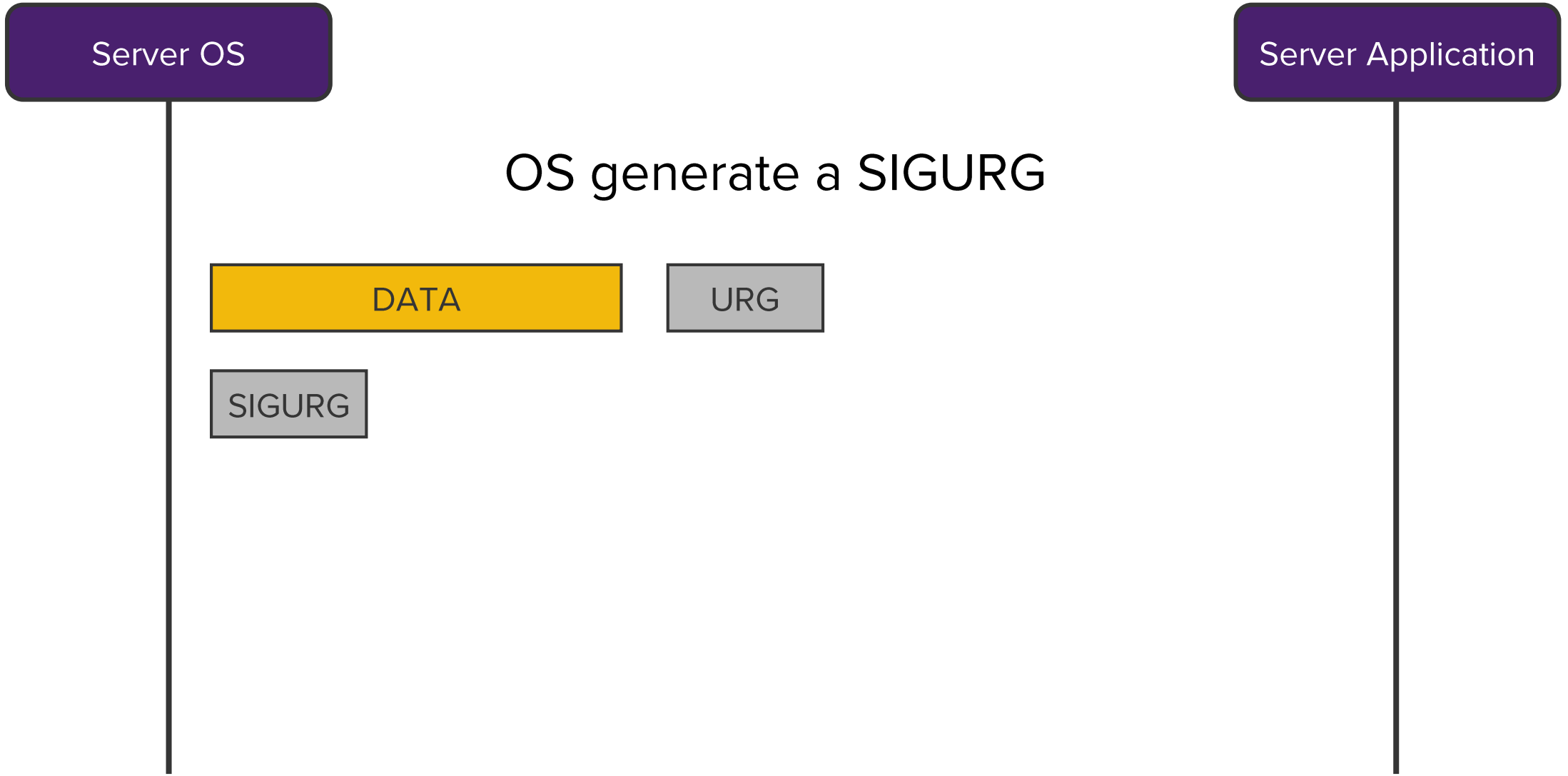
Server Application

OS extracts the urgent data from the band

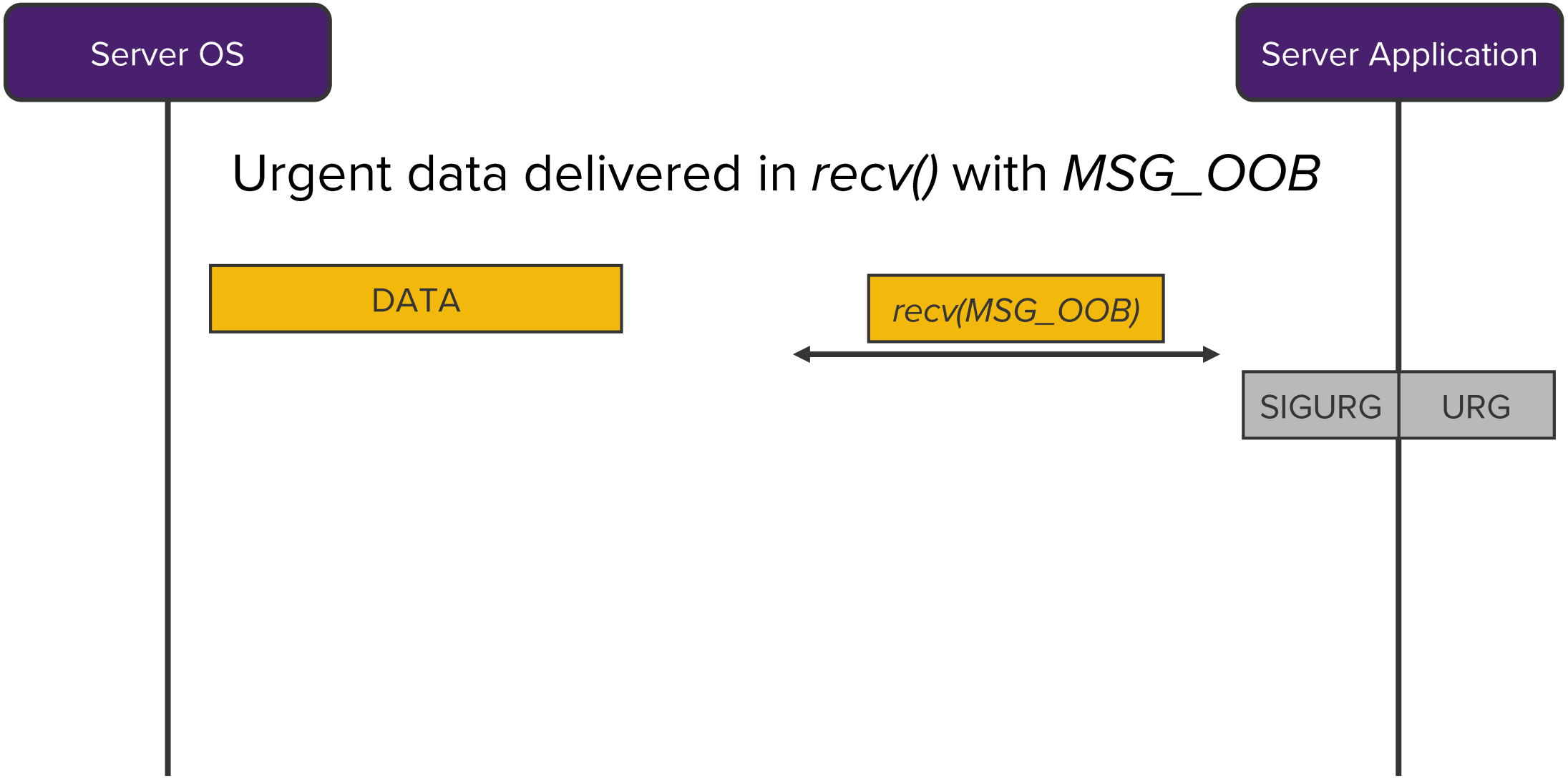
DATA

URG

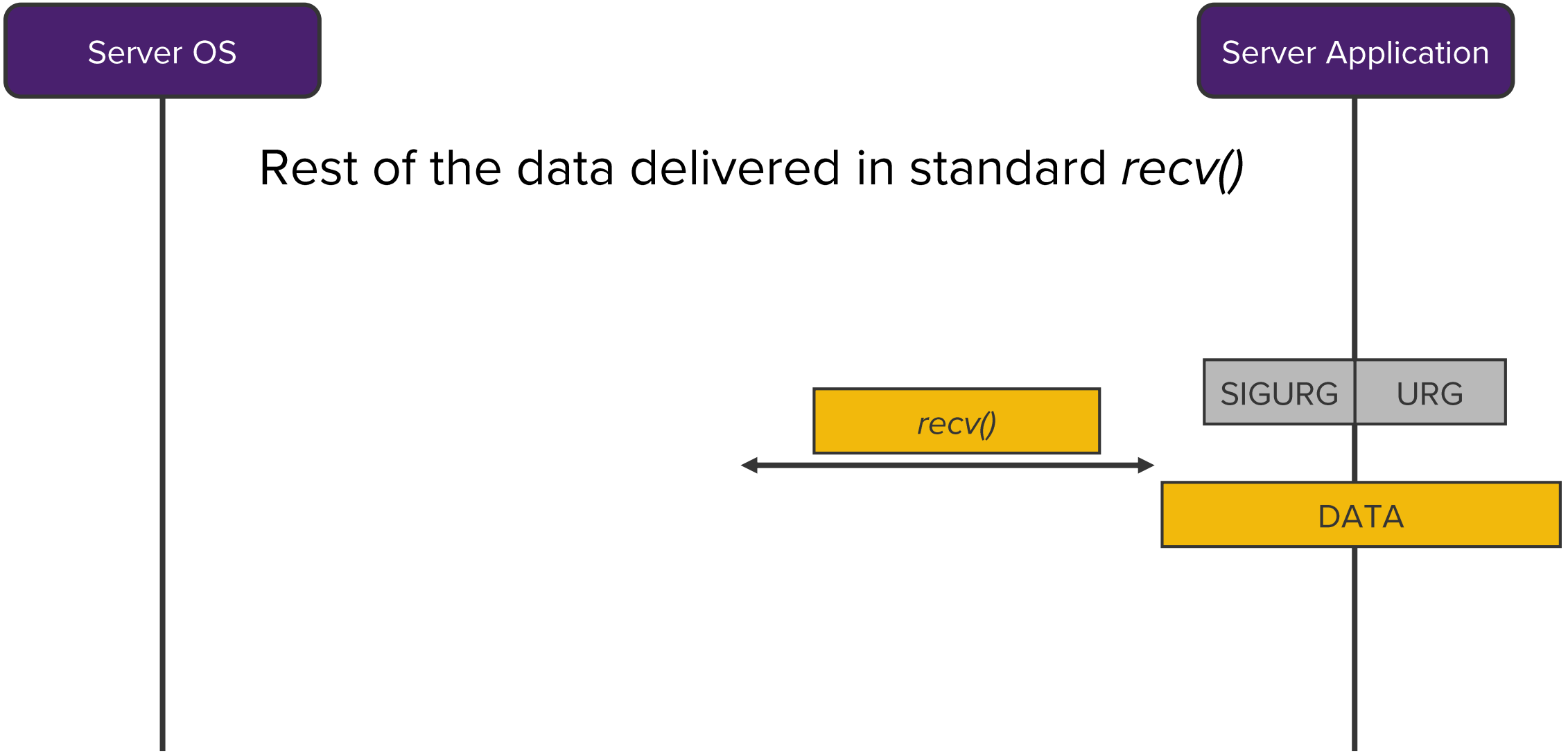
TCP Urgent data



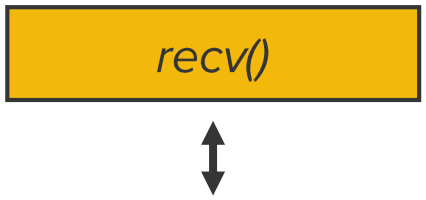
TCP Urgent data



TCP Urgent data

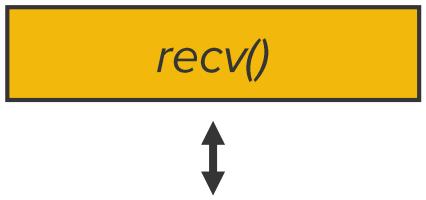


VxWorks implementation




```
int iptcp_usr_get_from_recv_queue(...) {

 // Urgent Data is present, but not requested with the MSG_OOB flag
 if ((int32)(tcb->recv.urg_ptr - len +
             tcb->recv.seq_next - sock->ipcom.rcv_bytes) <= 0) {

    // Calculate the urgent data offset inside the window, in order to
    // copy data up to, but not including the urgent data
    len = tcb->recv.urg_ptr - 1 - tcb->recv.seq_next - sock->ipcom.rcv_bytes;
 }
 ...
}
```

VxWorks implementation



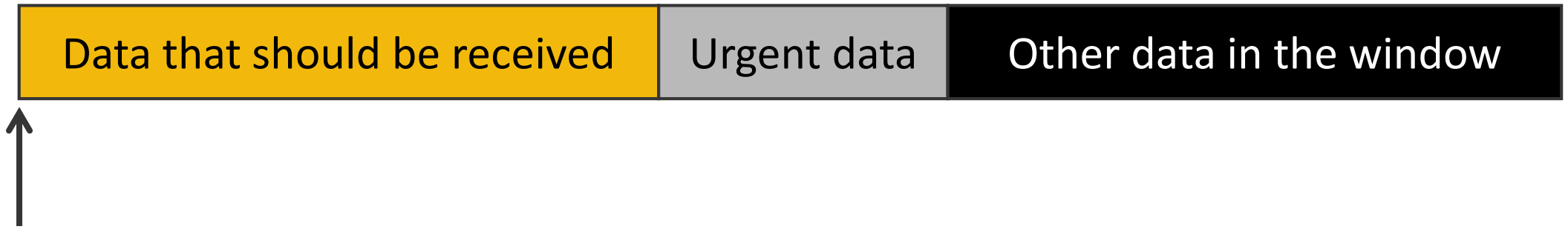

```
int iptcp_usr_get_from_recv_queue(...) {

 // Urgent Data is present, but not requested with the MSG_OOB flag
 if ((int32)(tcb->recv.urg_ptr - len +
             tcb->recv.seq_next - sock->ipcom.rcv_bytes) <= 0) {

    // Calculate the urgent data offset inside the window, in order to
    // copy data up to, but not including the urgent data
len = tcb->recv.urg_ptr - 1 - tcb->recv.seq_next - sock->ipcom.rcv_bytes;
 }
 ...
}
```

VxWorks implementation

```
len = tcb->recv.urg_ptr - 1 - tcb->recv.seq_next - sock->ipcom.rcv_bytes;
```

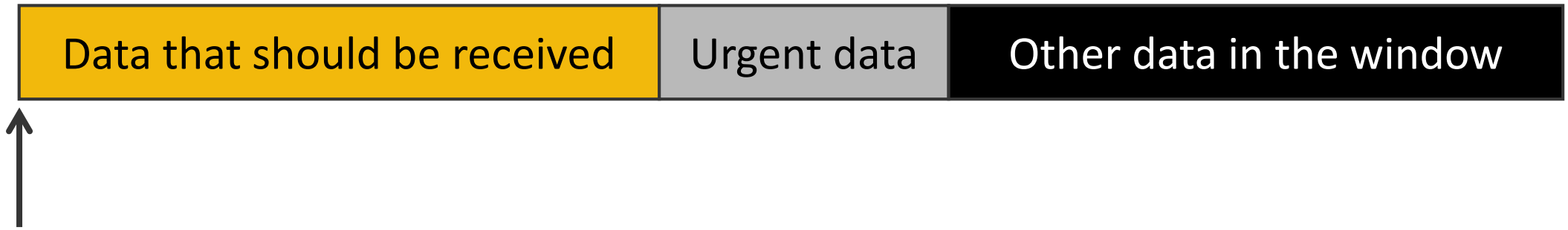
VxWorks implementation

```
len = tcb->recv.urg_ptr - 1 - tcb->recv.seq_next - sock->ipcom.rcv_bytes;
```



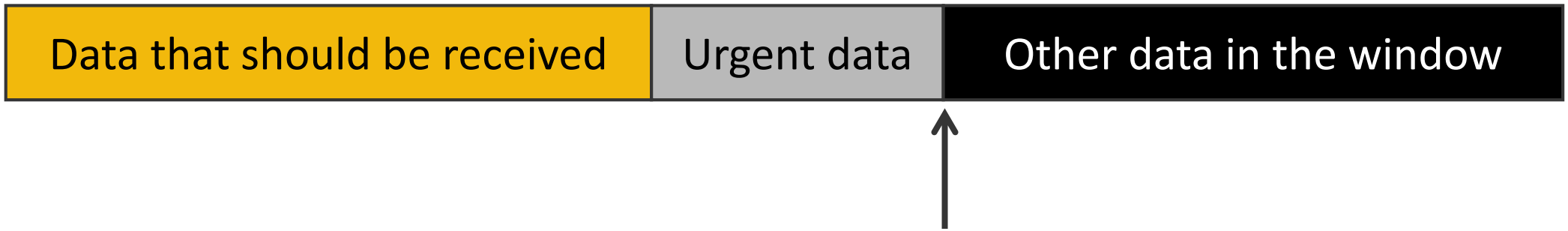
VxWorks implementation

```
len = tcb->recv.urg_ptr - 1 - seq_start
```



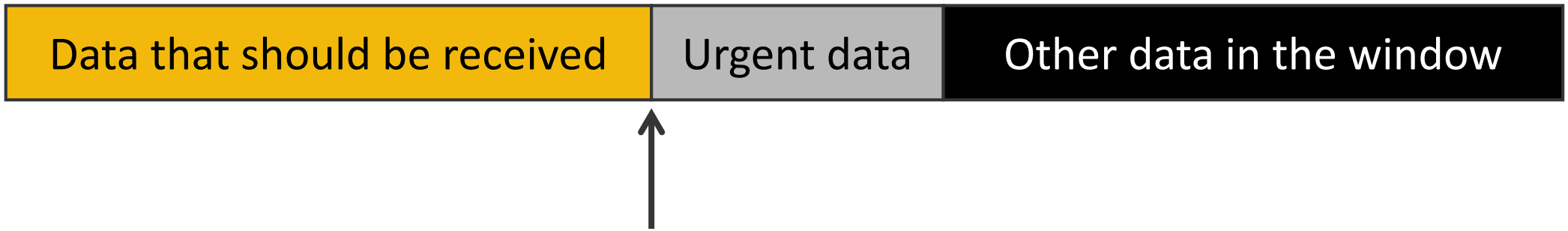
VxWorks implementation

```
len = tcb->recv.urg_ptr - 1 - seq_start
```



VxWorks implementation

```
len = tcb->recv.urg_ptr - 1 - seq_start
```



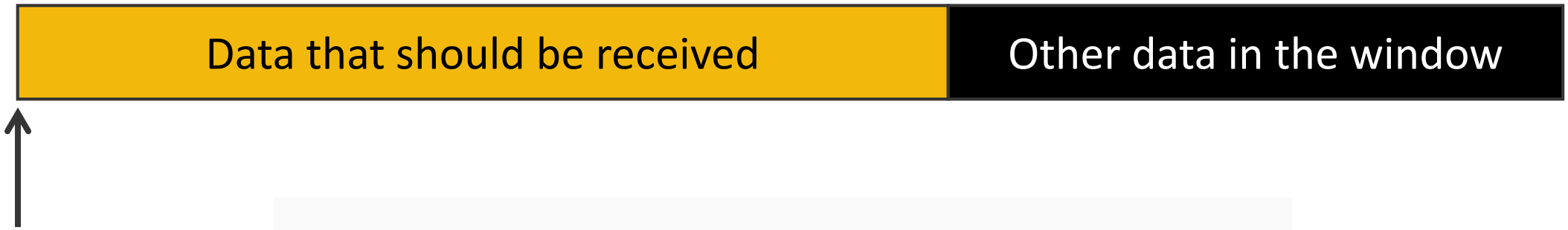
Urgent Pointer = 0

```
len = tcb->recv.urg_ptr - 1 - seq_start
```


```
Urgent Pointer = 0
tcb->recv.urg_ptr = seq_start + 0
```

Urgent Pointer = 0

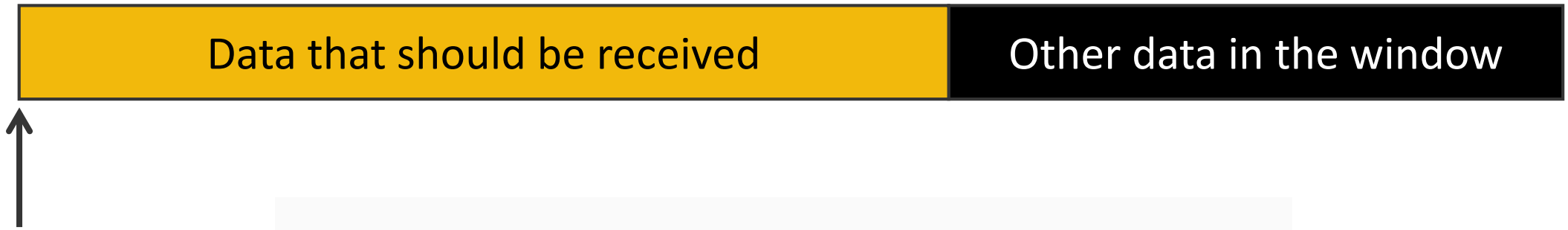
```
len = tcb->recv.urg_ptr - 1 - seq_start
```




```
Urgent Pointer = 0
tcb->recv.urg_ptr = seq_start + 0
```

Urgent Pointer = 0

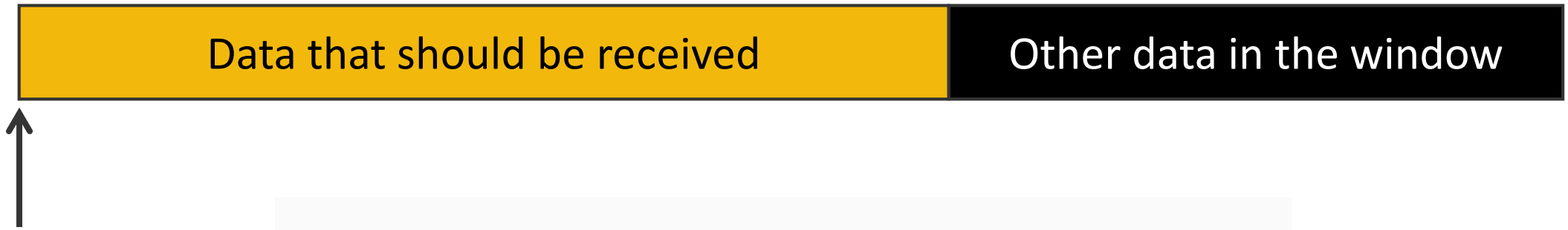
```
len = tcb->recv.urg_ptr - 1 - seq_start
```




```
Urgent Pointer = 0
tcb->recv.urg_ptr = seq_start + 0
```

Urgent Pointer = 0

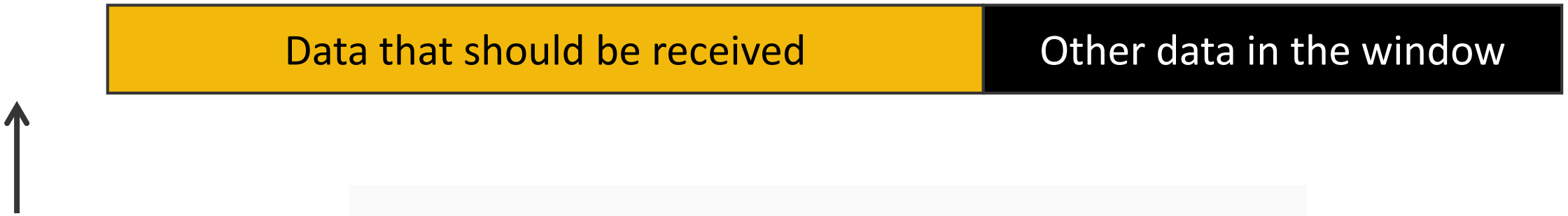
```
len = seq_start - 1 - seq_start
```




```
Urgent Pointer = 0
tcb->recv.urg_ptr = seq_start + 0
```

Urgent Pointer = 0

```
len = -1
```

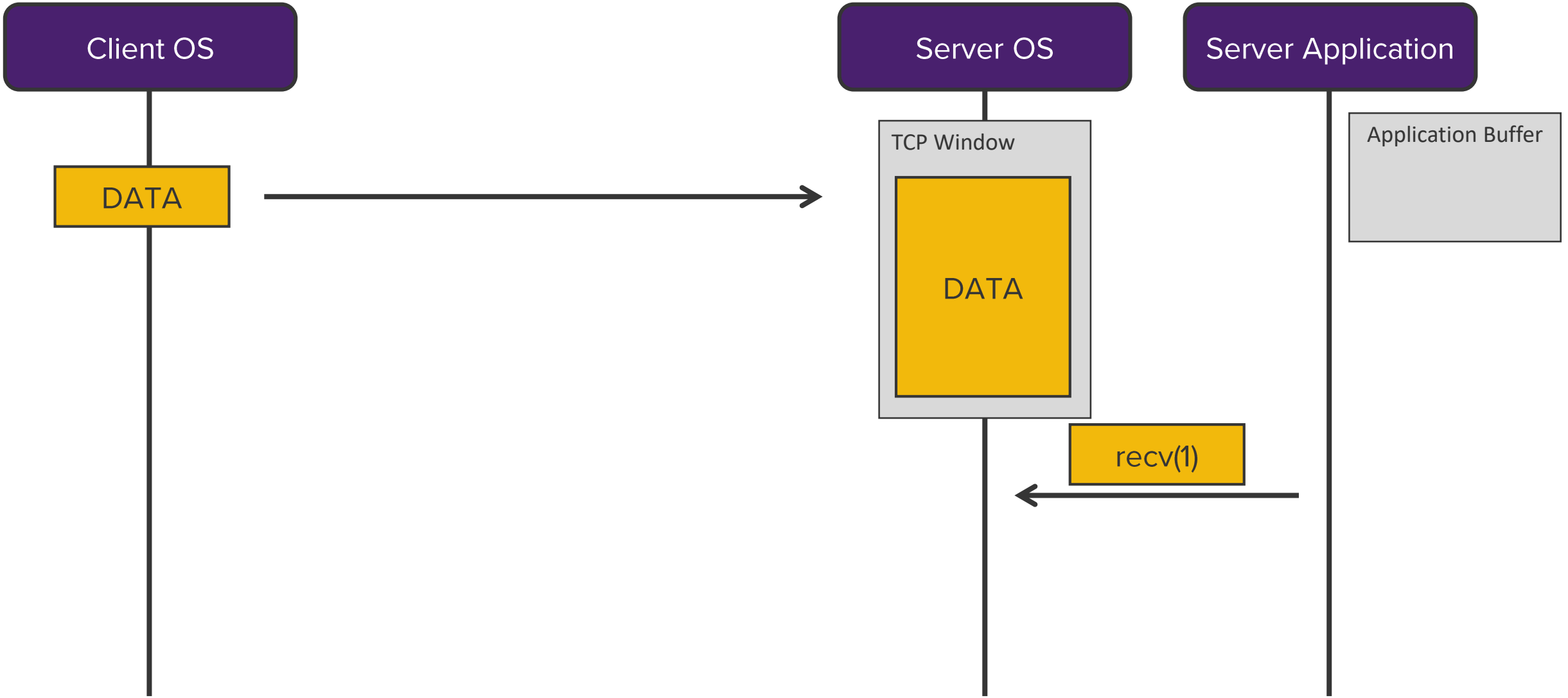



```
Urgent Pointer = 0
tcb->recv.urg_ptr = seq_start + 0
```

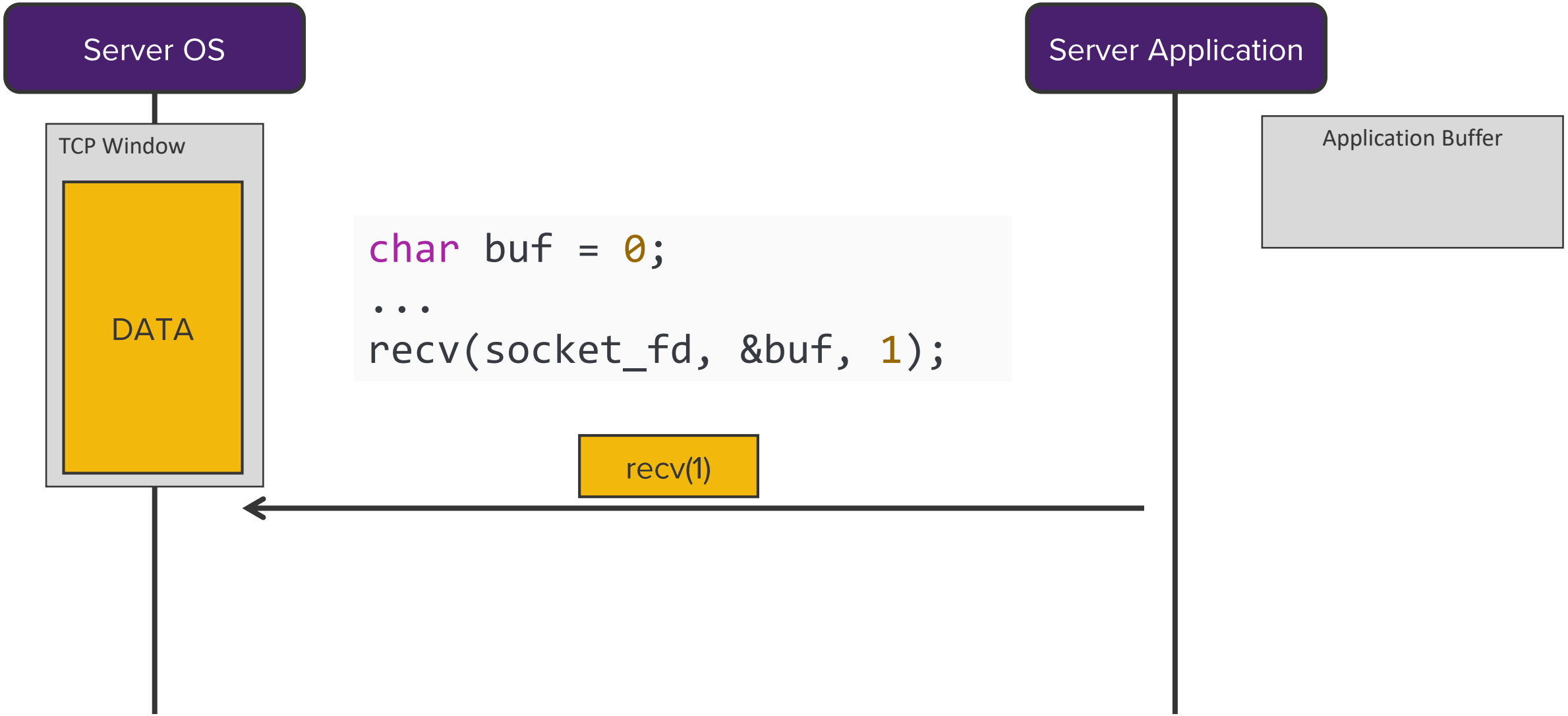
Urgent Pointer = 0

```
int iptcp_usr_get_from_recv_queue(...) {

 // Urgent Data is present, but not requested with the MSG_OOB flag
 if ((int32)(tcb->recv.urg_ptr - len +
             tcb->recv.seq_next - sock->ipcom.rcv_bytes) <= 0) {

   // Calculate the urgent data offset inside the window,
   // in order to copy data up to, but not including the urgent data
   len = (unsigned int32) -1;
 }
...
}
```

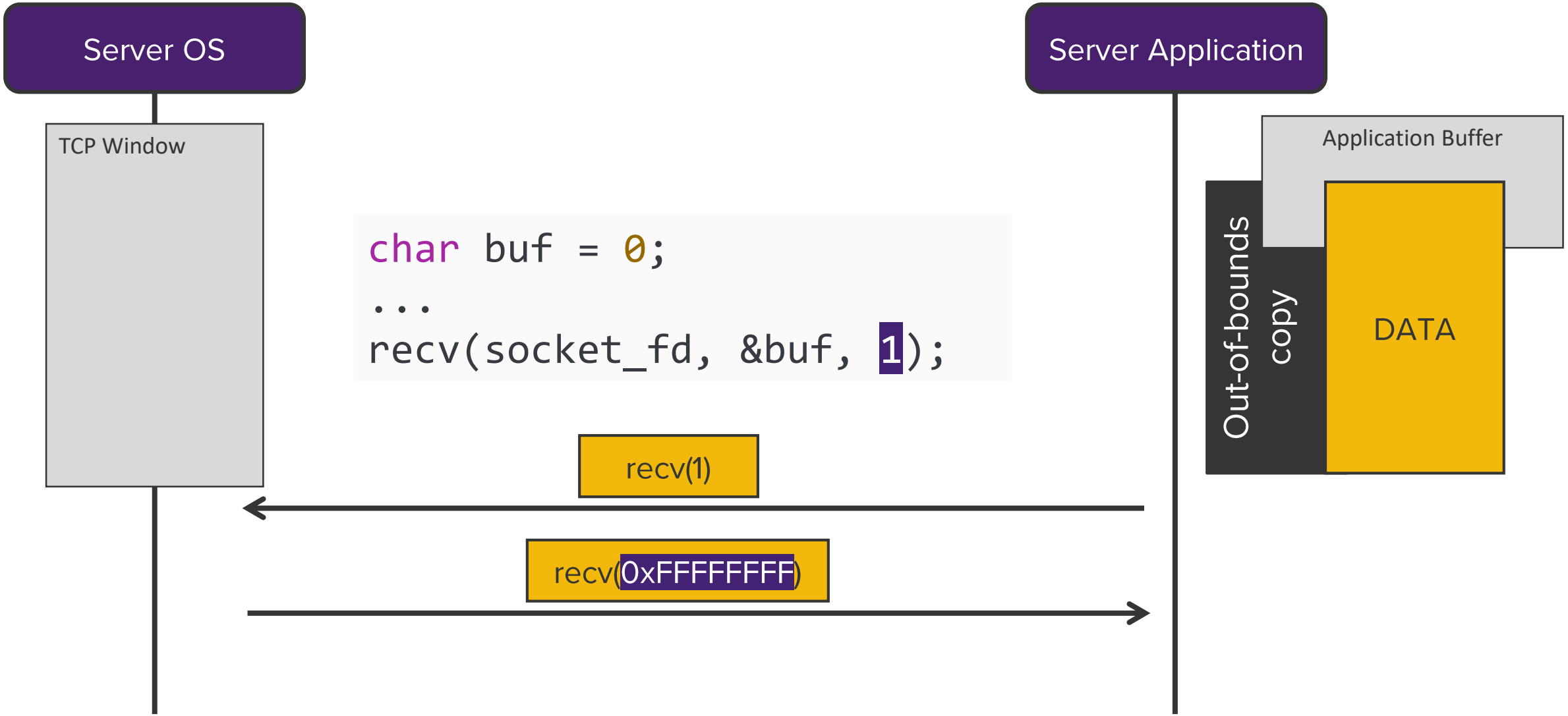
Urgent Pointer = 0



Urgent Pointer = 0



Urgent Pointer = 0

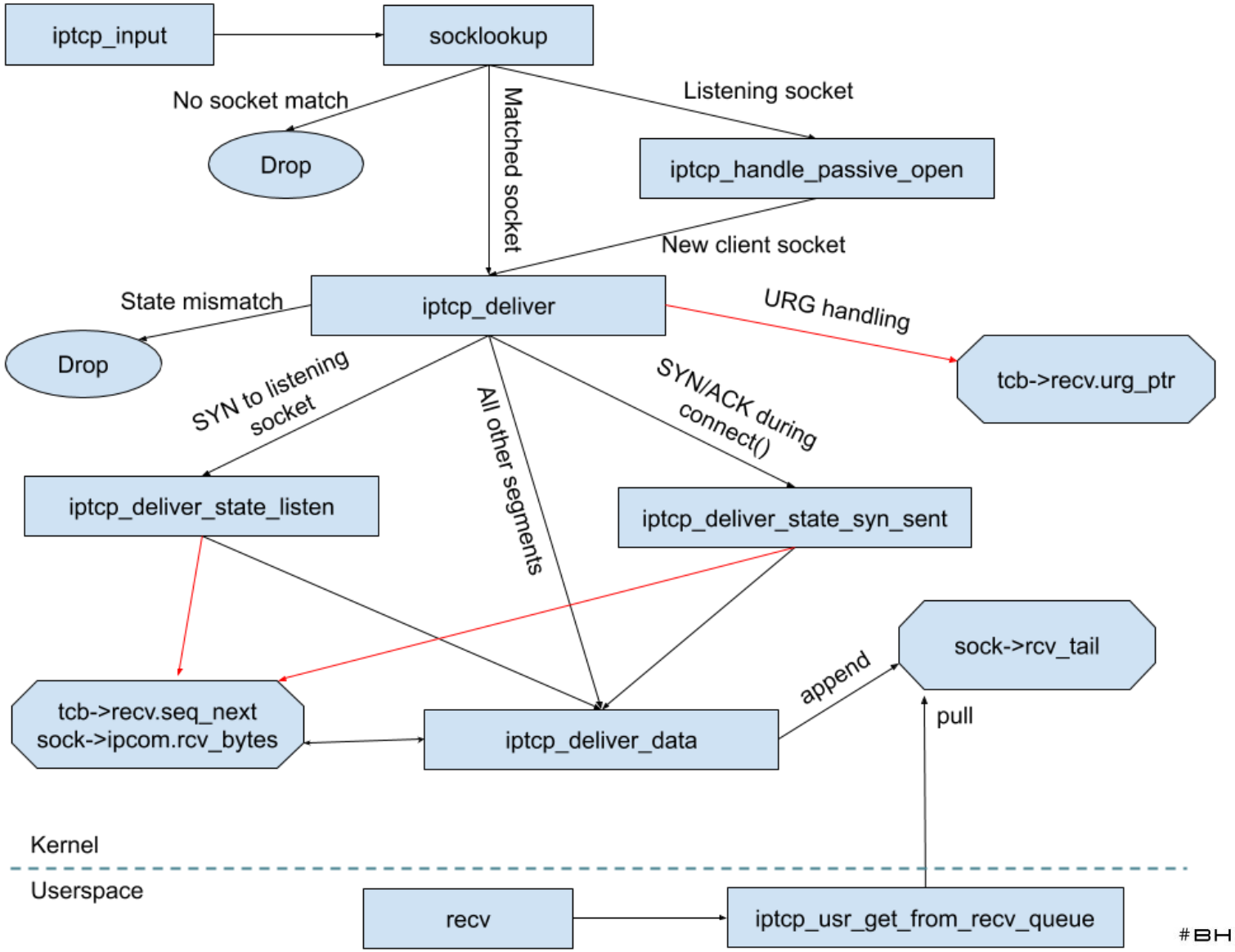


2013: Code changed; bug averted

A closer look

```
Int iptcp_usr_get_from_recv_queue(...)
{
// Urgent Data is present, but not requested with the MSG_OOB flag
if ((int32)(tcb->recv.urg_ptr - len +
            tcb->recv.seq_next - sock->ipcom.rcv_bytes) <= 0)
{
    // Calculate the urgent data offset inside the window, in order to
    // copy data up to, but not including the urgent data
len = tcb->recv.urg_ptr - 1 - tcb->recv.seq_next - sock->ipcom.rcv_bytes;
}
...
}
```

State confusions can be confusing



Five-way handshake

Source	Destination	Length	Proto	Info
192.168.108.1	192.168.108.10	58	TCP	29019 → 59747 [SYN] Seq=0 Win=8192 Len=0

```
▾ Options: (4 bytes), Unknown (0x1d), End of Option List (EOL)
  ▾ TCP Option - Unknown
      Kind: TCP Authentication Option (29)
      Length: 3
      Payload: 61
  ▸ TCP Option - End of Option List (EOL)
```

Socket Object

State	**IPTCP_STATE_LISTEN**
seq_next	**NULL**
urg_ptr	**NULL**
4 - tuple	**SPORT,DPORT, SRC, DST**

Five-way handshake

Source	Destination	Length	Proto	Info
192.168.108.1	192.168.108.10	58	TCP	29019 → 59747 [SYN] Seq=0 Win=8192 Len=0
192.168.108.1	192.168.108.10	54	TCP	29019 → 59747 [FIN, SYN, URG] Seq=0 Win=8192 Urg=10 Len=0

Socket Object

State	**IPTCP_STATE_LISTEN**
seq_next	**0**
urg_ptr	**10**
4 - tuple	**SPORT,DPORT, SRC, DST**

Five-way handshake

Source	Destination	Length	Proto	Info
192.168.108.1	192.168.108.10	58	TCP	29019 → 59747 [SYN] Seq=0 Win=8192 Len=0
192.168.108.1	192.168.108.10	54	TCP	29019 → 59747 [FIN, SYN, URG] Seq=0 Win=8192 Urg=10 Len=0
192.168.108.1	192.168.108.10	54	TCP	29019 → 59747 [SYN] Seq=1000000 Win=8192 Len=0

Socket Object

State	**IPTCP_STATE_SYN_RCV**
seq_next	**1000000**
urg_ptr	**10**
4 - tuple	**SPORT,DPORT, SRC, DST**

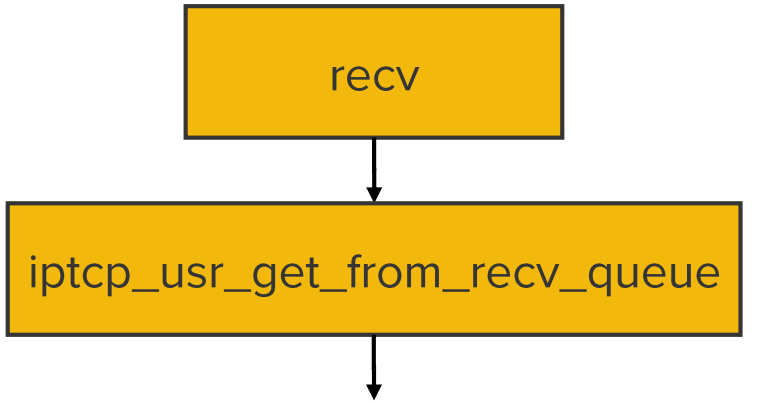
Five-way handshake

Source	Destination	Length	Proto	Info
192.168.108.1	192.168.108.10	58	TCP	29019 → 59747 [SYN] Seq=0 Win=8192 Len=0
192.168.108.1	192.168.108.10	54	TCP	29019 → 59747 [FIN, SYN, URG] Seq=0 Win=8192 Urg=10 Len=0
192.168.108.1	192.168.108.10	54	TCP	29019 → 59747 [SYN] Seq=1000000 Win=8192 Len=0
192.168.108.10	192.168.108.1	58	TCP	59747 → 29019 [SYN, ACK] Seq=1885851430 Ack=1000001 Win=60000 Len=0 MSS=1460
192.168.108.1	192.168.108.10	1078	TCP	29019 → 59747 [ACK] Seq=1000001 Ack=1885851431 Win=8192 Len=1024

Socket Object

State	**IPTCP_STATE_SYN_RCV**
seq_next	**1000000 + rcv_bytes**
urg_ptr	**10**
4 - tuple	**SPORT,DPORT, SRC, DST**

Triggering an overflow

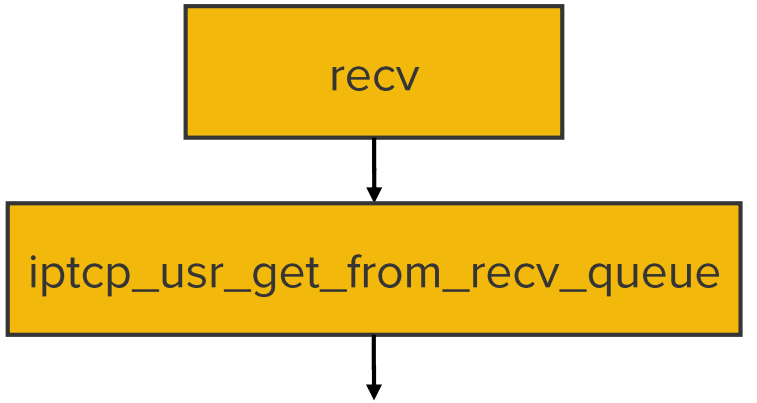



```
len = tcb->recv.urg_ptr - 1 –
      tcb->recv.seq_next - sock->ipcom.rcv_bytes;
```

Socket Object

State	IPTCP_STATE_SYN_RCV
seq_next	1000000 + rcv_bytes
urg_ptr	10
4 - tuple	SPORT,DPORT, SRC, DST

Triggering an overflow

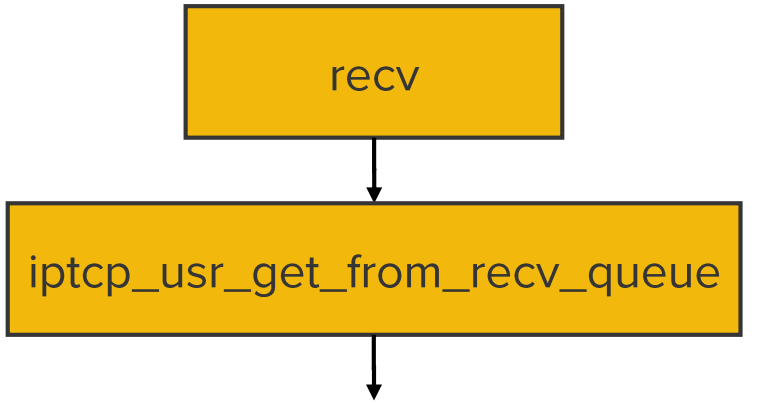



```
len = 10 - 1 -
      (1000000 + sock->ipcom.rcv_bytes) - sock->ipcom.rcv_bytes;
```

Socket Object

State	**IPTCP_STATE_SYN_RCV**
seq_next	**1000000 + rcv_bytes**
urg_ptr	**10**
4 - tuple	**SPORT,DPORT, SRC, DST**

Triggering an overflow




```
len = 9 -
      1000000;
```

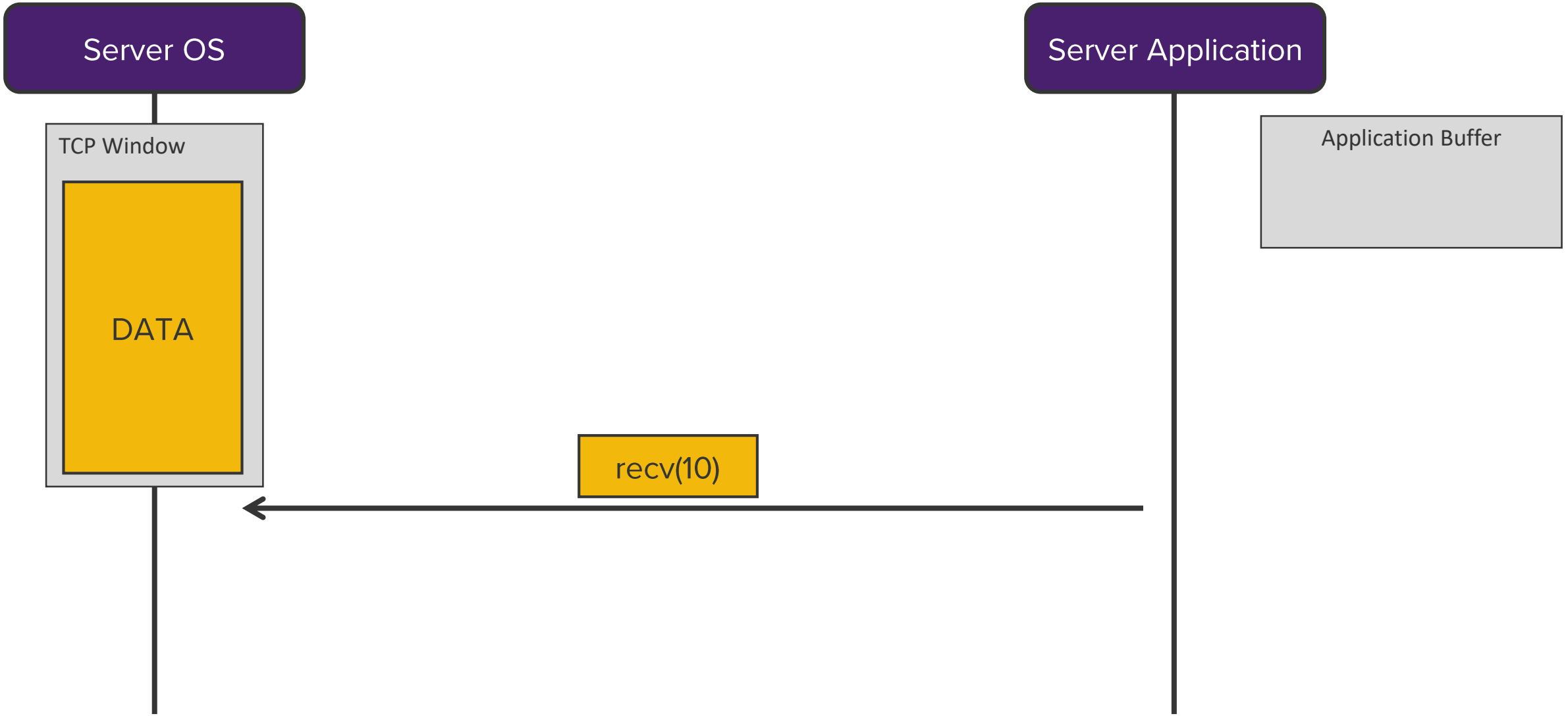
Socket Object

State	**IPTCP_STATE_SYN_RCV**
seq_next	**1000000 + rcv_bytes**
urg_ptr	**10**
4 - tuple	**SPORT,DPORT, SRC, DST**

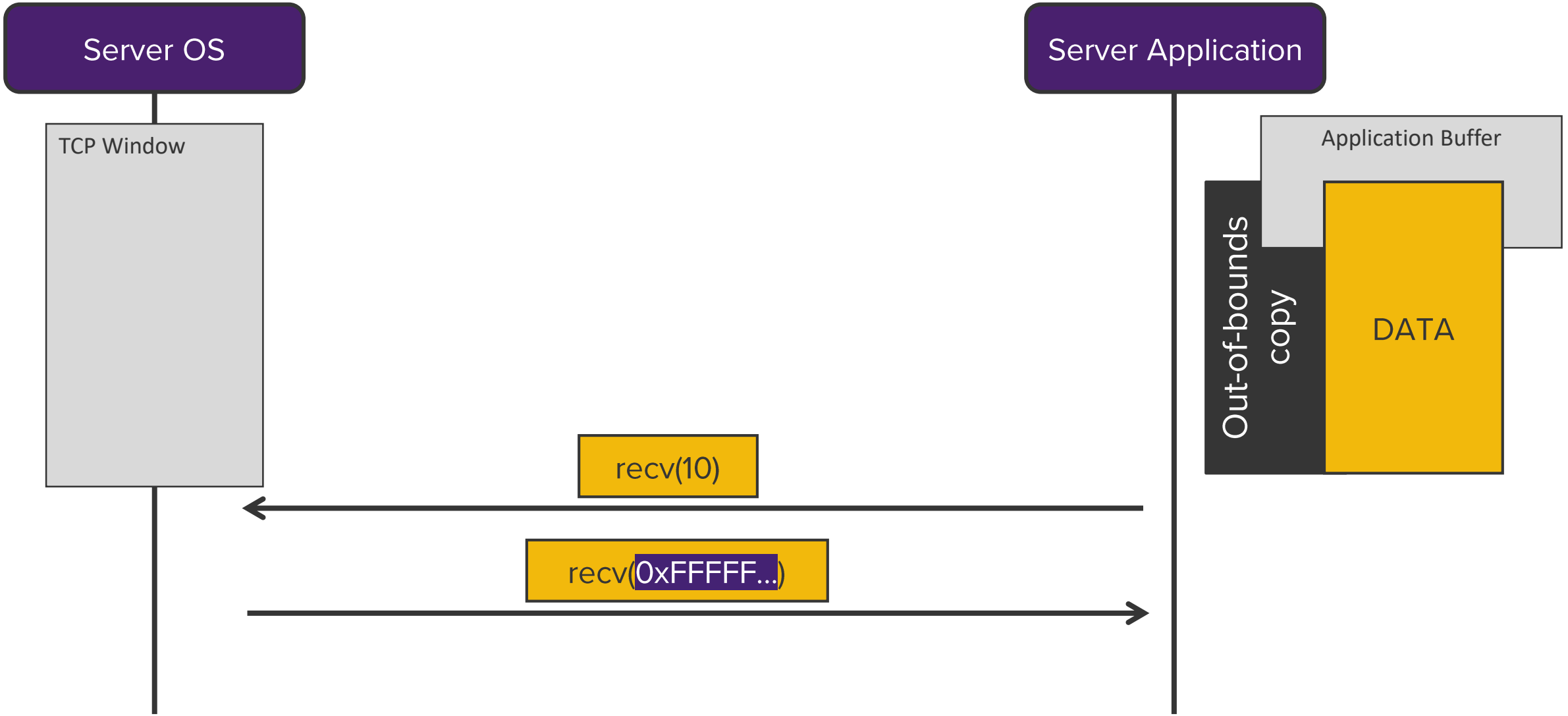
Triggering an overflow

```
int iptcp_usr_get_from_recv_queue(...) {

 // Urgent Data is present, but not requested with the MSG_OOB flag
 if ((int32)(tcb->recv.urg_ptr - len +
             tcb->recv.seq_next - sock->ipcom.rcv_bytes) <= 0) {

   // Calculate the urgent data offset inside the window,
   // in order to copy data up to, but not including the urgent data
   len = (unsigned int32) -999991;
 }
...
}
```

Triggering an overflow



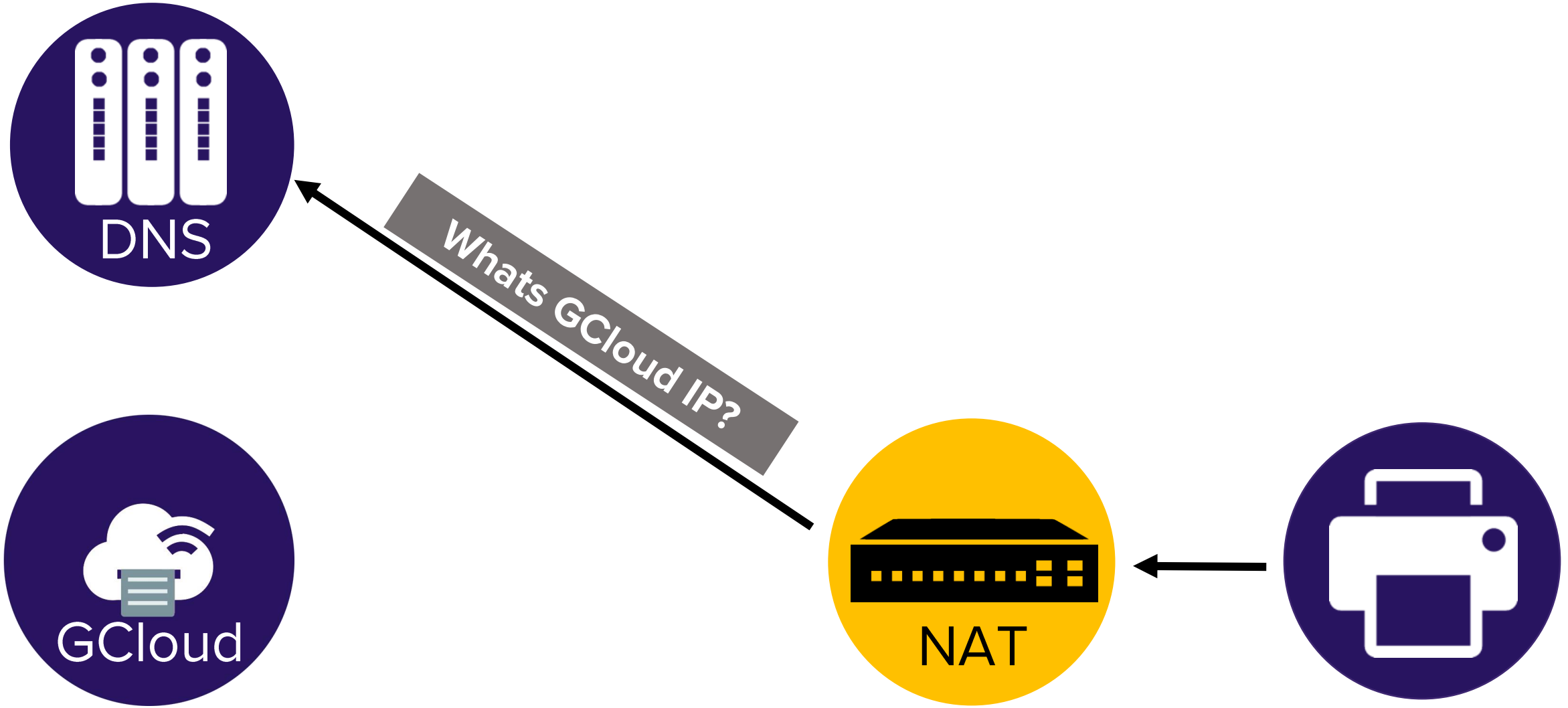
Triggering an overflow



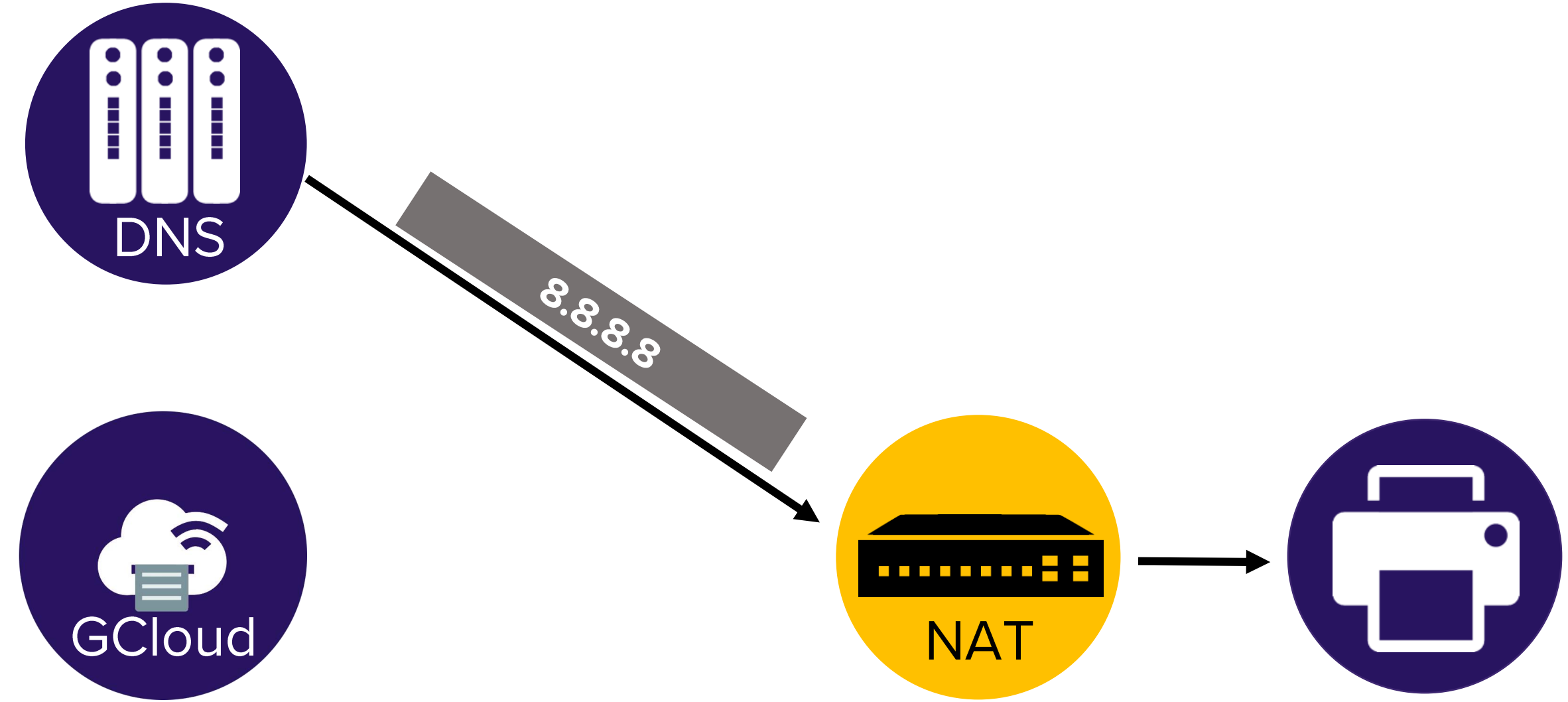
TCP Urgent Pointer RCEs

Urg0	5-Way Handshake
6.5.0 – 6.9.4	**Last Week**
2006 – 2013	*2019*

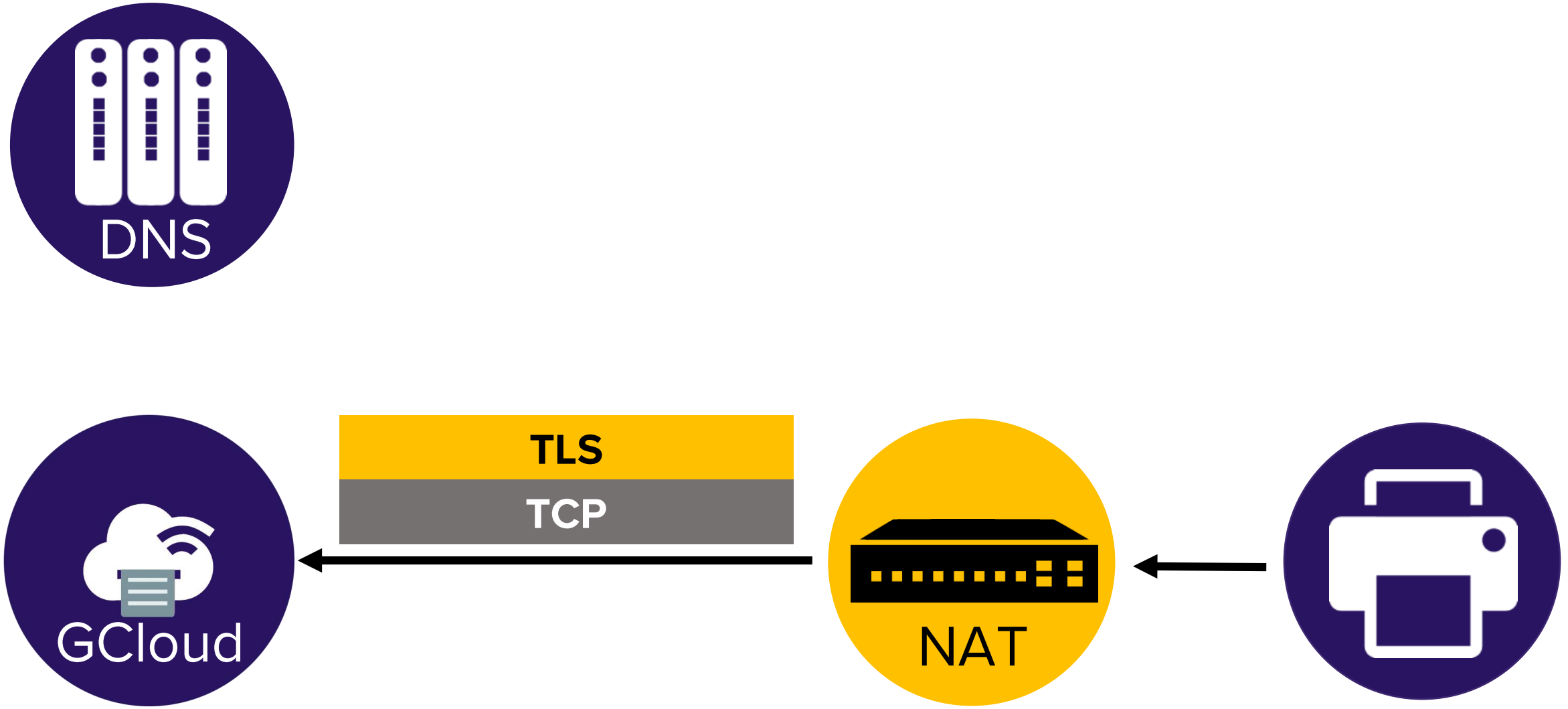
Bypassing NAT with TCP Urgent RCEs



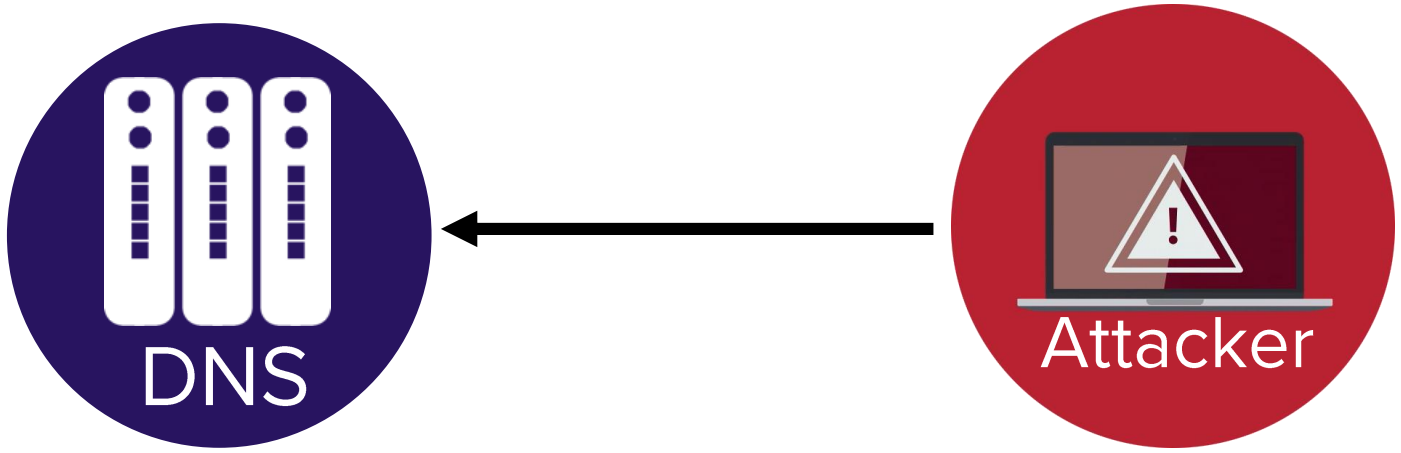
Bypassing NAT with TCP Urgent RCEs



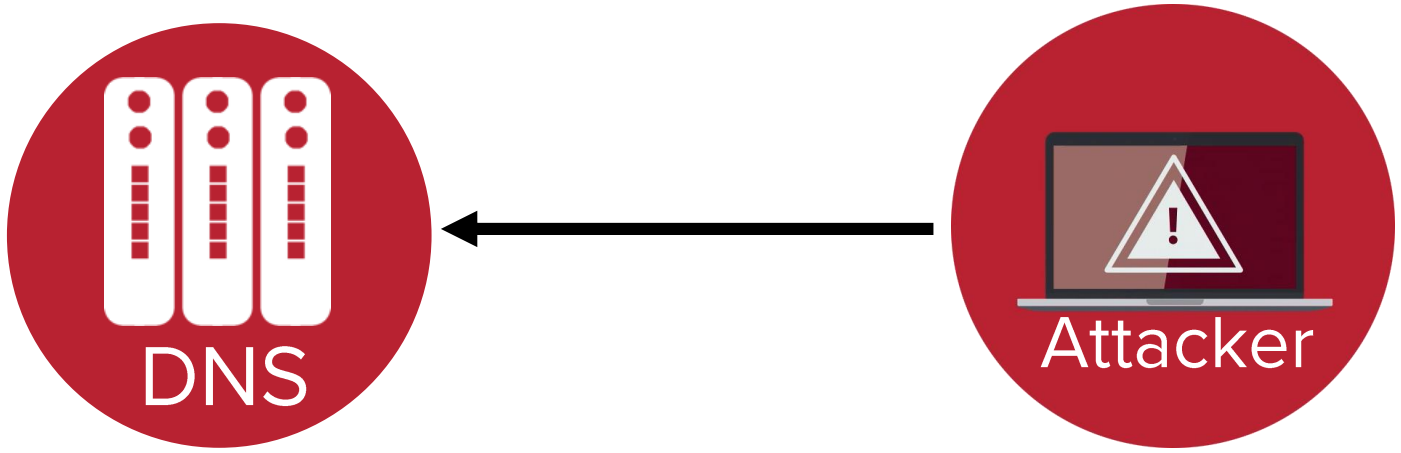
Bypassing NAT with TCP Urgent RCEs



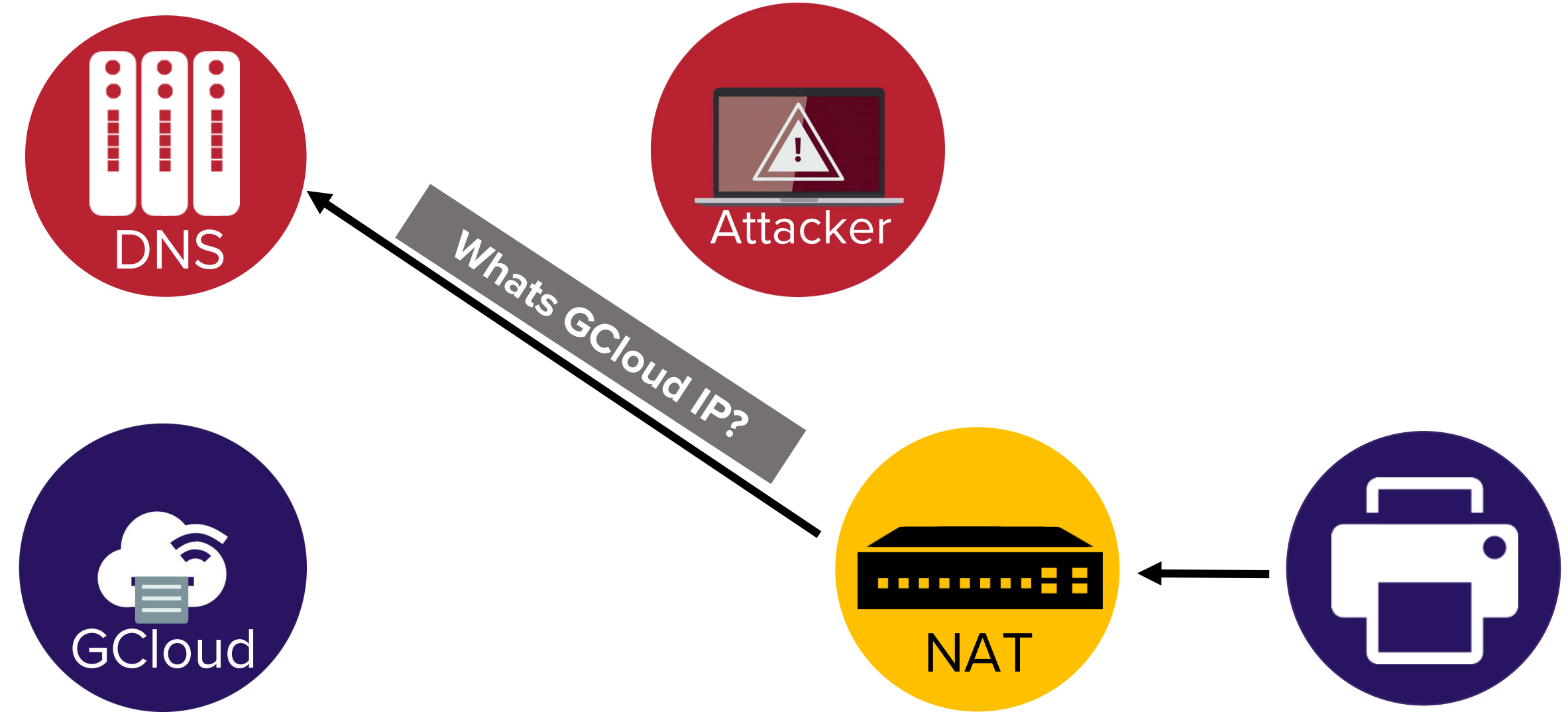
Bypassing NAT with TCP Urgent RCEs



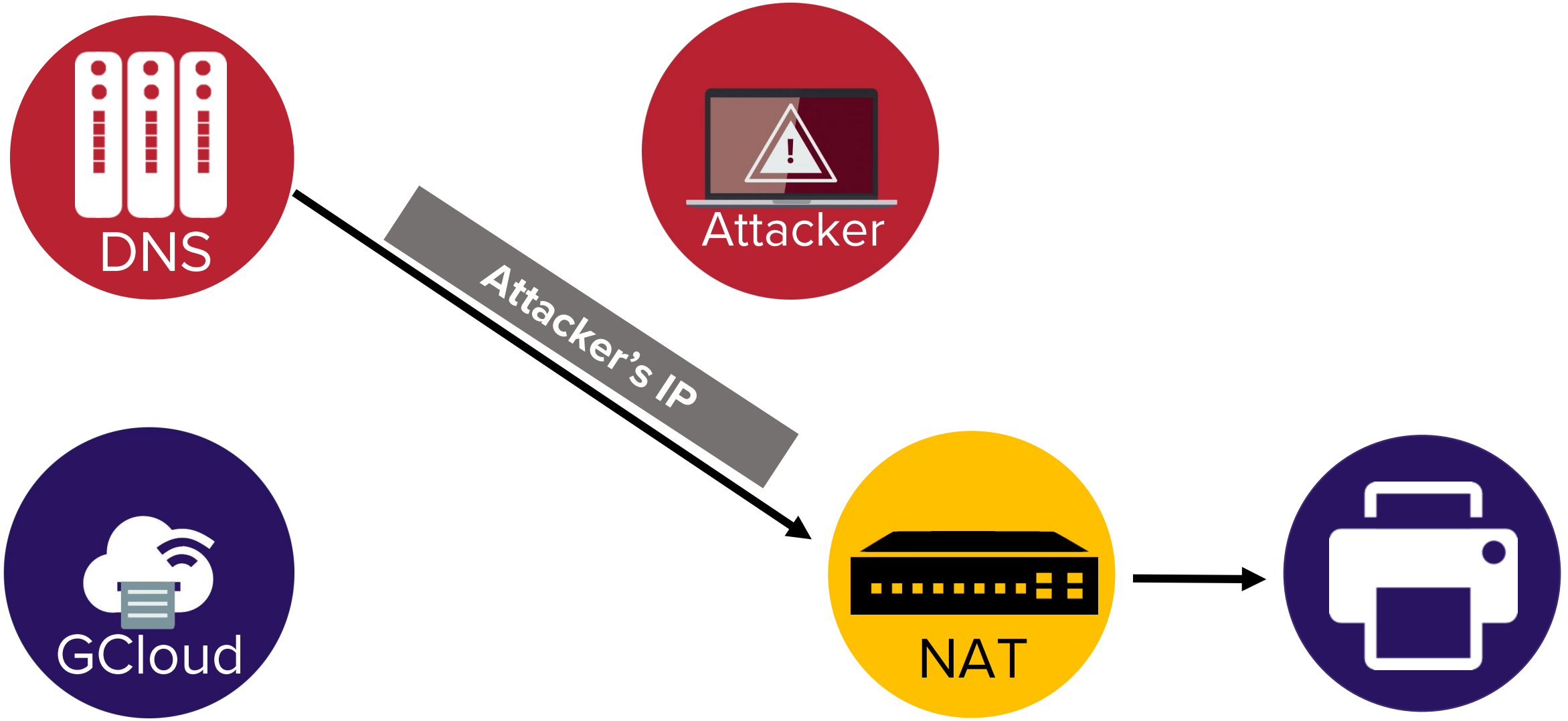
Bypassing NAT with TCP Urgent RCEs



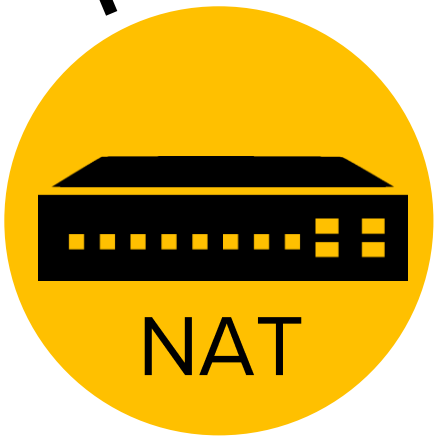
Bypassing NAT with TCP Urgent RCEs



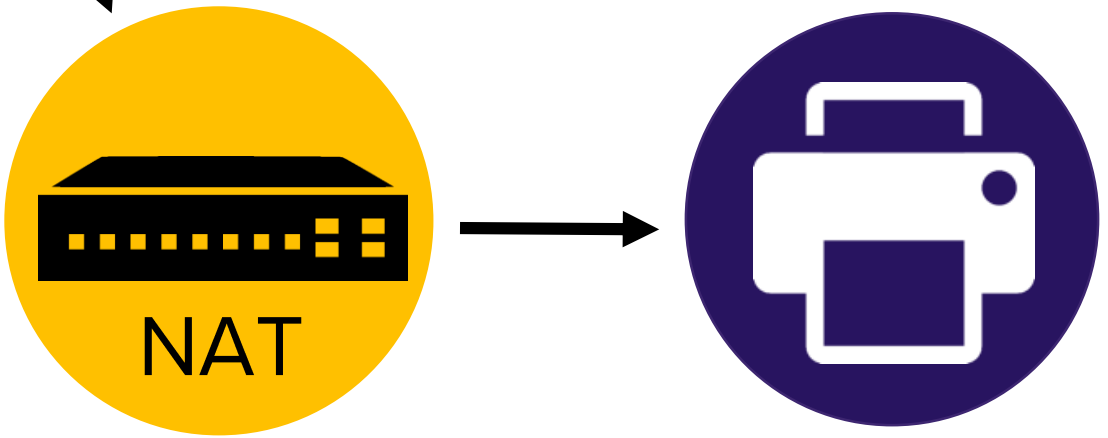
Bypassing NAT with TCP Urgent RCEs



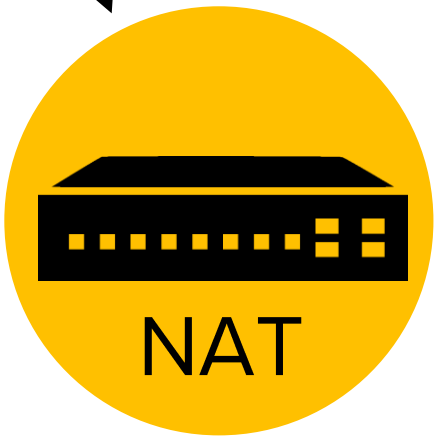
Bypassing NAT with TCP Urgent RCEs



Bypassing NAT with TCP Urgent RCEs

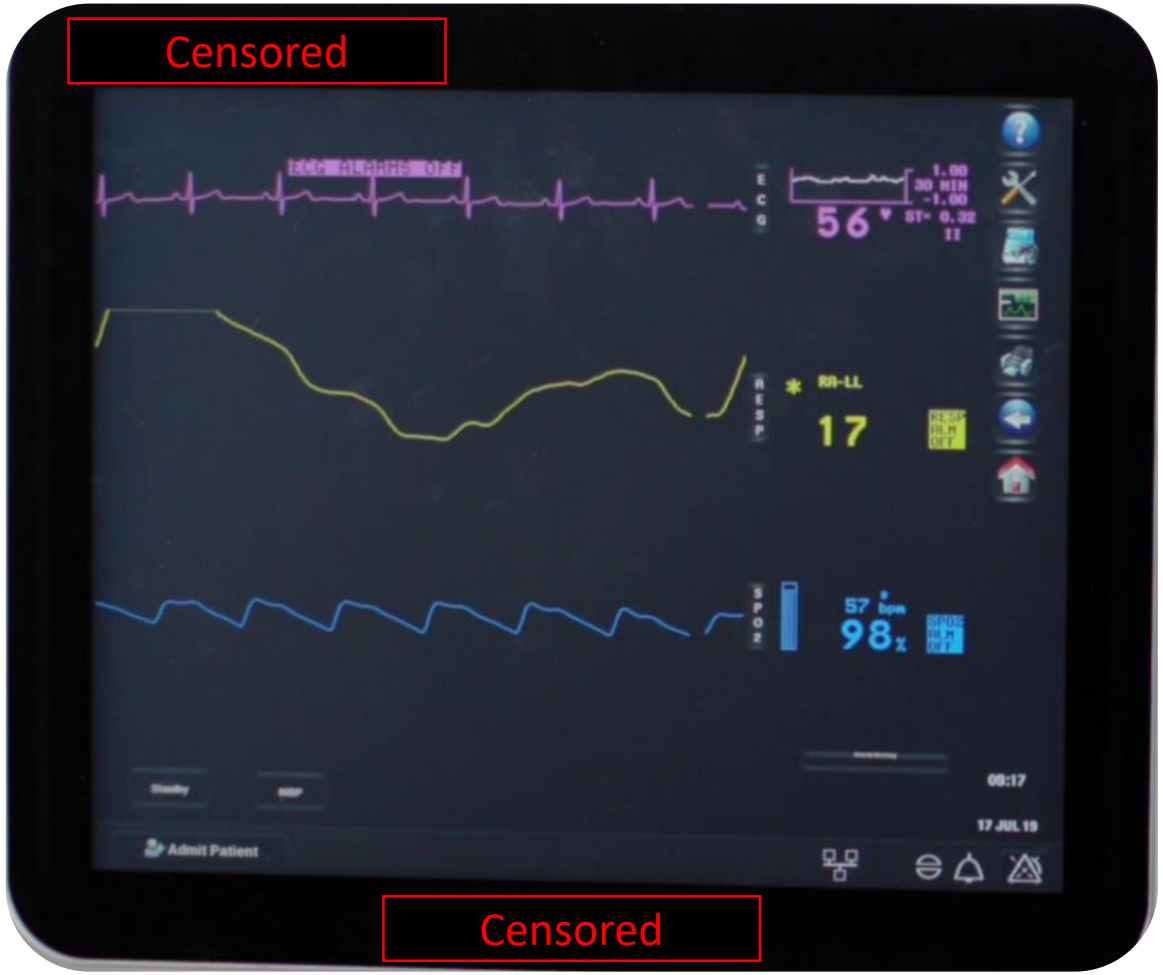


Bypassing NAT with TCP Urgent RCEs



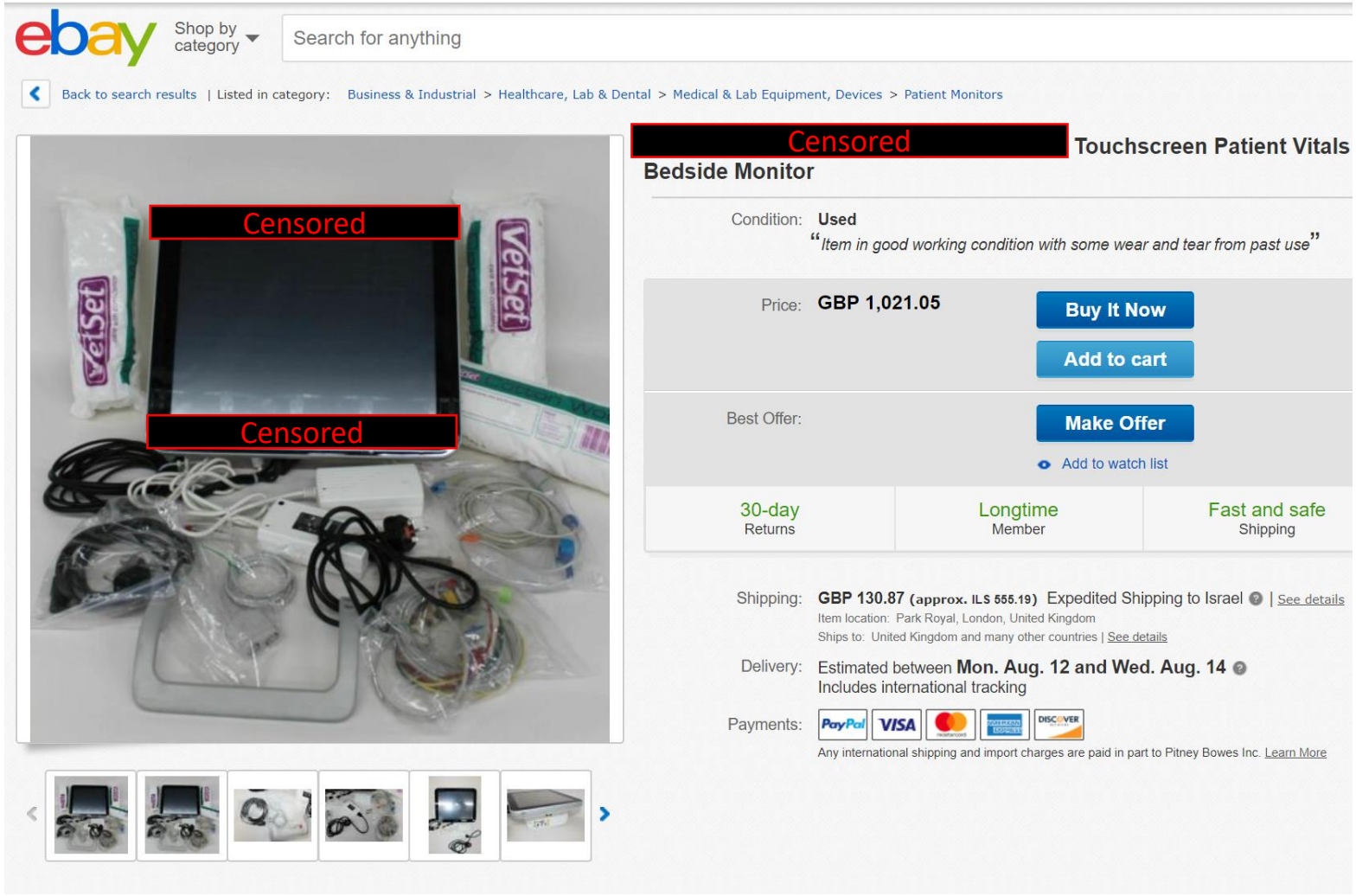
Censored

Patient monitor

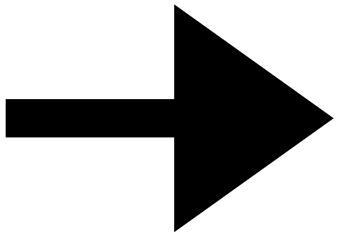


- Run VxWorks 6.6
- Vulnerable to the Urg0 RCE
- Listening TCP Port that receives to heap
- You can buy it on eBay!

Getting the firmware in 3 easy steps



Getting the firmware in 3 easy steps



In Transit

03/05/2019 - 12:41 P.M.
Lod, Israel
Power of attorney documentation is missing and is required for clearance. We're working to obtain this information.

Less

03/05/2019 - 12:41 P.M.

Your package is being processed for submission to the FDA or Department of Agriculture. / We've contacted the receiver.

Less

03/05/2019 - 12:41 P.M.

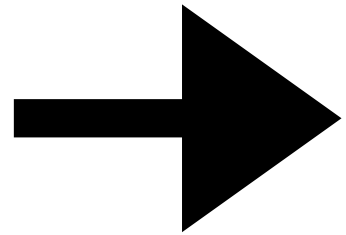
Lod, Israel

An import license is missing and is requir[illegible] clearance.

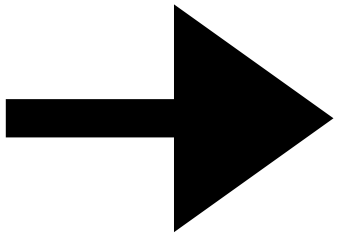
Ask UPS

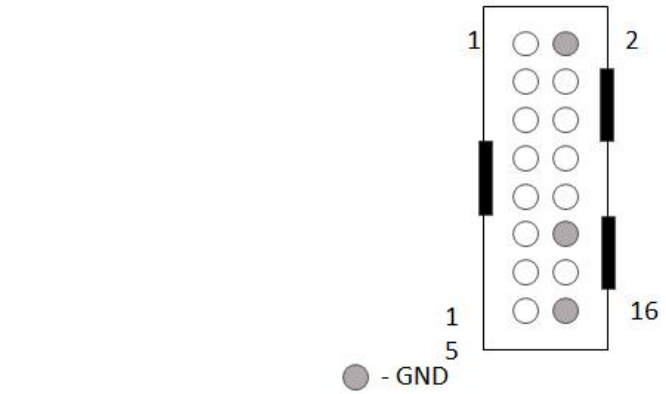
03/05/2019 - 11:26 A.M.

As requested by the receiver, we'll hold the package, and contact them for instructio ... More



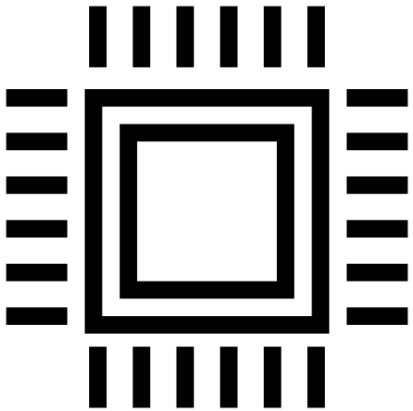
Getting the firmware in 3 easy steps



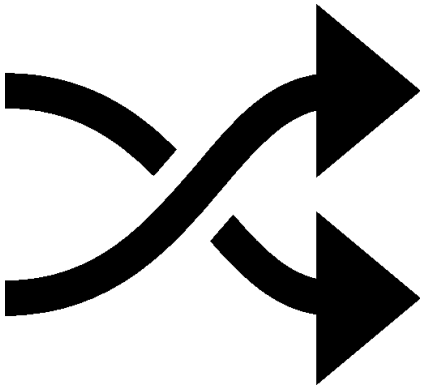


Signal	Pin	Pin	Signal
TDO	1	2	N/C
TDI	3	4	TRST-
(RUNSTOP-)	5	6	JTAG-VREF
TCK	7	8	(CHKSTPIN-)
TMS	9	10	N/C
(SRESET-)	11	12	GND
HRESET-	13	14	N/C (KEY PIN)
(CKSTOPOUT-)	15	16	GND

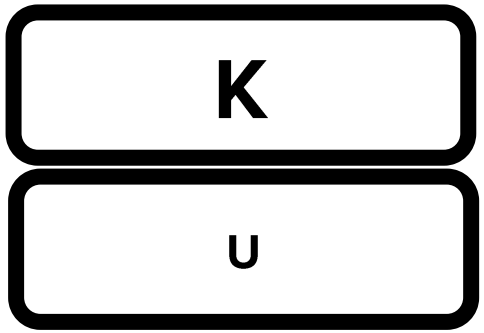
Patient monitor (lack of) security



NO DEP
(NX-BIT)

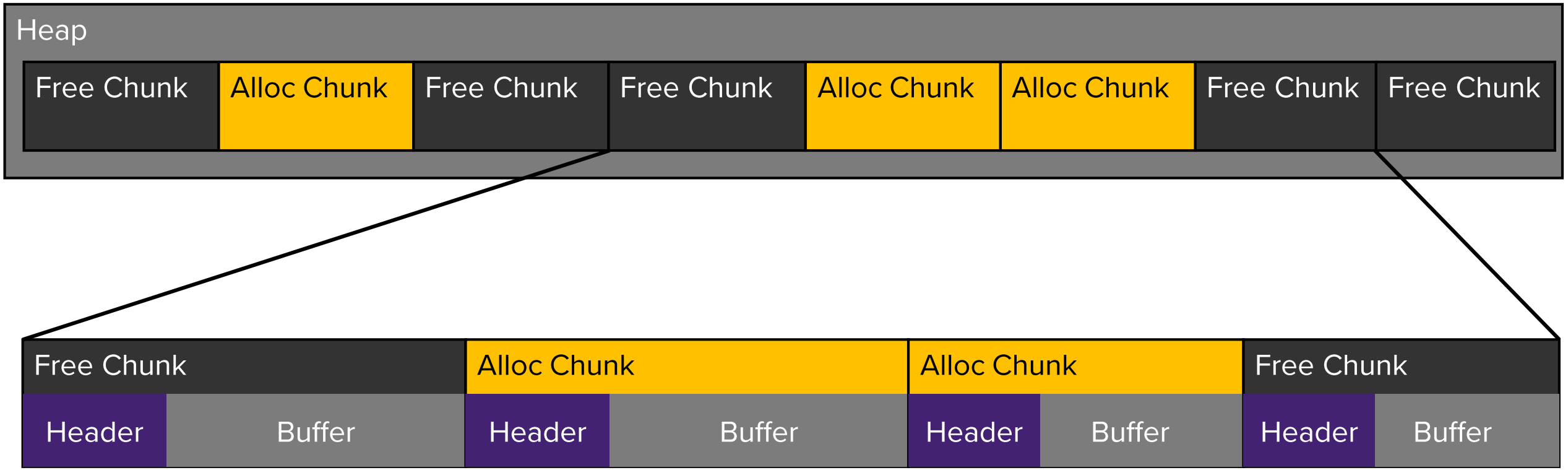


NO ASLR



NO
KERNEL/USER
SEPERATION

VxWorks Heap



VxWorks Heap

FREE_CHUNK_HDR

```
uint32_t prevSize
uint32_t size
free_chunk_hdr_t *next
free_chunk_hdr_t *prev
```

ALLOC_CHUNK_HDR

```
uint32_t prevSize
uint32_t size
mem_part *memPartId
uint32_t headGuard
```

VxWorks Heap

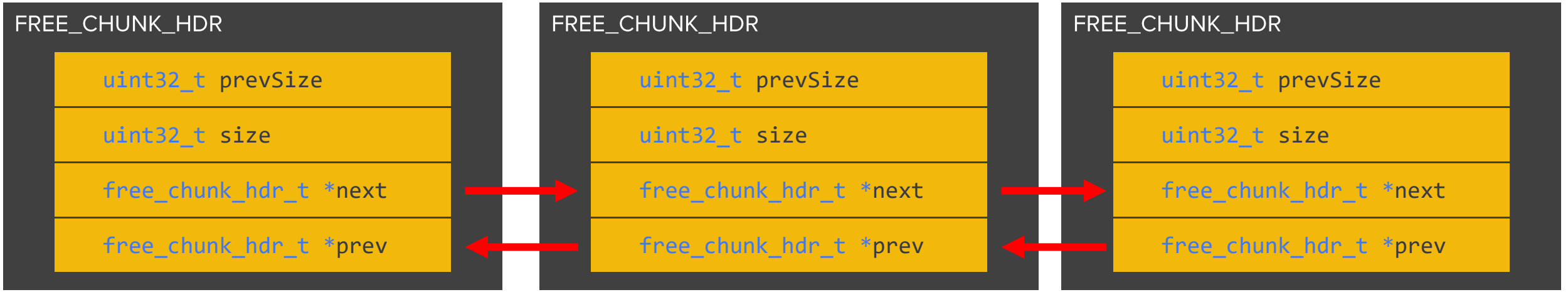
FREE_CHUNK_HDR

```
uint32_t prevSize
uint32_t size
free_chunk_hdr_t *next
free_chunk_hdr_t *prev
```

ALLOC_CHUNK_HDR

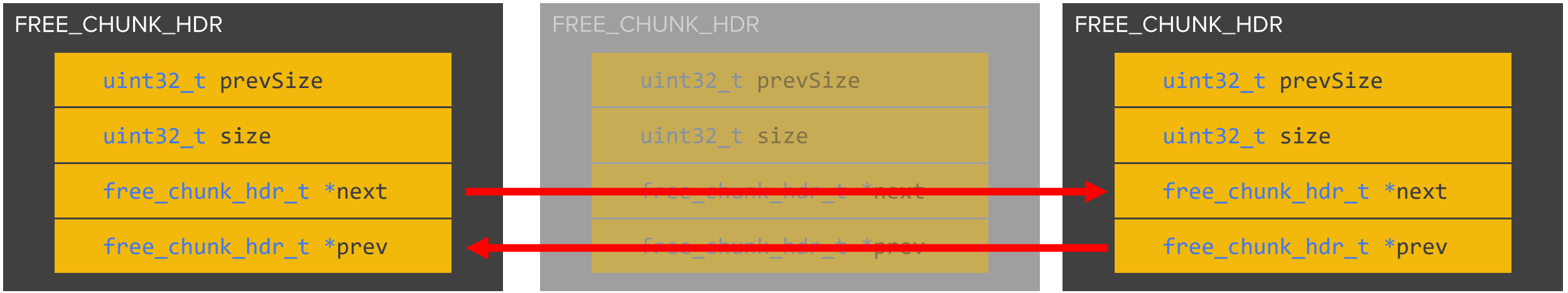
```
uint32_t prevSize
uint32_t size
mem_part *memPartId
uint32_t headGuard
```

VxWorks Heap – Free list



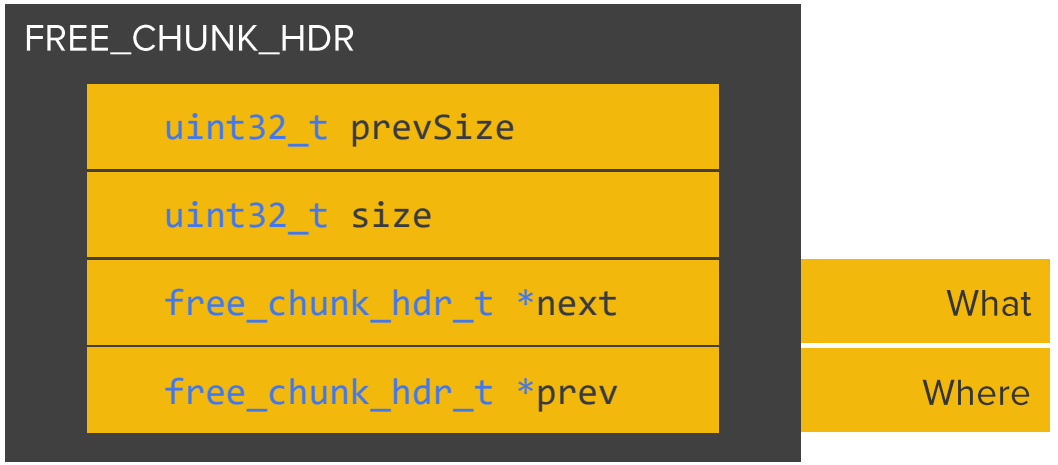
VxWorks Heap – Free list

```
void memPartDeleteFree (mem_part * partId, FREE_CHUNK_HDR * node)
{
    /* This is called inside an allocation */
    ...
    ((FREE_CHUNK_HDR *)(node->prev))->next = node->next;
    ((FREE_CHUNK_HDR *)(node->next))->prev = node->prev;
    ...
}
```

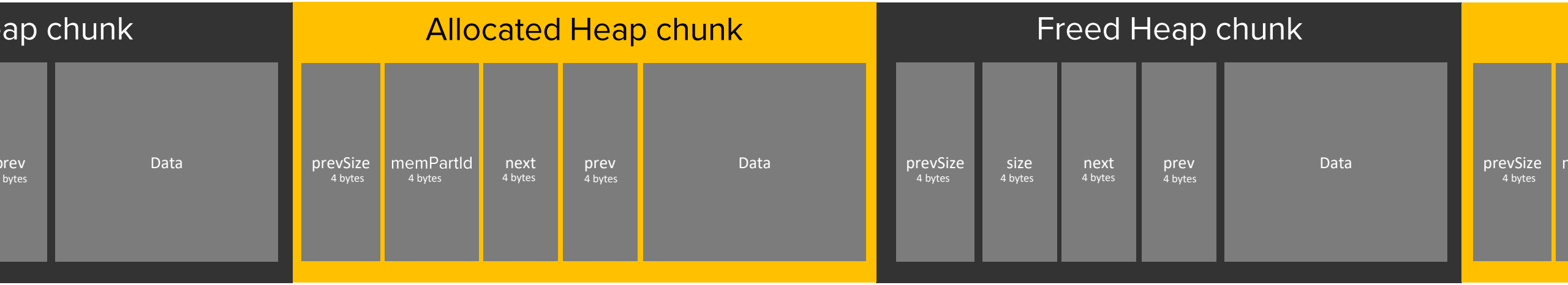


VxWorks Heap – Mirrored Write

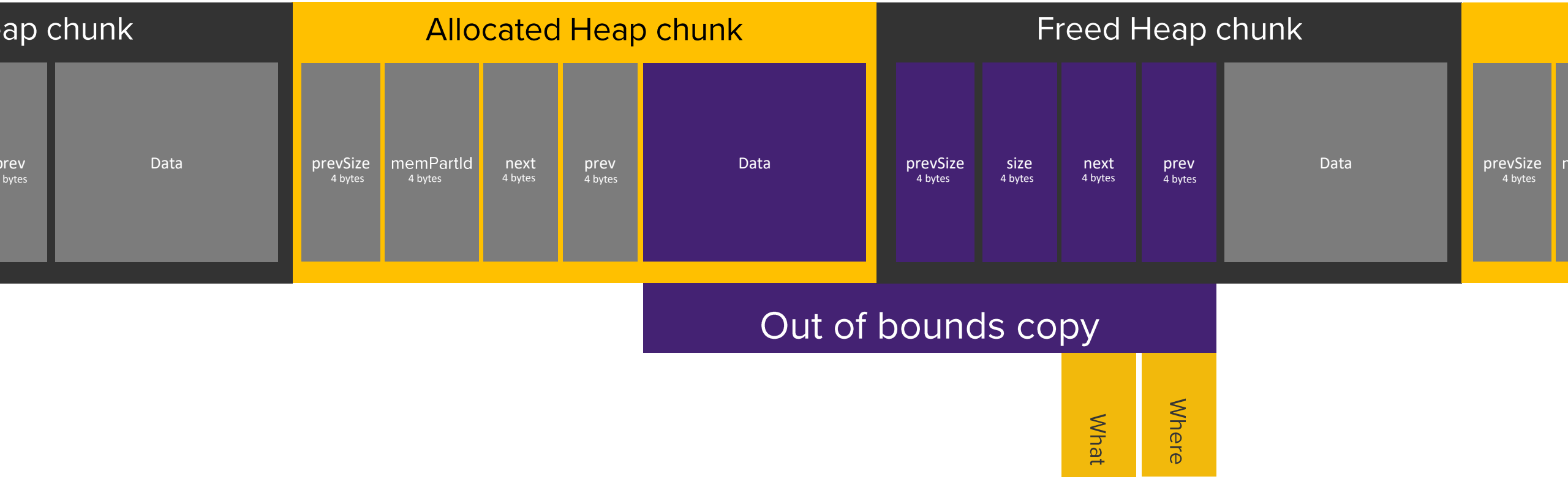
```
void memPartDeleteFree (mem_part * partId, FREE_CHUNK_HDR * node)
{
    /* This is called inside an allocation */
    ...
    ((FREE_CHUNK_HDR *)(Where))->next = What;
    ((FREE_CHUNK_HDR *)(What))->prev = Where;
    ...
}
```



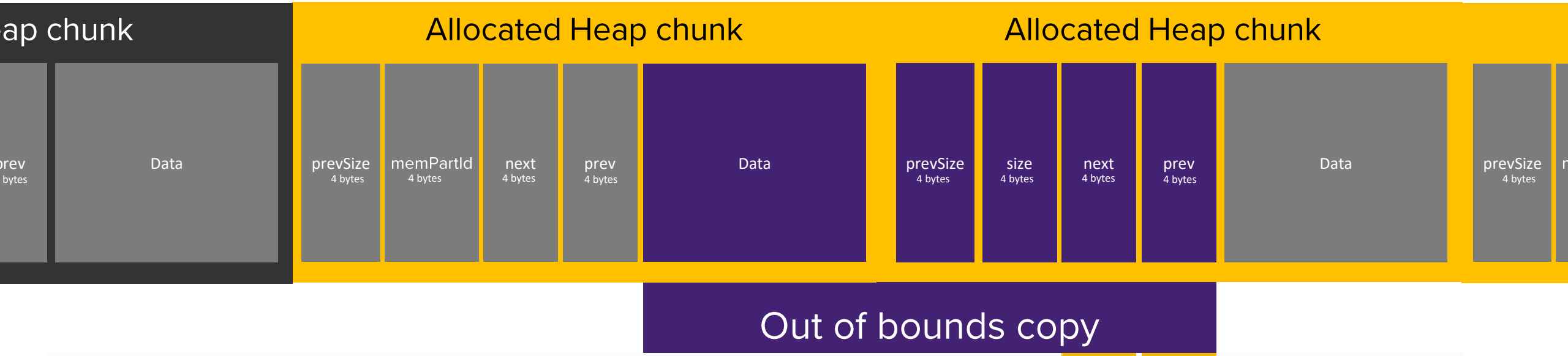
Heap exploitation strategy



Heap exploitation strategy



Heap exploitation strategy

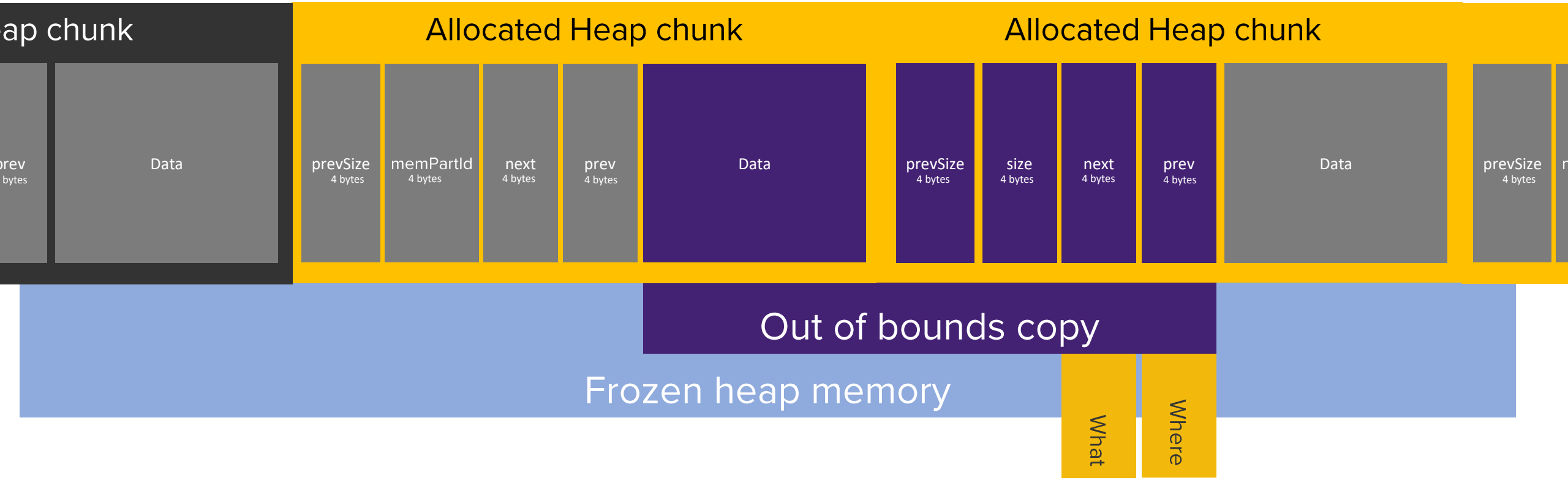



```
void memPartDeleteFree (mem_part * partId,
                        FREE_CHUNK_HDR * node)
{
    /* This is called inside an allocation */
    ...
    ((FREE_CHUNK_HDR *)(Where))->next = What;
    ((FREE_CHUNK_HDR *)(What))->prev = Where;
    ...
}
```

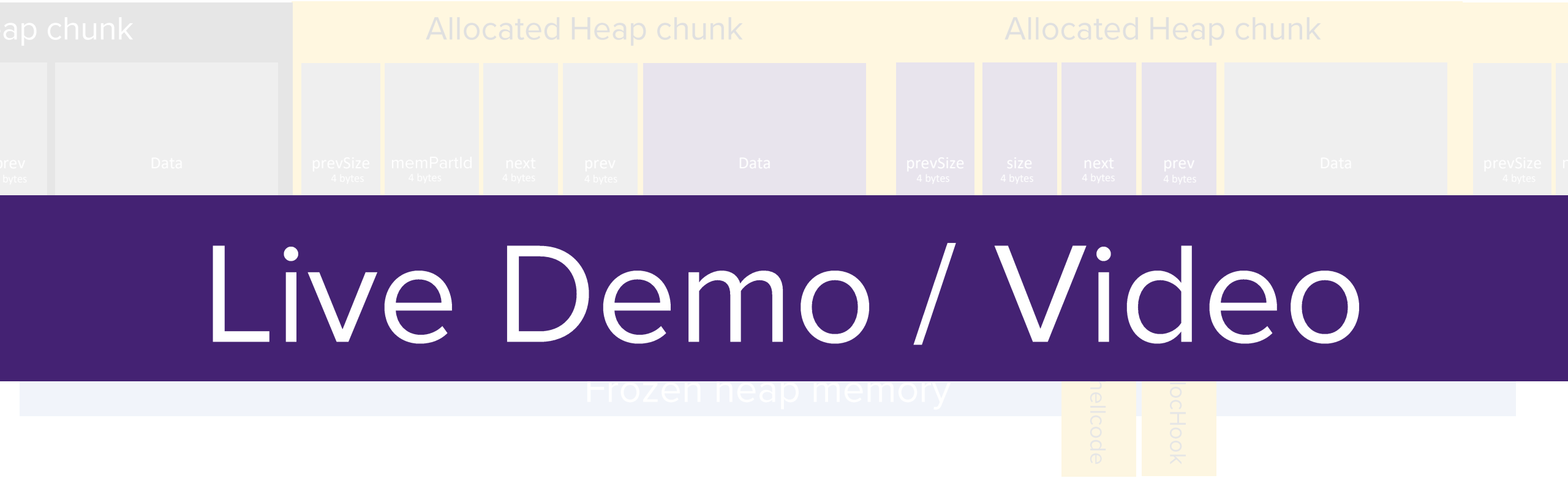
VxWorks Heap hooks

```
typedef struct mem_part
{
    ...
    /* alloc hooks */
    FUNC_ALLOC_HOOK   allocHook;     /* hook for memPartAlignedAlloc       */
    FUNC_FREE_HOOK    freeHook;      /* hook for memPartFree               */
    FUNC_REALLOC_HOOK reallocHook;   /* hook for memPartRealloc            */
    FUNC_DELETE_HOOK  deleteHook;    /* hook for memPartDelete             */
    void *            hookArg;       /* argument for hooks                 */
    ...
}
```

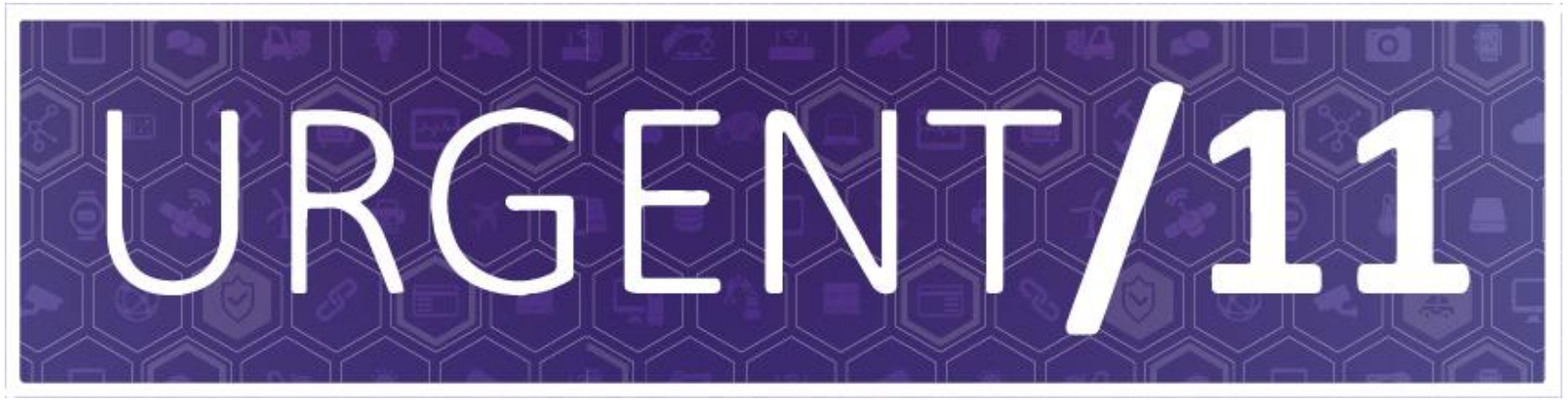
Heap exploitation strategy



Heap exploitation strategy



Live Demo / Video



Takeaways

RTOS implementations require more research

Esoteric TCP/IP features needs to be phased out

Identifying the use of underlying OS is very hard, but needed

Questions?

For more info & whitepaper:
https://armis.com/urgent11

ARMIS™