Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

sdw
Stiftung der Deutschen Wirtschaft

Descartes
research

Addressing Shortcomings of Existing DDoS Protection Software Using Software-Defined Networking

SSP 2018, Hildesheim

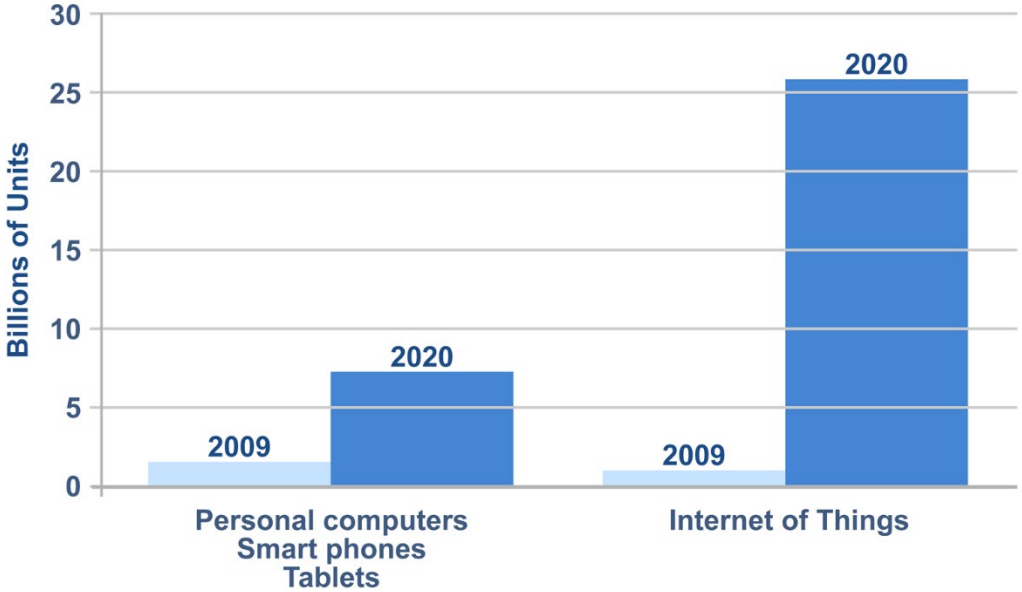
Lukas Iffländer, Stefan Geissler, Jürgen Walter, **Lukas Beierlieb, Samuel Kounev**

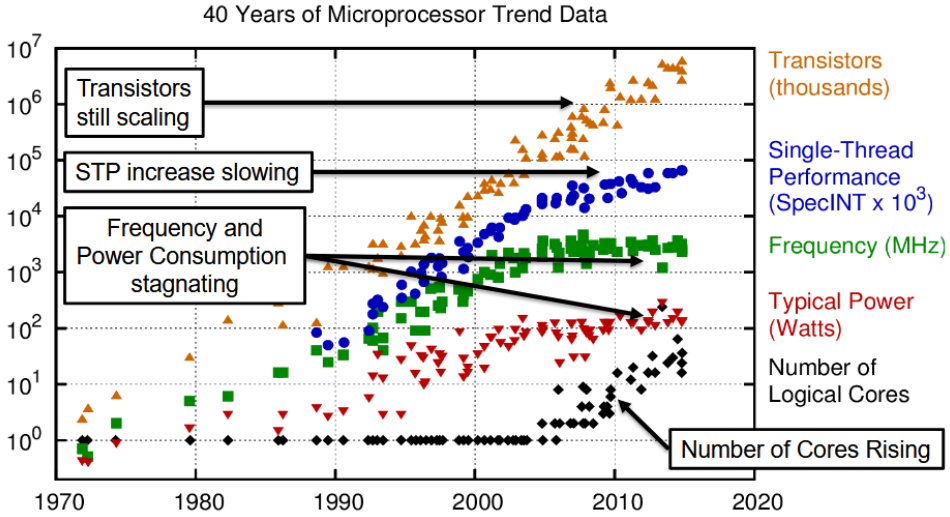
08.11.2018

http://se.informatik.uni-wuerzburg.de/

Motivation

- No definite defense possible, only mitigation
- Long time security threat
- More dangerous than ever:
  - Increasing number of IoT devices
  - Generally lower security level
- Marginal performance increase of defense systems hardware





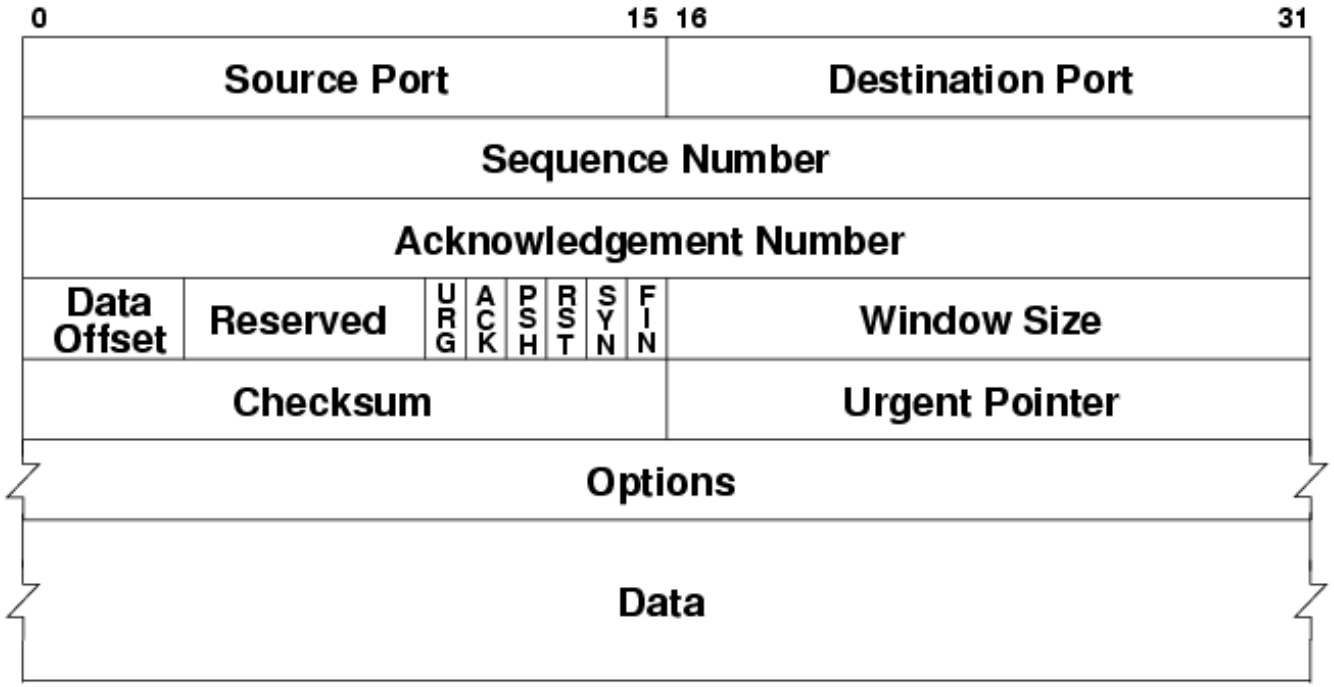
Need for more effective mitigation approaches

SYN Flood

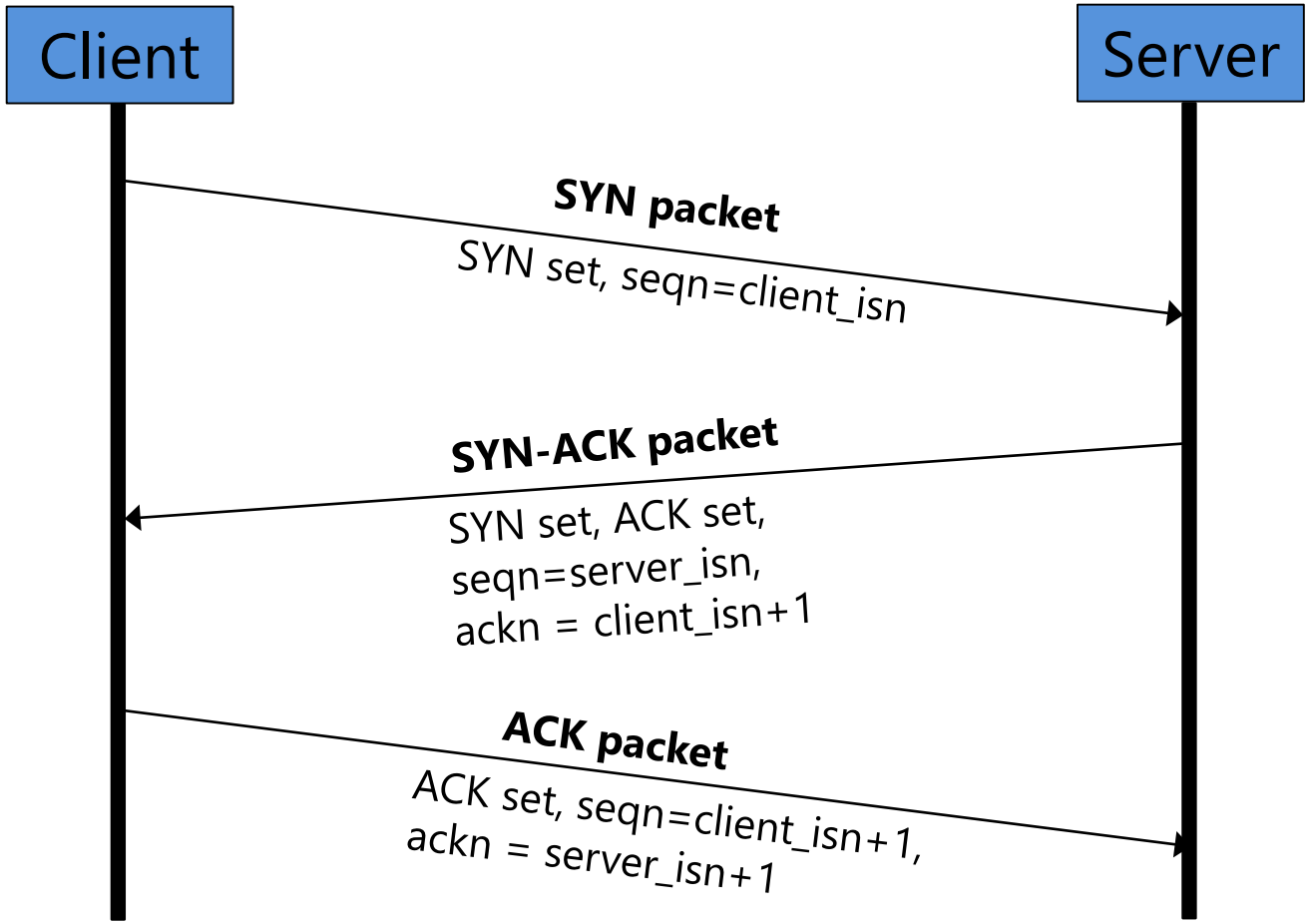
EXPLANATION

TCP

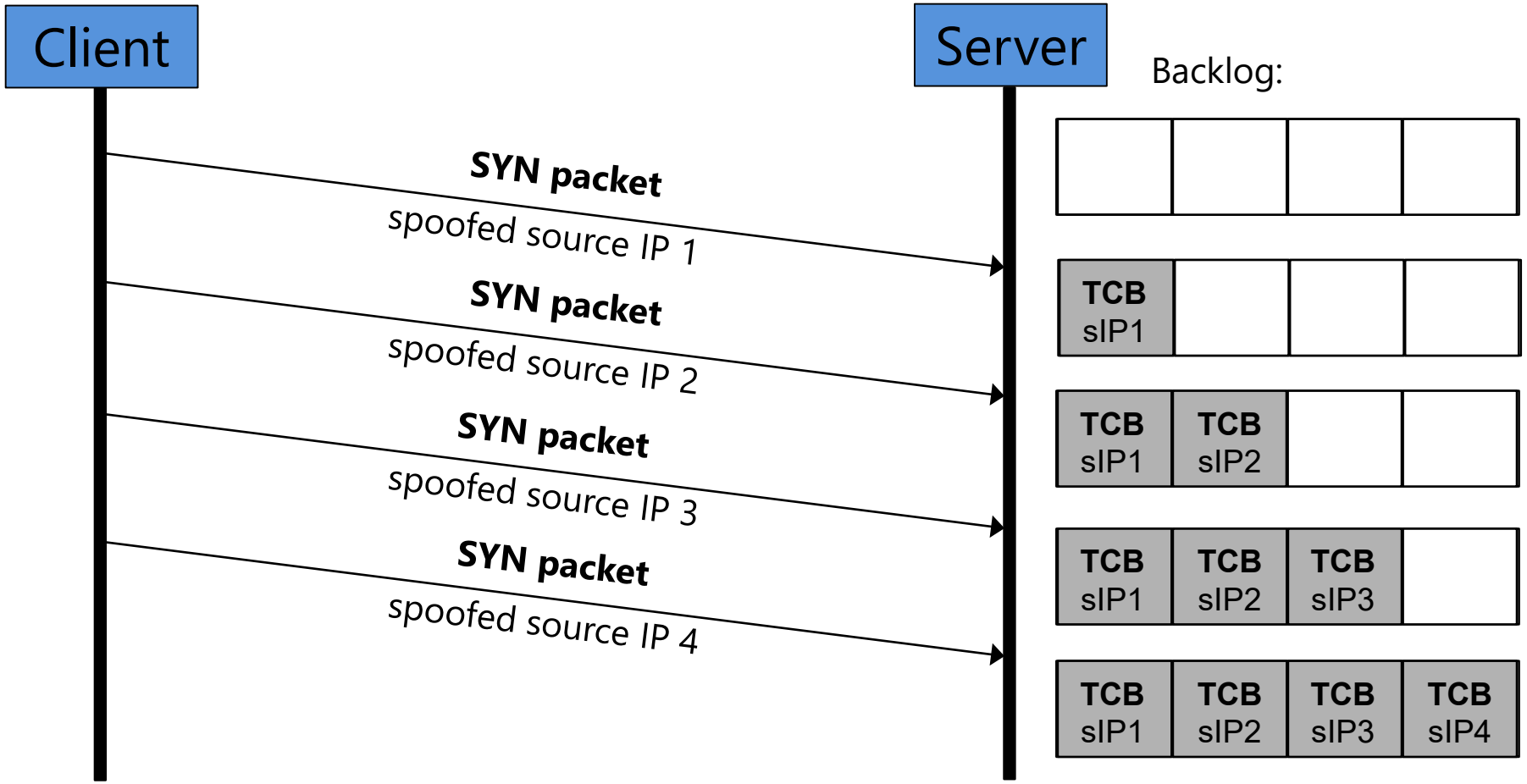
- TCP is a reliable transport protocol:
  - Retransmission of lost packets
  - Sorting of out-of-order packets
- Sequence number on every packet is necessary
- Initial sequence numbers are established in a three-way handshake



TCP Handshake



SYN Flood


Client
Server
Backlog:
SYN packet
spoofed source IP 1
SYN packet
spoofed source IP 2
SYN packet
spoofed source IP 3
SYN packet
spoofed source IP 4
TCB sIP1
TCB sIP1 TCB sIP2
TCB sIP1 TCB sIP2 TCB sIP3
TCB sIP1 TCB sIP2 TCB sIP3 TCB sIP4

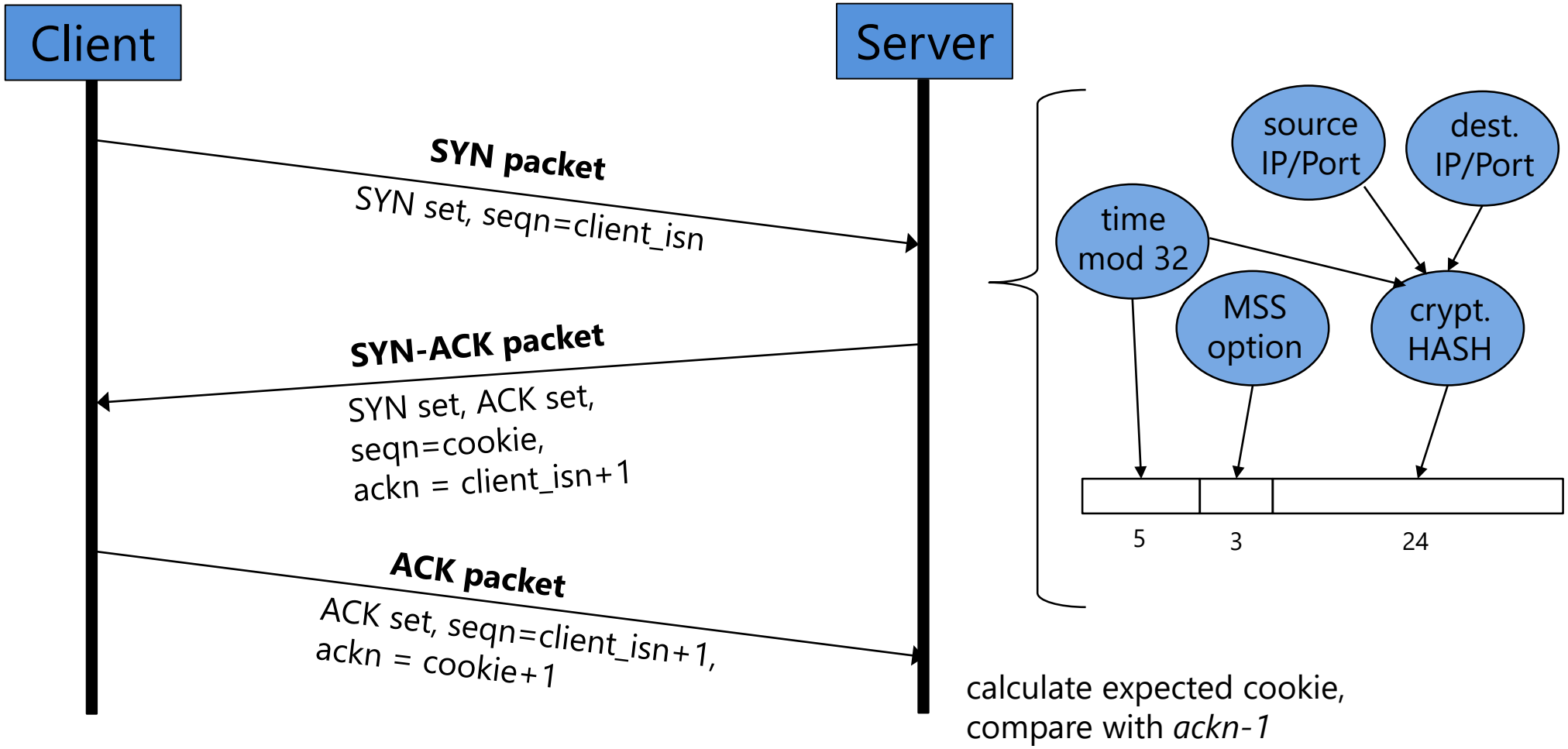
SYN Flood Explanation

- Attacker can spoof any source IP address
- Server has to create TCB and keep it for a while
- SYN packets are small:
  - 14 byte (Ethernet header) + 20 byte (IP header) + 20 byte (TCP header) = 54 byte
  - 1 Mbit/s can transport 2314 pps
  - 1 Mpps requires 432 Mbit/s

SYN Flood

EXISTING DEFENSE MECHANISMS

SYN Cookies


Client
Server
SYN packet
SYN set, seqn=client_isn
SYN-ACK packet
SYN set, ACK set,
seqn=cookie,
ackn = client_isn+1
ACK packet
ACK set, seqn=client_isn+1,
ackn = cookie+1
source IP/Port
dest. IP/Port
time mod 32
MSS option
crypt. HASH
5
3
24
calculate expected cookie,
compare with *ackn-1*

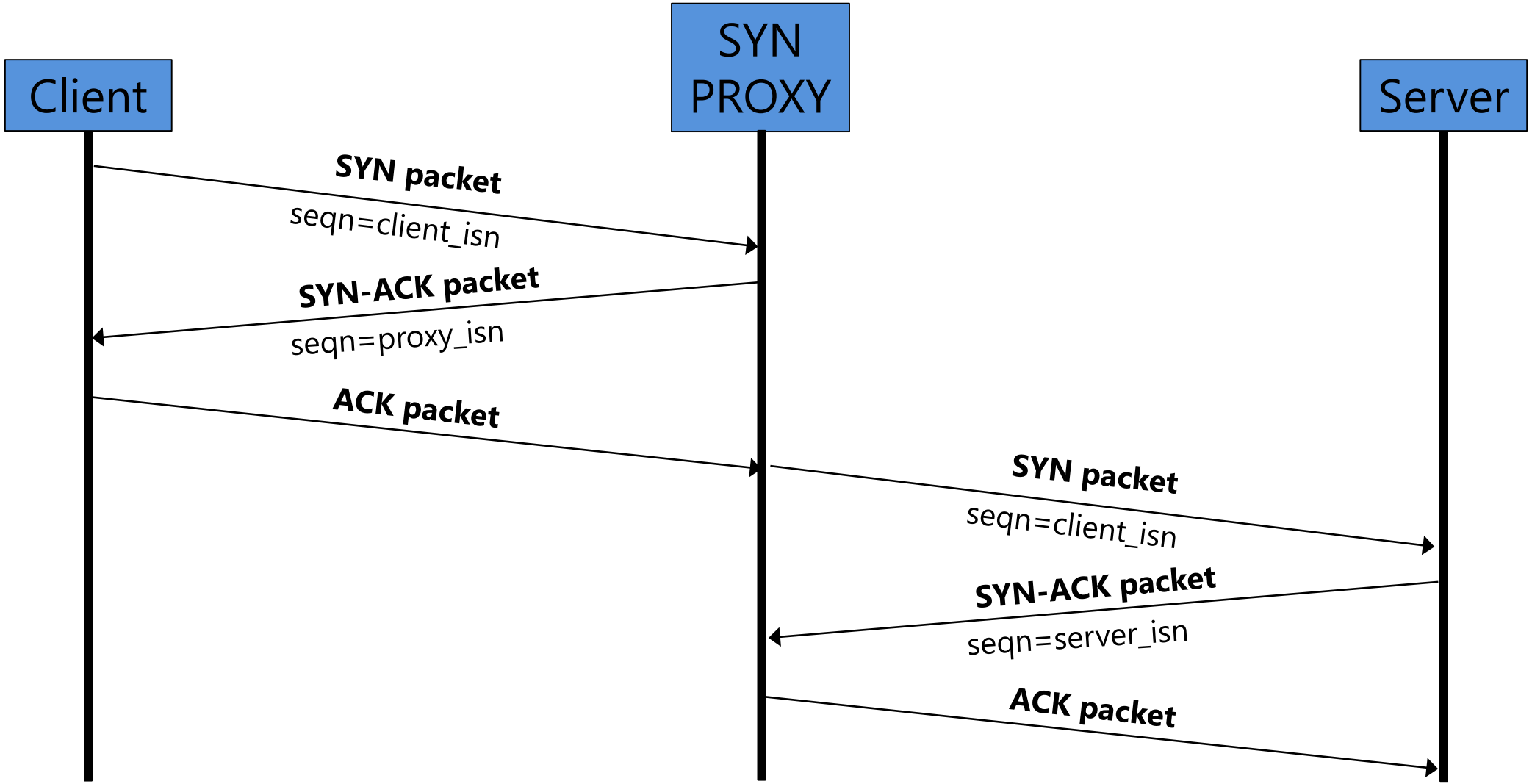
SYN Cookies

- Amount of half-open connections not limited by backlog

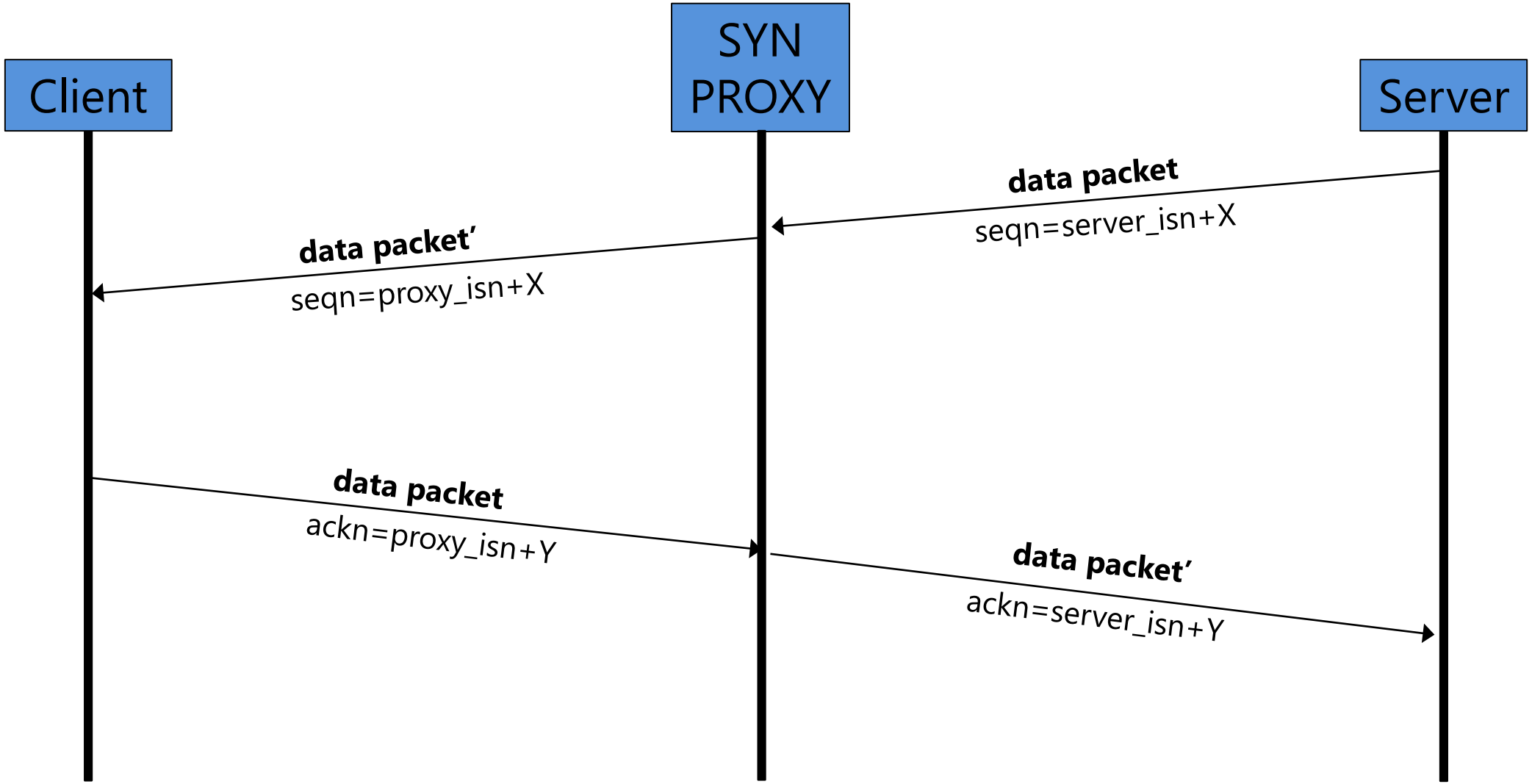
- CPU is burdened with hash calculations
- TCP options are restricted

- Only active when backlog is full

SYN PROXY – Connection Establishment



SYN PROXY – Data Transfer



SYN PROXY

- Implemented as an IPtables module
- Does not have to run on target machine
- Only complete handshakes reach target

- Proxy cannot predetermine server's ISN
- seqn/ackn translation always necessary

Limitations

SYN Cookies

- Has to run on service host
- No independent scaling

SYN PROXY

- Stateful
- Network bottle neck
- Independent scaling complex

In a Nutshell

- Problem:
  - Existing solutions can not easily be scaled indepentently from the service host
- Idea:
  - Complete handshake in proxy
  - Route subsequent packets directly
- Benefit:
  - Server handles only established connections
  - Proxy can specialize on handshake handling
- Action:
  - Develop proxy network function, utilize SDN, modify server kernel

SYN Flood

SDN/NFV APPROACH

SDN and NFV

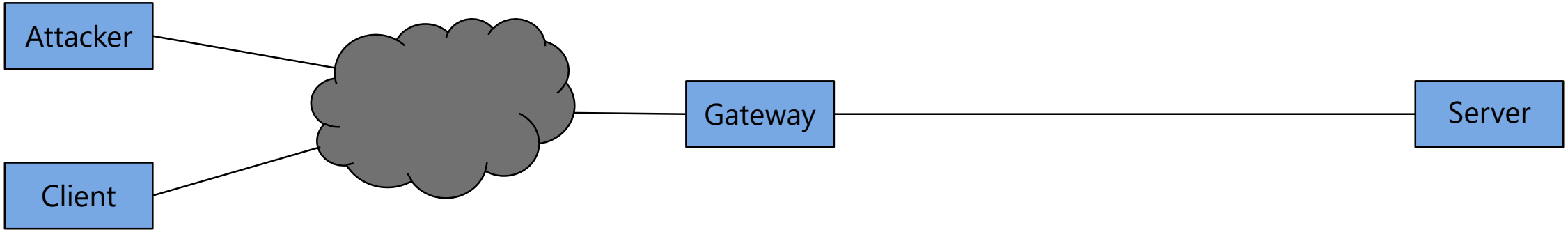
Software Defined Networking

- SDN switches' behavior is determined by a set of flows
- SDN controller modify and monitor flow sets of connected switches
- A flow consists of:
  - Match
  - Action
  - Stats

Network Function Virtualization

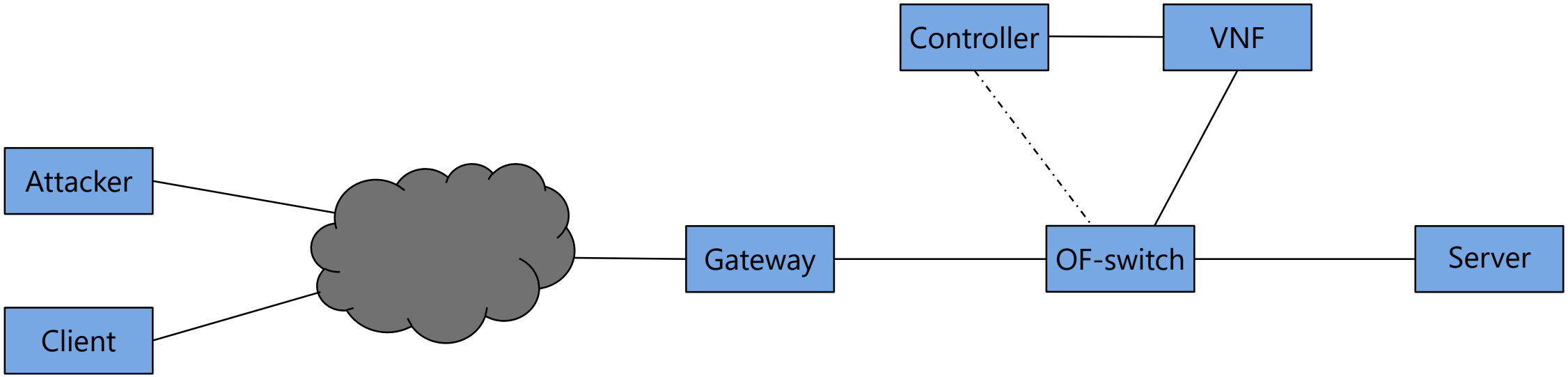
- Network functions modify packets not addressed to them
  - Firewall
  - Switch
  - IDS
- Virtualized NF is running on COTS hardware (instead of being an ASIC)

SDN/NFV Approach



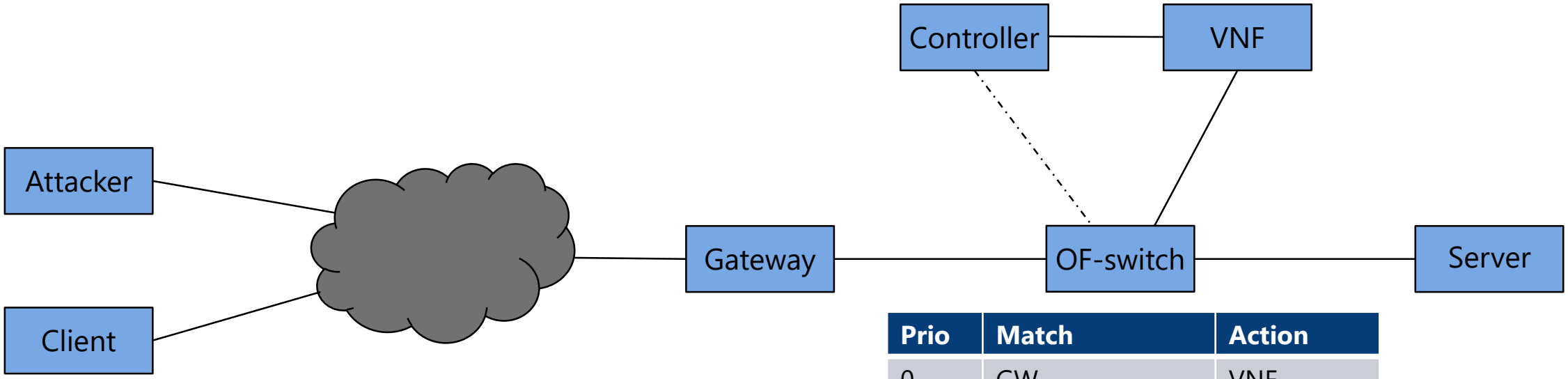
Traditional network

SDN/NFV Approach



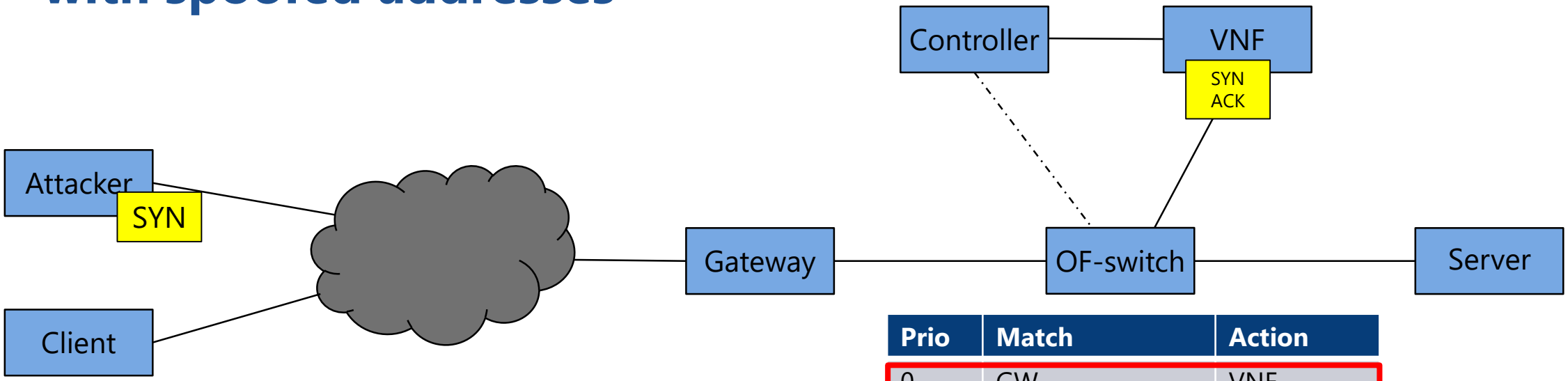
SDN enabled network

SDN/NFV Approach



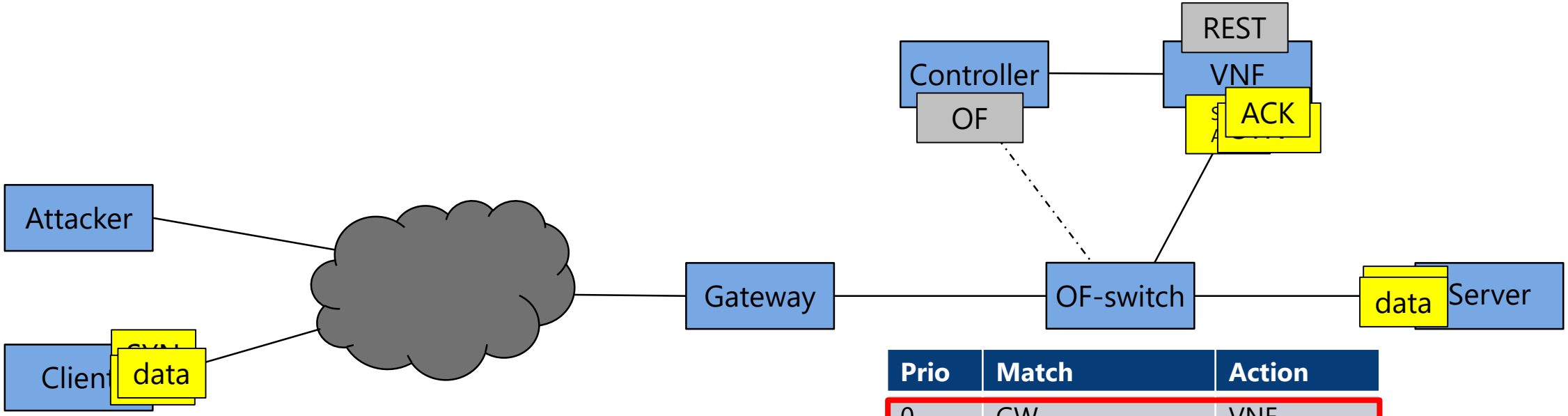
Prio	Match	Action
0	GW	VNF
0	SERV	VNF
0	VNF	GW
1	VNF, daddr=s_ip	SERV

Attacker sends SYN packets with spoofed addresses



Prio	Match	Action
0	GW	VNF
0	SERV	VNF
0	VNF	GW
1	VNF, daddr=s_ip	SERV

Client opens connection



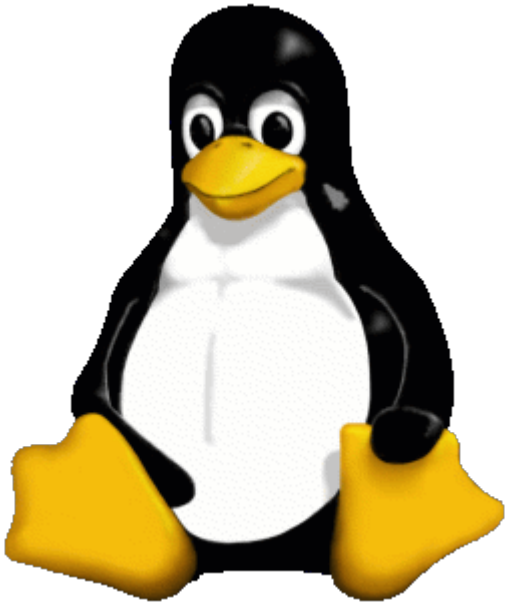
Prio	Match	Action
0	GW	VNF
0	SERV	VNF
0	VNF	GW
1	VNF, daddr=s_ip	SERV
10	GW, from client	SERV
10	SERV, to client	GW

IMPLEMENTATION

Implementation: Kernel Modification

- Simple concept
- Only 8 lines of code

- Kernel recompilation necessary
- Complete handshake required



Implementation: VNF

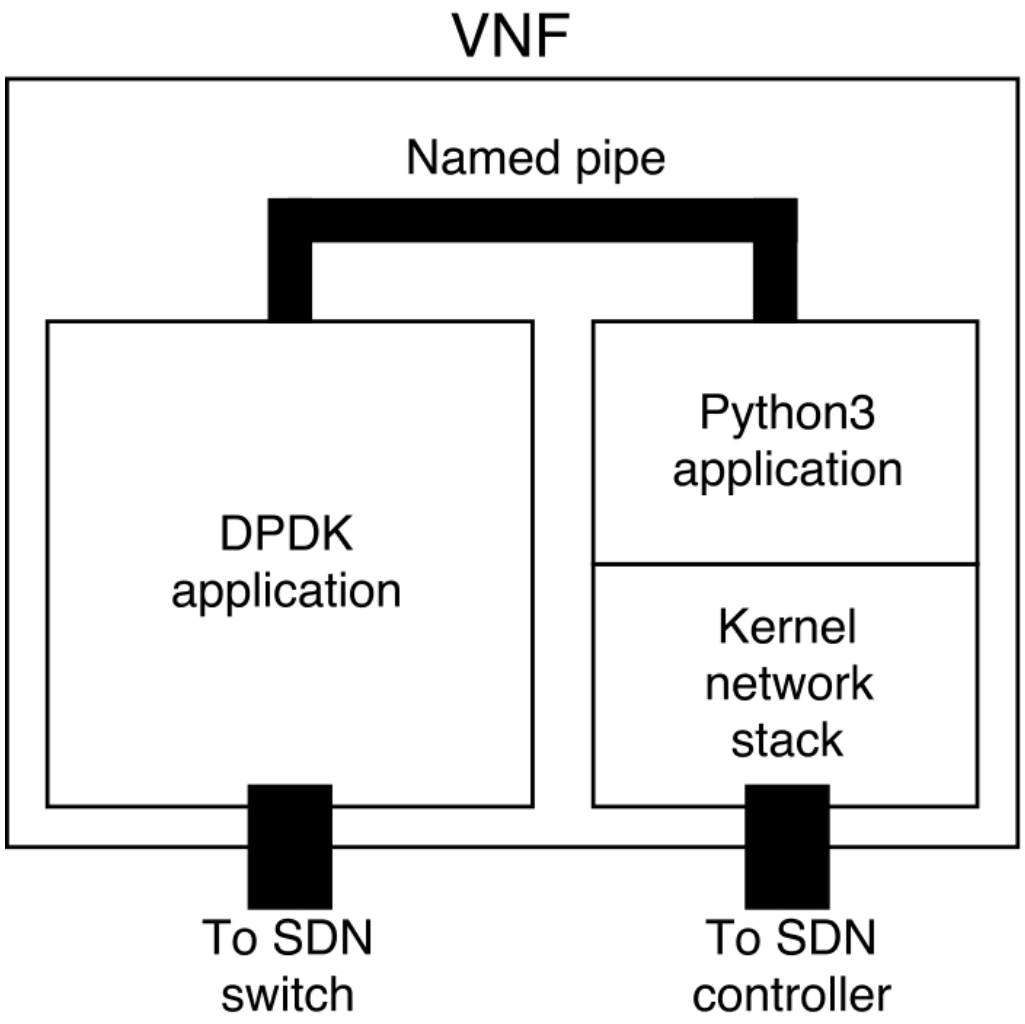
VNF is split in DPDK application and Python application

DPDK:

- Handshaking

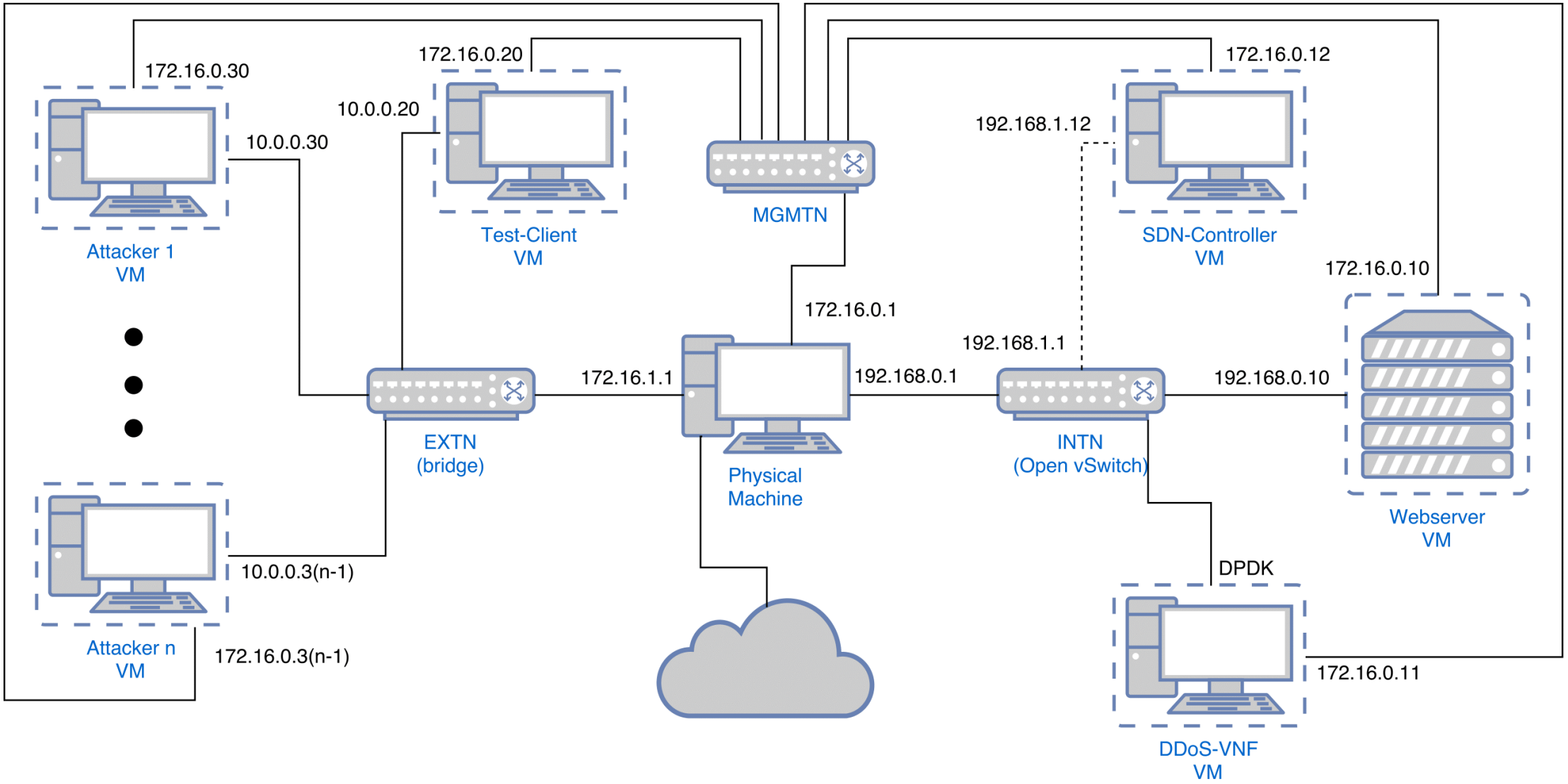
Python:

- REST requests
- (HTTP flood defense)



INITIAL EVALUATION

Virtual Testbed



Testing Methodology

- Attacker floods SYN packets with delay between each packet
- Client sends 50 SYN packets in 0.5s intervals
- Score is the amount of answered client SYN packets

First Results

	No Protection	SYN Cookies	Our Approach
AVG Successfull Connections out of 50	0	35	30.2

- Promising results for first implementation
- Feasability of idea verified

Limitations

- Client starts sending packets right after SYN-ACK
- At that moment still outstanding
  - Connection establishment with server
  - Adaptation of SDN-Configuration
- First packets are lost and have to be retransmitted.

CONCLUSION & FUTURE WORK

Conclusion

- Approach to use SDN to mitigate SYN-Flood attacks
  - Only complete handshakes reach service
  - VNF only responsible for establishing new connection
- Performant because of DPDK
- Scalable, since there are no dependencies and stateless

Future Work

- Evaluation with hardware NICs
- Comparison between different archetectures for multi core
- Move Kernel modification in standalone module
- Analyze how to circumvent packet-loss problem
- On demand enabling
- Click & Use deployment (Docker/VM/Ansible)

Thank you for your attention!

Phone: +49 (931) 31 89947
Mail: lukas.ifflaender@uni-wuerzburg.de
Web: https://go.uniwue.de/ifflaender

Bibliography

[BBCC14] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "Openstate: programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 44–51, 2014.

[BNR+17] J. Boite, P.-A. Nardin, F. Rebecchi, M. Bouet, and V. Conan, "Statesec: Stateful monitoring for ddos protection in software defined networks," 2017.

[Bor12] D. Borman, "Tcp options and maximum segment size (mss)," 2012.

[D+04] L. Deri *et al.*, "Improving passive packet capture: Beyond device polling," in *Proceedings of SANE*, vol. 2004. Amsterdam, Netherlands, 2004, pp. 85–93.

[DYL+12] Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High performance network virtualization with sr-iov," *Journal of Parallel and Distributed Computing*, vol. 72, no. 11, pp. 1471–1480, 2012.

[dyn] [Online]. Available: http://www.bbc.com/news/technology-37728015

[Edd07] W. M. Eddy, "Tcp syn flooding attacks and common mitigations," 2007.

[FM15] C. J. Fung and B. McCormick, "Vguard: A distributed denial of service attack

...

BACKUP

Implementation: Kernel Modification

from *net/ipv4/tcp_ipv4.c*:

```
static u32 tcp_v4_init_seq(const struct sk_buff *skb)
{
    return secure_tcp_seq(ip_hdr(skb)->daddr,
                          ip_hdr(skb)->saddr,
                          tcp_hdr(skb)->dest,
                          tcp_hdr(skb)->source);
}
```

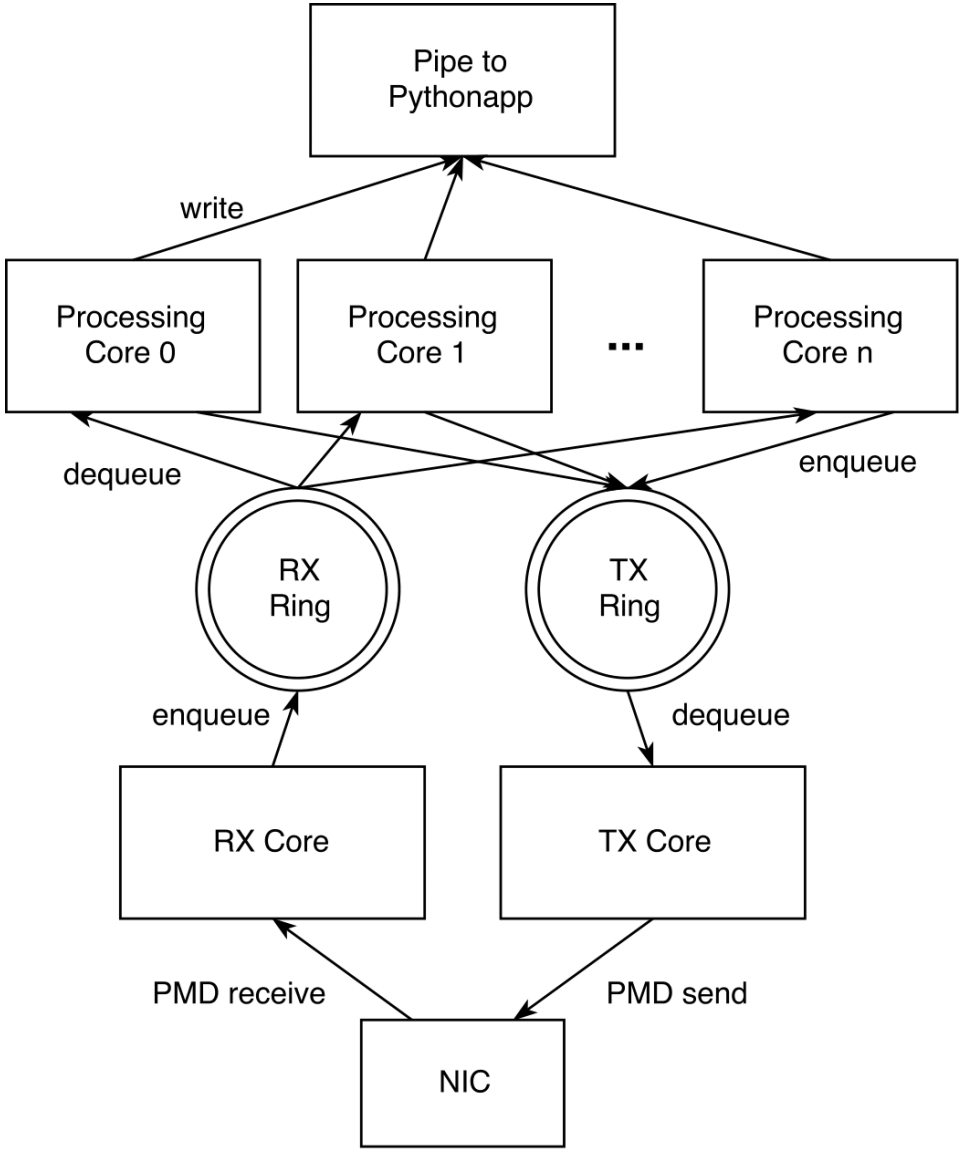
Implementation: Kernel Modification

```
static u32 tcp_v4_init_seq(const struct sk_buff *skb)
{
    struct tcphdr *tcph;
    unsigned char *payload_start, *payload_end;
    tcph = tcp_hdr(skb);
    payload_start = (unsigned char *)((unsigned char *)tcph+(tcph->doff * 4));
    payload_end = skb_tail_pointer(skb);

    if (payload_end - payload_start == 4)
        return cpu_to_be32(*((u32 *)payload_start));

    return secure_tcp_seq(ip_hdr(skb)->daddr,
                          ip_hdr(skb)->saddr,
                          tcp_hdr(skb)->dest,
                          tcp_hdr(skb)->source);
}
```

DPDK Features

- Poll-mode, userspace NIC drivers
- Lockless ringbuffer
- Cryptographic library
- Packet processing library (e.g. Fragmentation)
- Kernel Network Interfaces
- Hugepage support (and requirement) for less TLB misses
- Thread-to-core pinning
- NUMA support

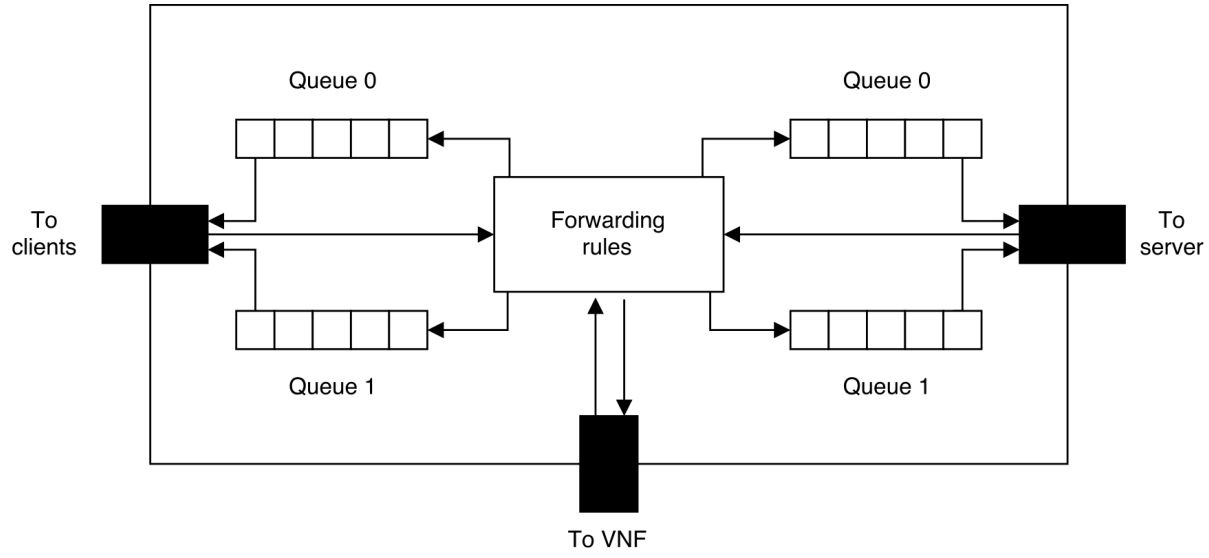
DPDK Application



Python application

SYN Flood Defense

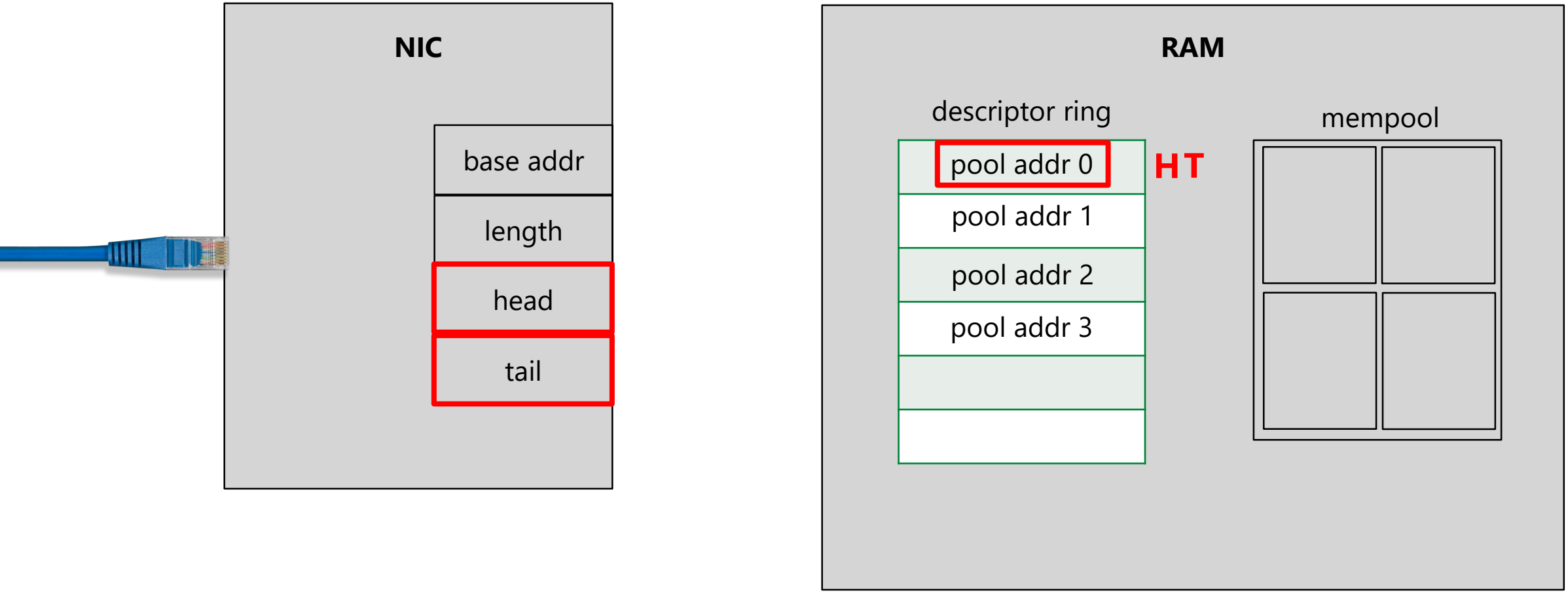
- Translates address and port from pipe into flow modification REST request



HTTP Flood Defense

- Monitor flow statistics
- Aggregate per IP address
- Requeue suspicious addresses

NICs and Drivers



Kernel Modification Validation

```
echo -n -e '\x01\x23\x45\x67' > payload
hping3 --count 1 --syn --dest-port 80 --data 4 --file payload 192.168.0.10
```

Kernel Modification Validation

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.30	192.168.0.10	TCP	58	2489 → 80 [SYN] Seq=2081396469 Win=512 Len=4
2	0.000241677	192.168.0.10	10.0.0.30	TCP	60	80 → 2489 [SYN, ACK] Seq=19088743 Ack=2081396470

```
▶ Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
▶ Ethernet II, Src: 0a:76:c6:e9:8f:43 (0a:76:c6:e9:8f:43), Dst: DecObsol_00:00:00 (aa:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 10.0.0.30, Dst: 192.168.0.10
▼ Transmission Control Protocol, Src Port: 2489, Dst Port: 80, Seq: 2081396469, Len: 4
    Source Port: 2489
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 4]
    Sequence number: 2081396469
    [Next sequence number: 2081396474]
  ▶ Acknowledgment number: 510359386
    0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x002 (SYN)
    Window size value: 512
    [Calculated window size: 512]
    Checksum: 0xe9b0 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    TCP payload (4 bytes)
```

```
0000  aa 00 00 00 00 00 0a 76  c6 e9 8f 43 08 00 45 00   .......v ...C..E.
0010  00 2c f9 93 00 00 3f 06  b7 68 0a 00 00 1e c0 a8   .,....?. .h......
0020  00 0a 09 b9 00 50 7c 0f  96 f5 1e 6b 77 5a 50 02   .....P|. ...kwZP.
0030  02 00 e9 b0 00 00 01 23  45 67                     ......# Eg
```

Kernel Modification Validation

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.30	192.168.0.10	TCP	58	2489 → 80 [SYN] Seq=2081396469 Win=512 Len=4
2	0.000241677	192.168.0.10	10.0.0.30	TCP	60	80 → 2489 [SYN, ACK] Seq=19088743 Ack=2081396470

```
▶ Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▶ Ethernet II, Src: DecObsol_00:00:00 (aa:00:00:00:00:00), Dst: 0a:76:c6:e9:8f:43 (0a:76:c6:e9:8f:43)
▶ Internet Protocol Version 4, Src: 192.168.0.10, Dst: 10.0.0.30
▼ Transmission Control Protocol, Src Port: 80, Dst Port: 2489, Seq: 19088743, Ack: 2081396470, Len: 0
    Source Port: 80
    Destination Port: 2489
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 19088743
    Acknowledgment number: 2081396470
    0110 .... = Header Length: 24 bytes (6)
  ▶ Flags: 0x012 (SYN, ACK)
    Window size value: 29200
    [Calculated window size: 29200]
    Checksum: 0xf79c [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (4 bytes), Maximum segment size
```

```
0000  0a 76 c6 e9 8f 43 aa 00  00 00 00 00 08 00 45 00   .v...C.. ......E.
0010  00 2c 00 00 40 00 40 06  6f fc c0 a8 00 0a 0a 00   .,..@.@. o.......
0020  00 1e 00 50 09 b9 01 23  45 67 7c 0f 96 f6 60 12   ...P...# Eg|...`.
0030  72 10 f7 9c 00 00 02 04  05 b4 00 00               r....... ....
```

THREADS Validation

iface	Δtime [μs]	source	dest	flags	datalen [byte]
client0	-	10.0.0.20 : 55510	192.168.0.10 : 80	SYN	0
client0	369	192.168.0.10 : 80	10.0.0.20 : 55510	SYN,ACK	0
client0	77	10.0.0.20 : 55510	192.168.0.10 : 80	ACK	0
client0	122	10.0.0.20 : 55510	192.168.0.10 : 80	ACK	193
serv0	11	10.0.0.20 : 55510	192.168.0.10 : 80	SYN	4
serv0	98	192.168.0.10 : 80	10.0.0.20 : 55510	SYN,ACK	0
serv0	1212	10.0.0.20 : 55510	192.168.0.10 : 80	ACK	0
client0	195971	10.0.0.20 : 55510	192.168.0.10 : 80	ACK	193
serv0	91	10.0.0.20 : 55510	192.168.0.10 : 80	ACK	193