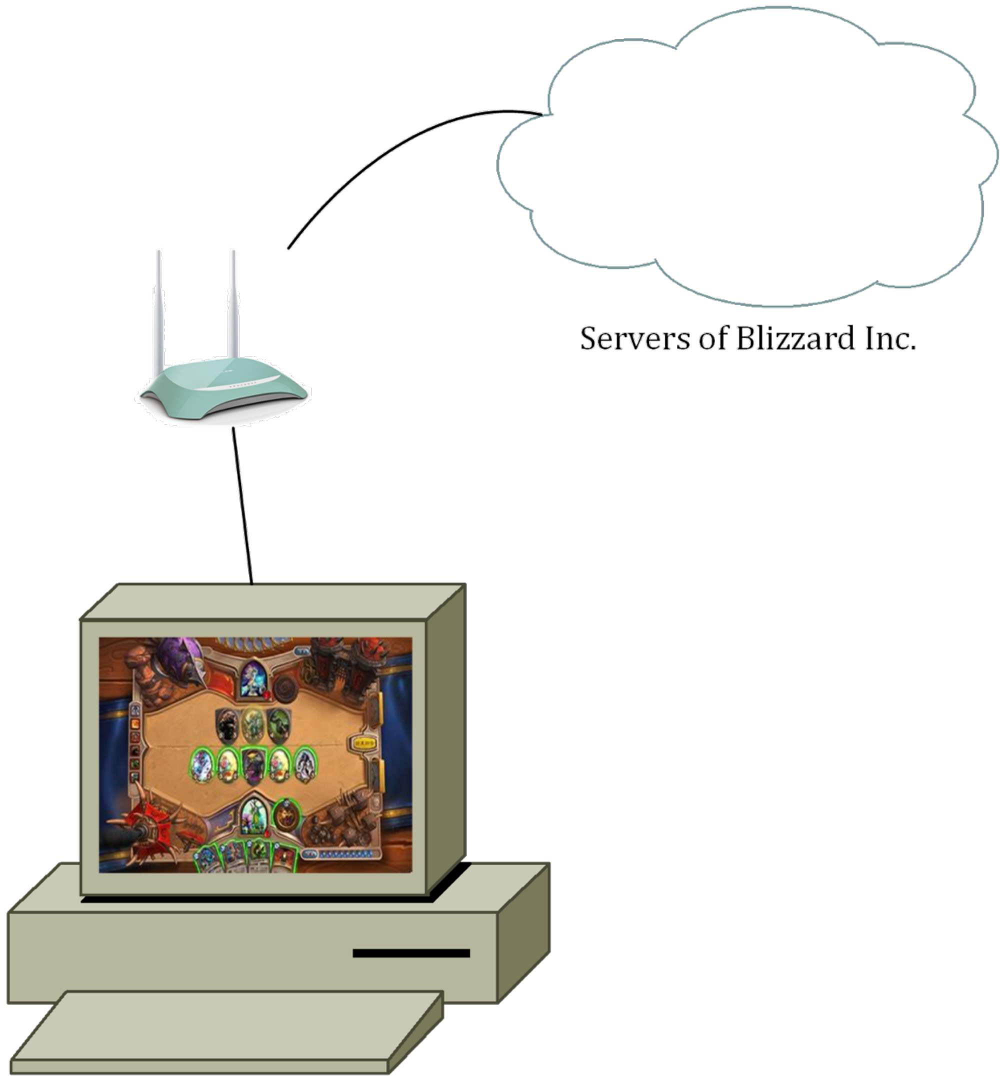
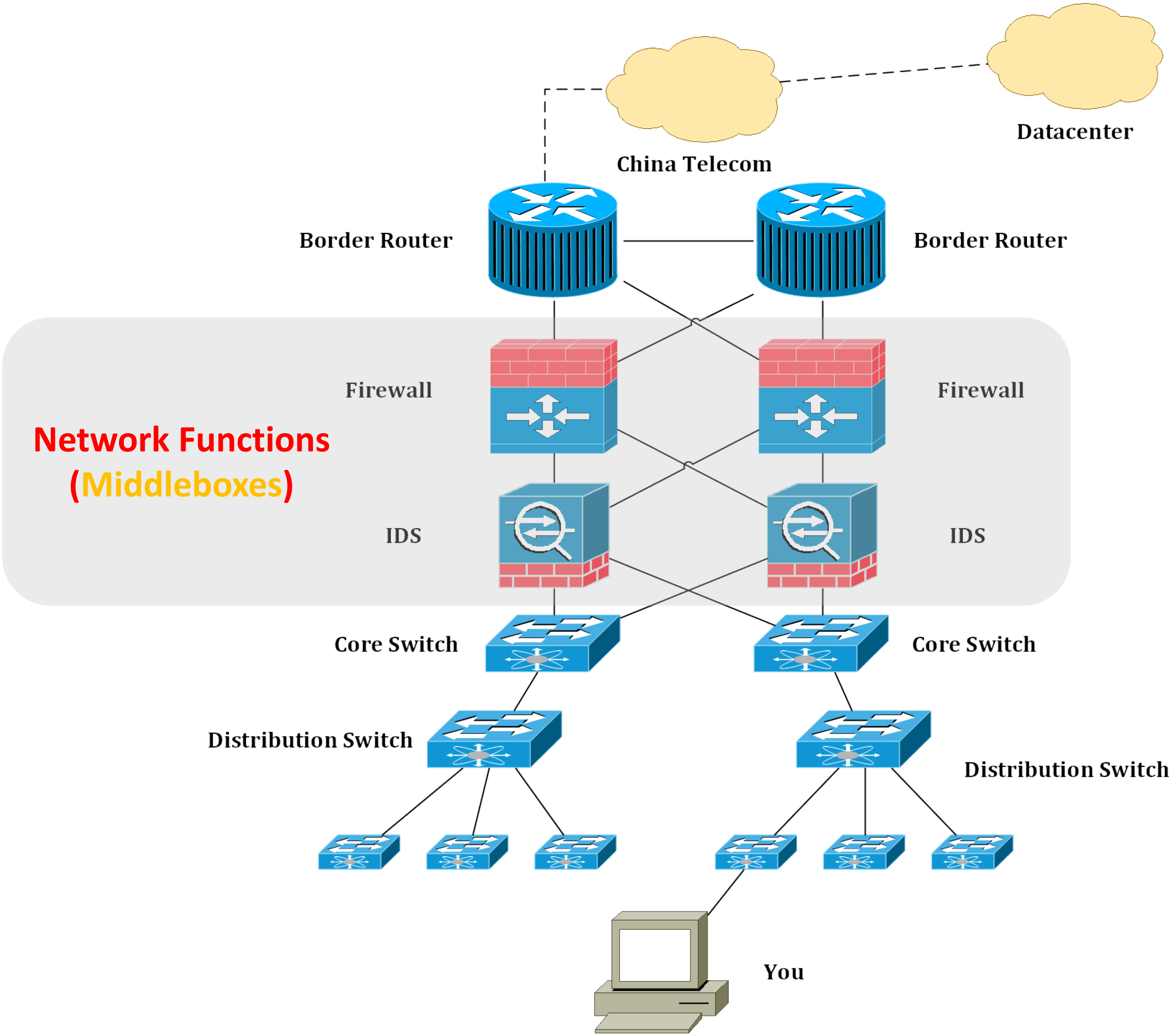
Network Function Virtualization

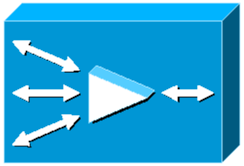
Haibo Chen
Institute of Parallel and Distributed Systems (IPADS)
Shanghai Jiao Tong University
http://ipads.se.sjtu.edu.cn/haibo_chen

How do you access servers?



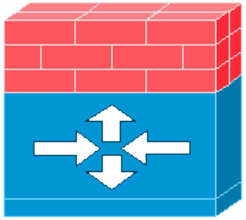


China Telecom
Datacenter
Border Router
Border Router
Network Functions
(Middleboxes)
Firewall
Firewall
IDS
IDS
Core Switch
Core Switch
Distribution Switch
Distribution Switch
You



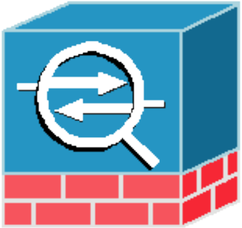
WAN Optimizer

$235,000



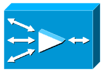
Firewall

$4,923



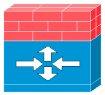
Intrusion Detection System

$43,700



WAN Optimizer

$235,000



Firewall

$4,923

Intrusion Detection System

$43,700

Intel E5-2650 v3 with 20 cores, 40 Gbps Ethernet NIC, 64 GB RAM

￥45,000

Goal of NFV

- Save money
  - Commodity server hardware
  - Workload consolidation
  - Lower power consumption
  - Simplified network maintenance

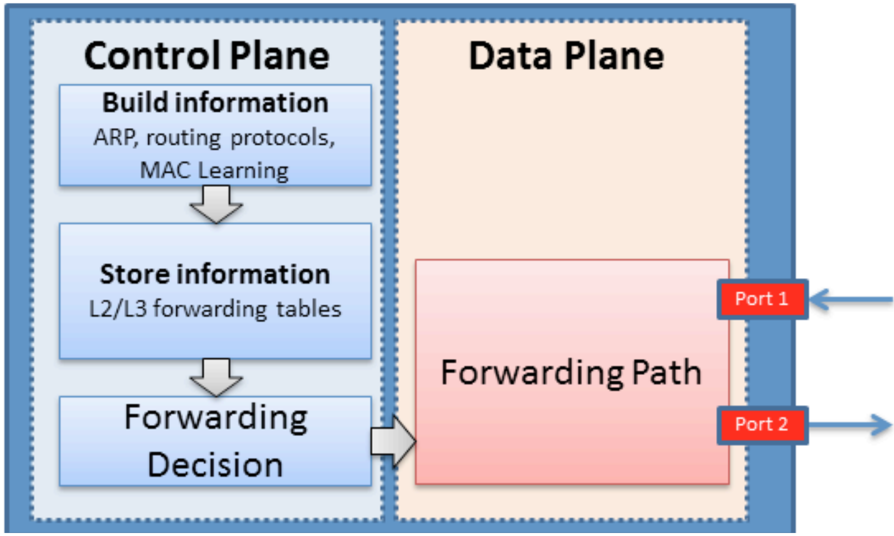
- Make money
  - Accelerated service deployment
  - Network infrastructure as a service

What is NFV?

- Network Function
  - Middleboxes (firewall, IDS, WAN optimizer)
- Virtualization
  - Commodity hardware
  - Consolidation
- **Network Services** running on **Cloud Infrastructure**

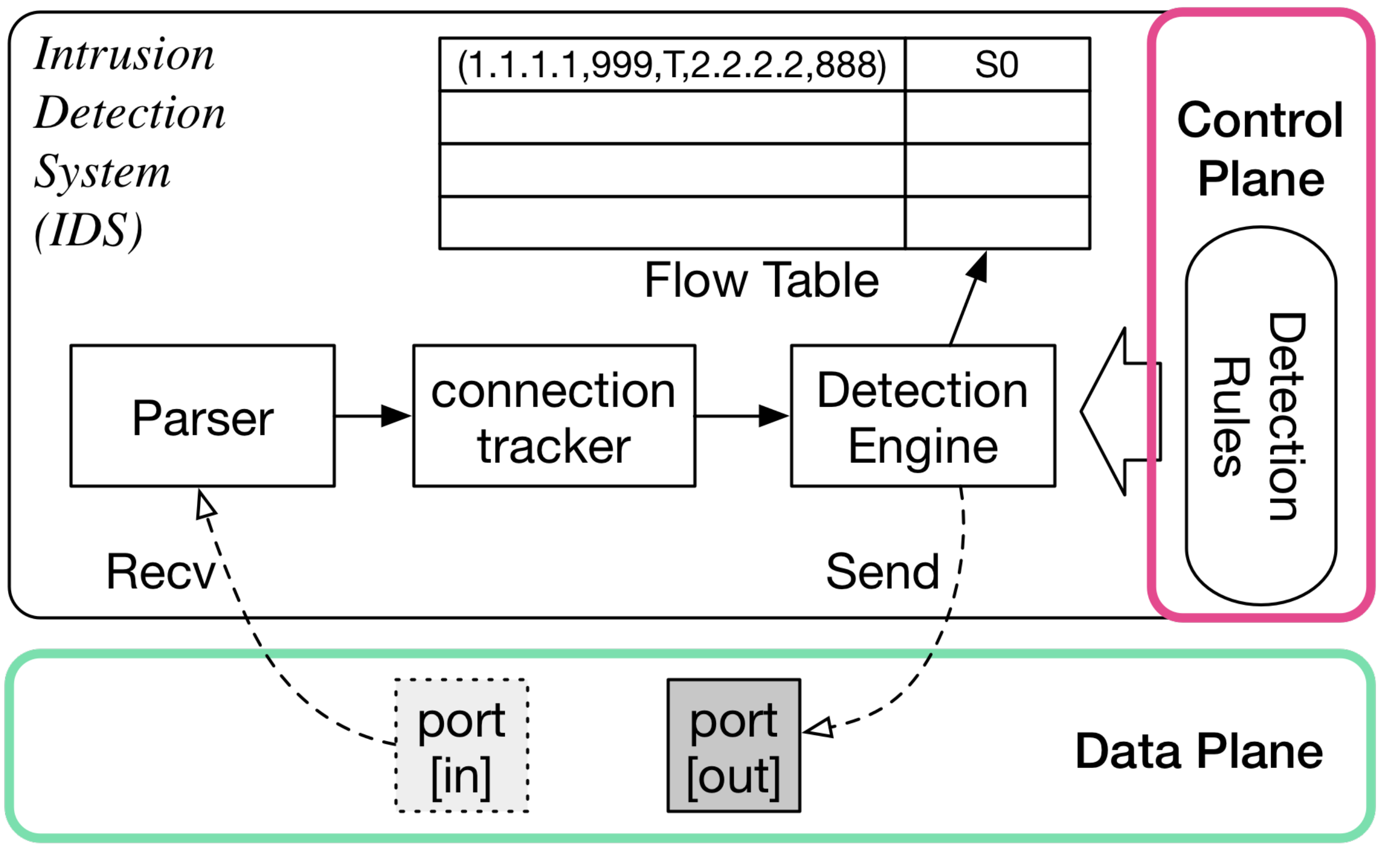
Relationship with SDN

- SDN: Software-defined networking
  - Focus on control plane innovation/virtualization
- NFV: network function virtualization
  - Focus on virtualizing data plane
- SDN & NFV
  - Can innovate separately
  - Or together
  - Like policy vs. mechanism

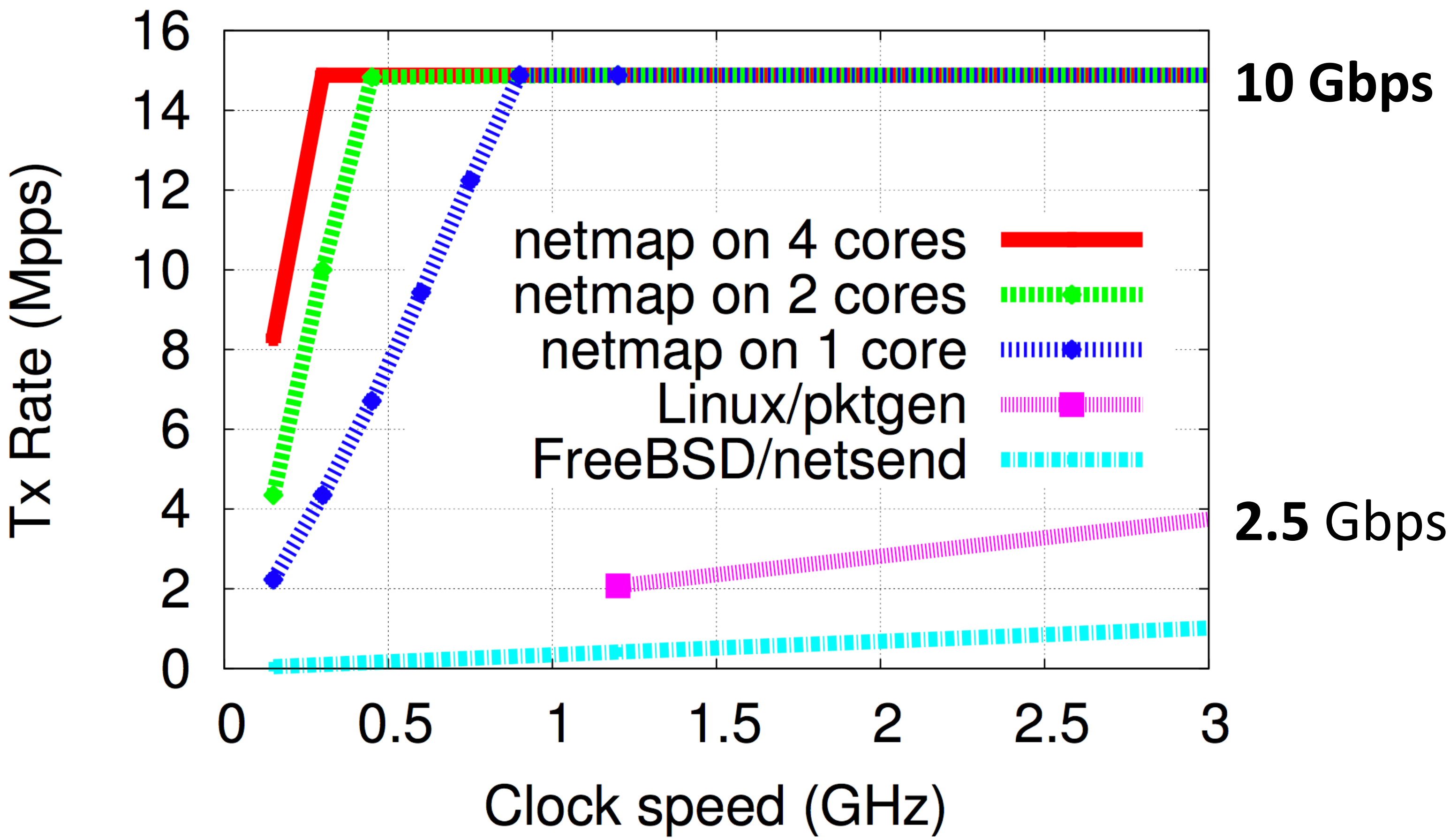


NFV: CHALLENGES

Challenges: Data Plane Performance



Challenges: Data Plane Performance



Challenges: Consolidation

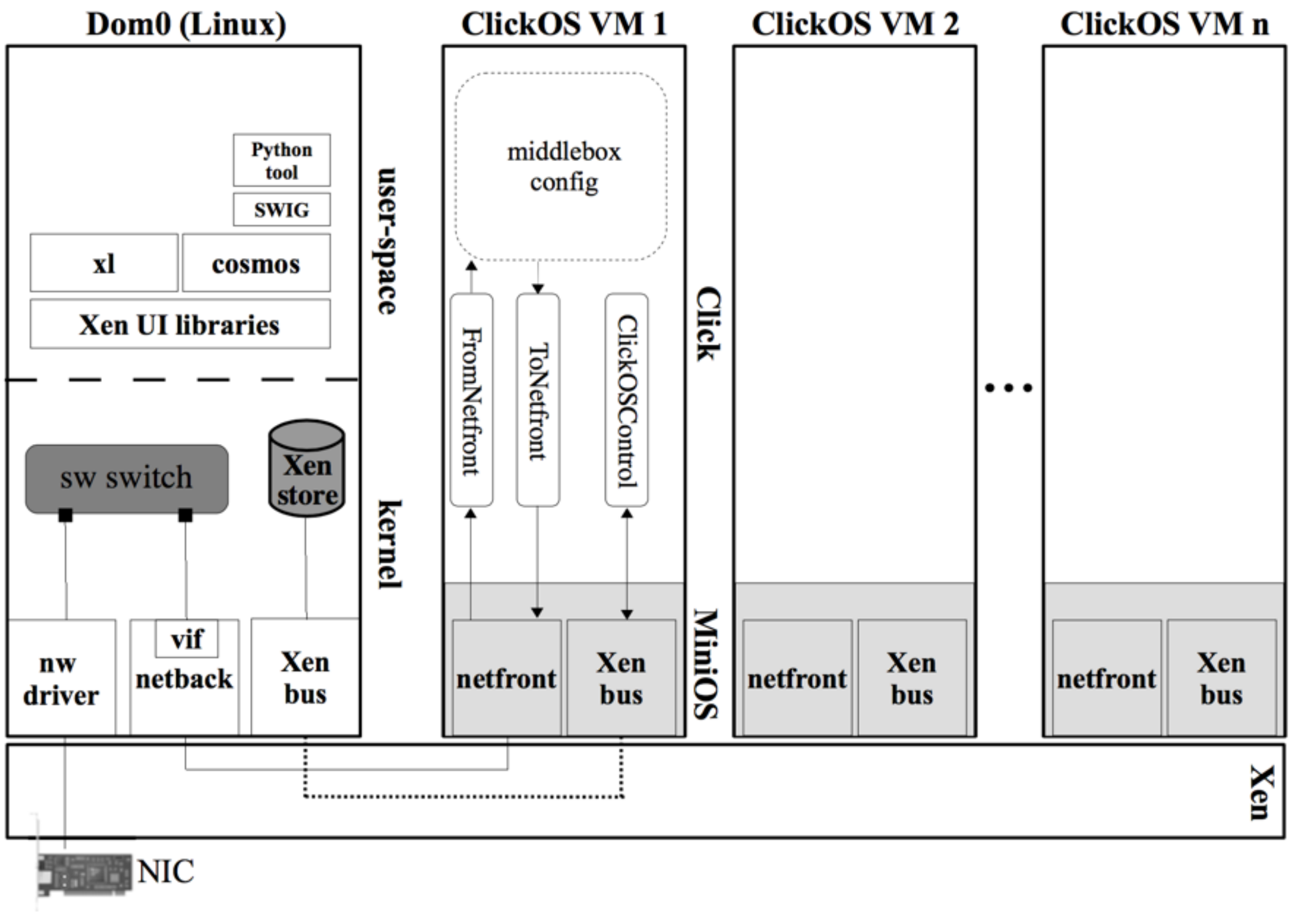
Middlebox	Functionality	Examples
NIDS	Monitor network activities for malicious behaviors or policy violations and report to operations staff	Bro IDS, Snort
Firewall	A firewall based on packet filter drops packets matching specific patterns and forwards the rest to next middlebox	iptables, pf
Load Balancer	Load balancers distribute network flows across several down- stream middleboxes	HAProxy, LVS
NAT	Remap one IP address space to another	iptables

Challenges: Consolidation

- Process
  - No performance penalty
  - No performance isolation
- Virtual machine
  - Virtualization overhead
  - Performance isolation
  - Easy deployment
  - Mature technology
- Container
  - No performance penalty
  - Performance isolation
  - Easy deployment

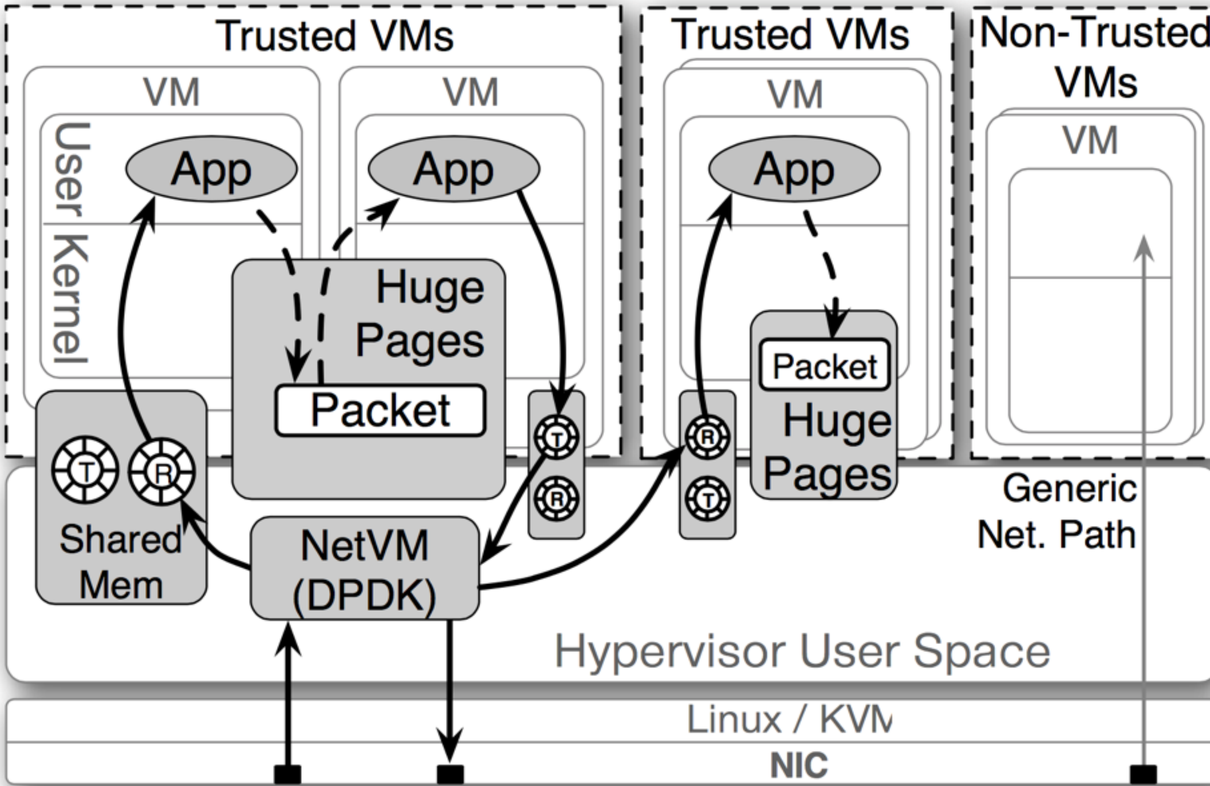
Challenges: Consolidation

- Virtual machine ClickOS (NSDI'14)



Challenges: Consolidation

- Virtual machine NetVM (NSDI'14)

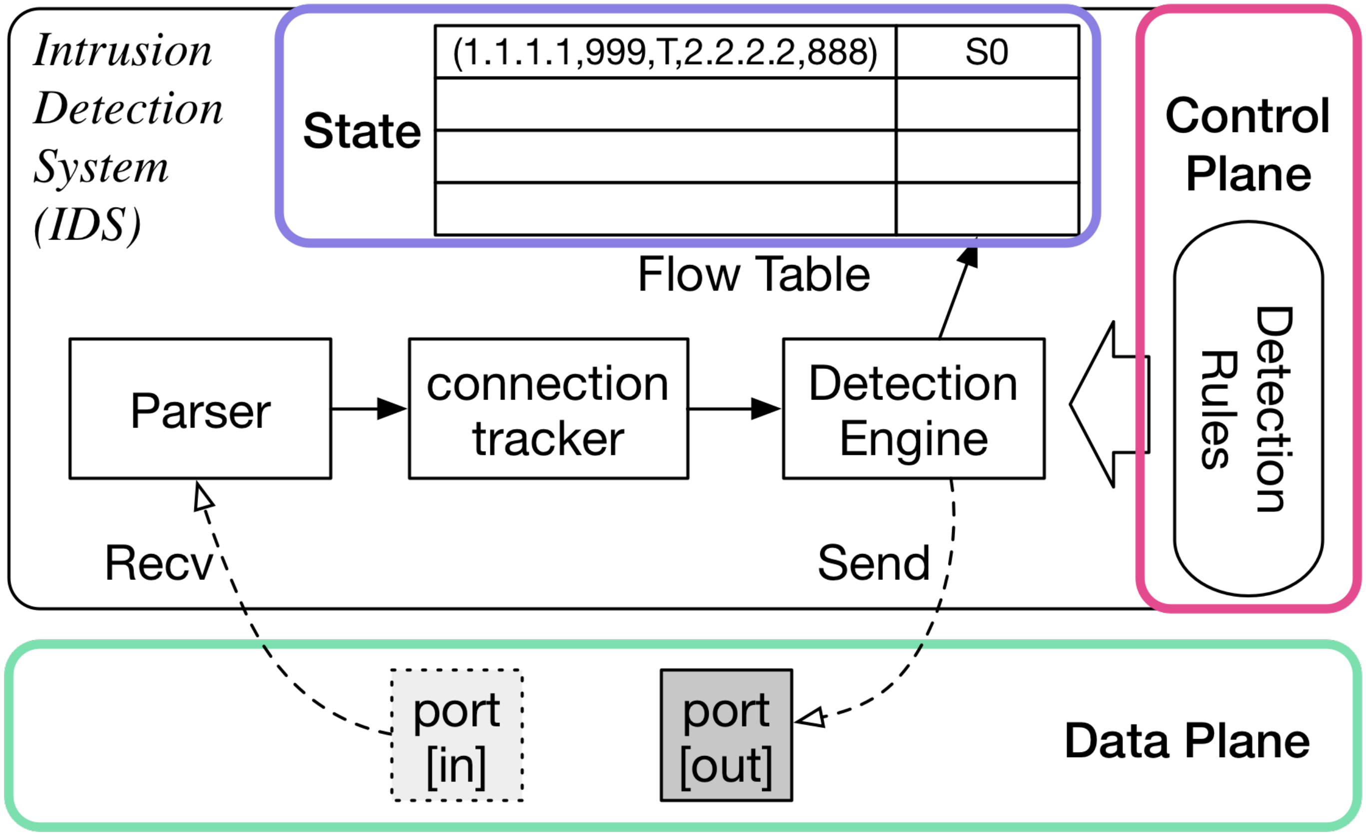


Challenges: Scalability

- Dynamic network
  - Workload
  - Reconfiguration
  - Resource allocation

- Scalability
  - Scale up
    - Multicore hardware
    - NUMA aware
  - Scale out
    - Replication
    - Migration

Challenges: State Management



Challenges: State Management

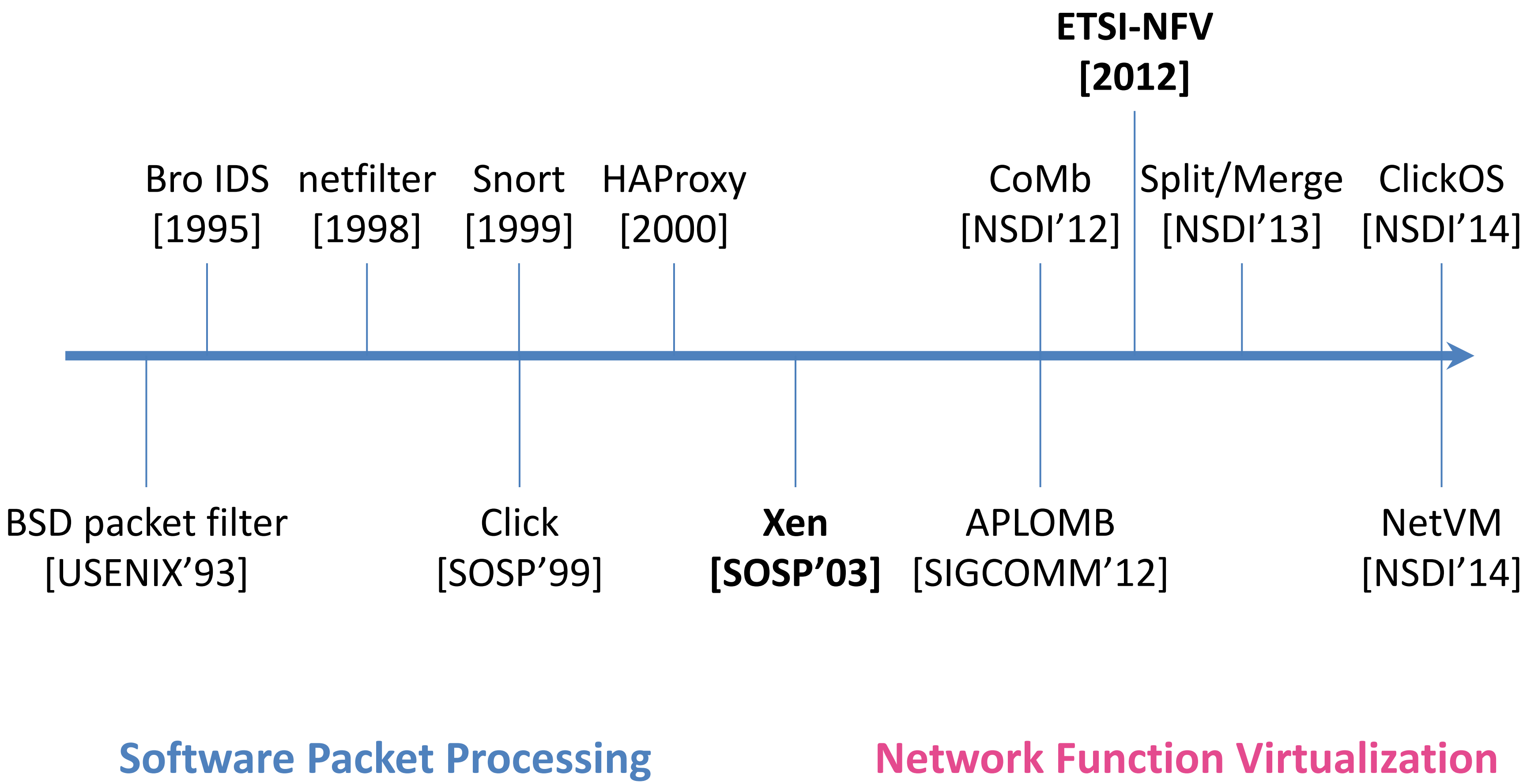
- State consistency
  - Distributed instances
  - Recovery and migration

- Requirements
  - Classified states: internal & external
  - Transactional boundary
  - Move states between replicas
  - Correctly route traffic

NFV: History

- Software middlebox
  - IDS: Bro, Snort
  - Firewall: iptables (1998), BSD pf (USENIX'93)
  - Load Balancer: HAProxy, LVS
  - NAT: iptables
- Click (SOSP'99)
- Middlebox consolidation
  - NIDS Cluster (RAID'07)
  - APLOMB (SIGCOMM'12)
  - CoMb (NSDI'12)
  - ClickOS (NSDI'14)

NFV: History



ETSI-NFV
[2012]
Bro IDS
[1995]
netfilter
[1998]
Snort
[1999]
HAProxy
[2000]
CoMb
[NSDI'12]
Split/Merge
[NSDI'13]
ClickOS
[NSDI'14]
BSD packet filter
[USENIX'93]
Click
[SOSP'99]
Xen
[SOSP'03]
APLOMB
[SIGCOMM'12]
NetVM
[NSDI'14]
Software Packet Processing
Network Function Virtualization

NFV: SUMMARY

NFV: Summary

NFV: Summary

- Virtualization of network appliances
- Performance and flexibility
- System design
  - Multicore and NUMA hardware
  - Distributed processing
  - I/O optimization

The Click Modular Router

Robert Morris, Eddie Kohler, John Jannotti, M. Frans Kaashoek

Background

- New network functions
  - NIDS
  - Random early detection dropping policy
  - Deep packet inspection
  - GFW
- Commercial off-the-shelf (COTS) hardware
  - Expensive
  - Closed
  - Static
  - Inflexible

Motivation

- Flexible
  - New functionality
- Modular
  - Combine elements
- Open
  - Allow developers to add elements
- Efficient
  - Comparable with hardware

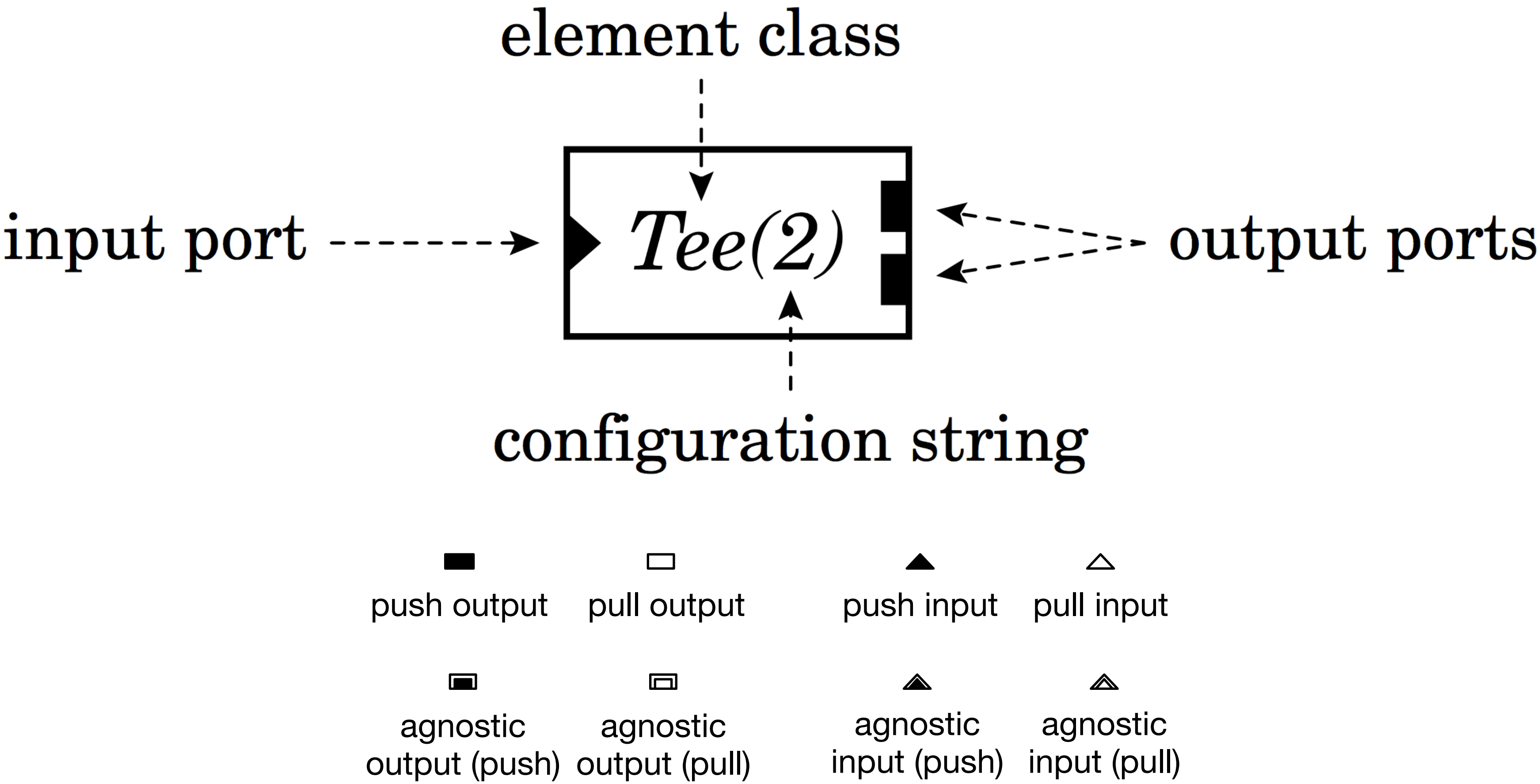
Click: Overview

- A modular design to allow easy construction of routers and other middleboxes
  - Elements
    - Push
    - Pull
  - A domain-specific language (DSL)
    - Declare elements
    - Connect elements
  - Lightweight framework
    - Orchestrate elements into a router

Elements

- Building block of Click router
  - C++ classes
  - Scheduled by framework
- Ports
  - Endpoints of connections
  - Associated with push or pull function (or *agnostic*)
- Configuration
  - Parameters to initialize elements
  - Elements specific

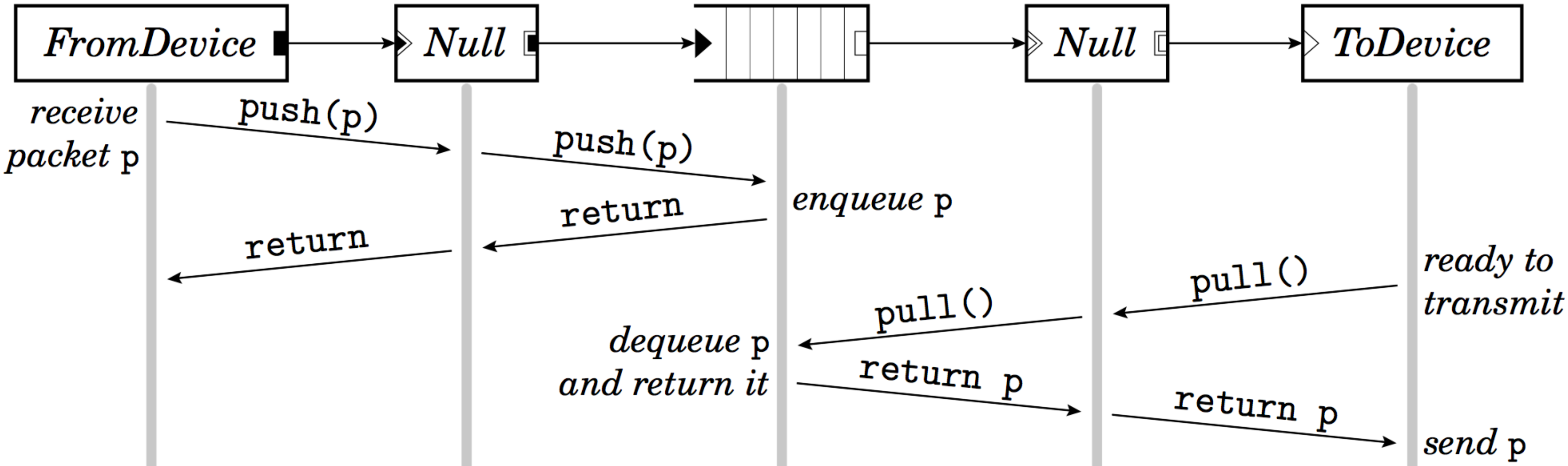
Element Example



C++ Class of Elements

```
class NullElement : public Element {
    public:
    NullElement()
        { add_input(); add_output(); }
    const char *class_name() const
        { return "Null"; }
    PushOrPull default_processing() const
        { return AGNOSTIC; }
    NullElement *clone() const
        { return new NullElement; }
    void push(int port_number, Packet *p)
        { output(0).push(p); }
    Packet *pull(int port_number)
        { return input(0).pull(); }
};
```

Push and Pull

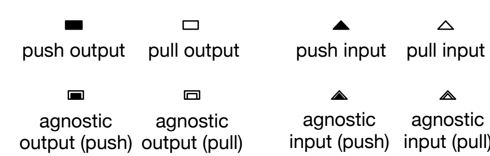


Push

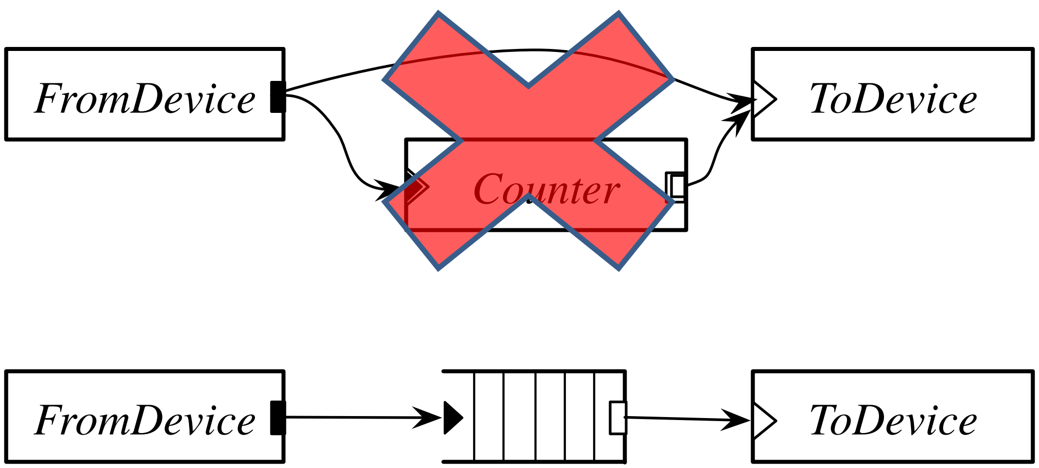
- From source element to downstream
- Triggered by event, e.g. packet arrival
- Denoted by solid square or triangle

Pull

- From destination element to upstream
- Packet transmission
- Denoted by empty square or triangle



More Examples



Packet Counter


```
// Declare three elements ...
src :: FromDevice(eth0);
ctr :: Counter;
sink :: Discard;
// ...and connect them together
src -> ctr;
ctr -> sink;
```

Ping Responder

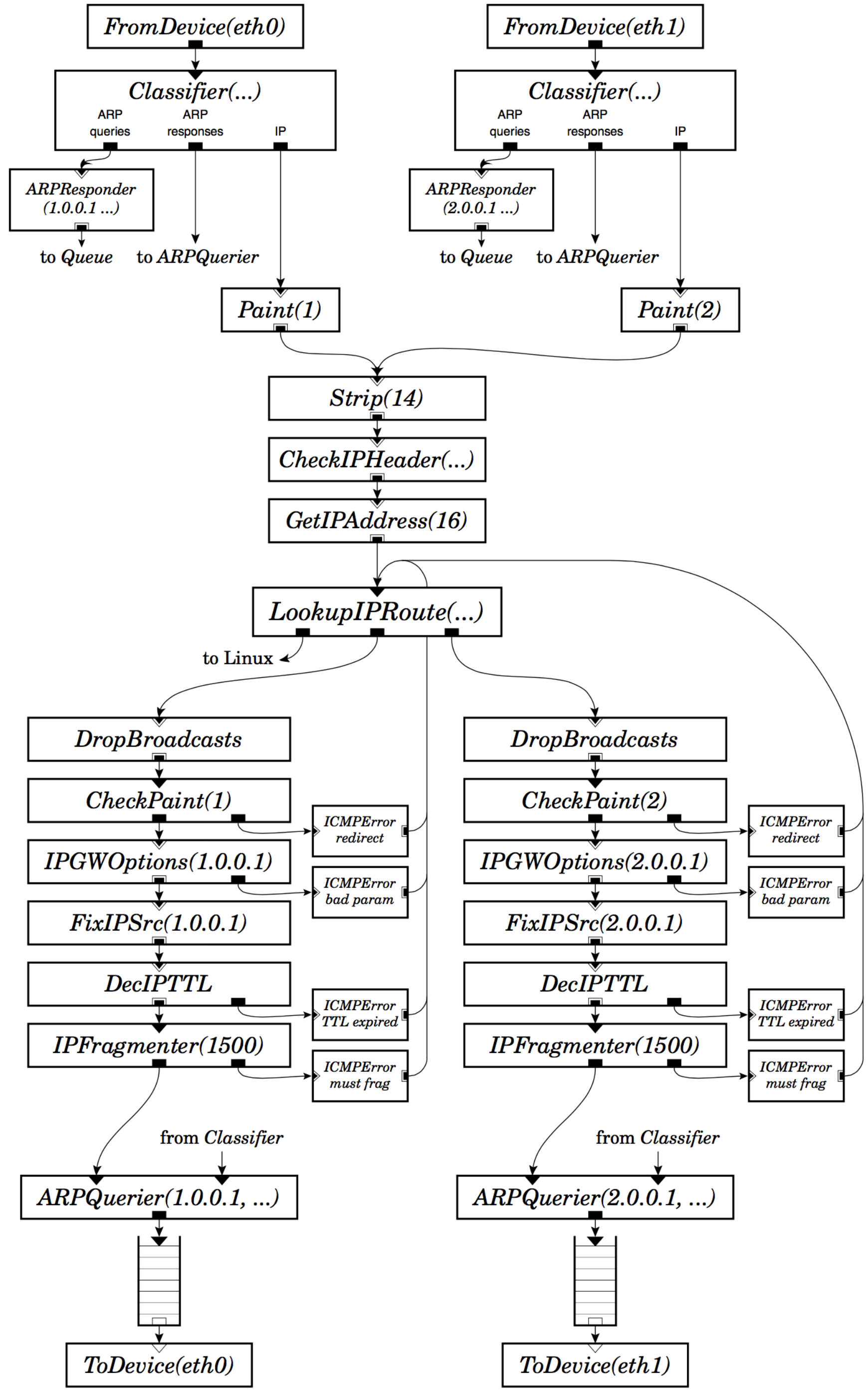
```
define($IP 192.168.13.235);
define($MAC 00-15-17-15-5d-75);

tap  :: KernelTap(192.168.13.235/16, ETHER $MAC);

c :: Classifier(
        12/0806 20/0001, // ARP Requests goes to output 0
        12/0806 20/0002, // ARP Replies to ouput 1
        12/0800, // ICMP Requests to output 2
        -);

arpq :: ARPQuerier($IP, $MAC);
arpr :: ARPResponder($IP $MAC);

tap-> c;
c[0] -> ARPPrint -> arpr -> Print("arpr", 60) -> tap;
c[1] -> [1]arpq;
Idle -> [0]arpq;
arpq -> Print("arpq", 60) -> ARPPrint -> tap;
c[2] -> CheckIPHeader(14) -> ICMPPingResponder() -> EtherMirror() -> tap;
c[3] -> Discard;
```



FromDevice(eth0)
FromDevice(eth1)
Classifier(...)
ARP queries
ARP responses
IP
Classifier(...)
ARP queries
ARP responses
IP
ARPResponder (1.0.0.1 ...)
ARPResponder (2.0.0.1 ...)
to Queue
to ARPQuerier
to Queue
to ARPQuerier
Paint(1)
Paint(2)
Strip(14)
CheckIPHeader(...)
GetIPAddress(16)
LookupIPRoute(...)
to Linux
DropBroadcasts
CheckPaint(1)
IPGWOptions(1.0.0.1)
FixIPSrc(1.0.0.1)
DecIPTTL
IPFragmenter(1500)
ICMPError redirect
ICMPError bad param
ICMPError TTL expired
ICMPError must frag
DropBroadcasts
CheckPaint(2)
IPGWOptions(2.0.0.1)
FixIPSrc(2.0.0.1)
DecIPTTL
IPFragmenter(1500)
ICMPError redirect
ICMPError bad param
ICMPError TTL expired
ICMPError must frag
from Classifier
ARPQuerier(1.0.0.1, ...)
from Classifier
ARPQuerier(2.0.0.1, ...)
ToDevice(eth0)
ToDevice(eth1)

Scheduling

- Single thread in the original paper
- Only some elements need to be scheduled
- Click schedules the CPU with a task queue
  - Flexible and lightweight stride scheduling [Waldspurger'95]
  - Element is the unit of CPU scheduling
  - Most elements are never placed on the task queue
  - Elements frequently initiates push or pull should be scheduled (`FromDevice`, `ToDevice`)

Implementation

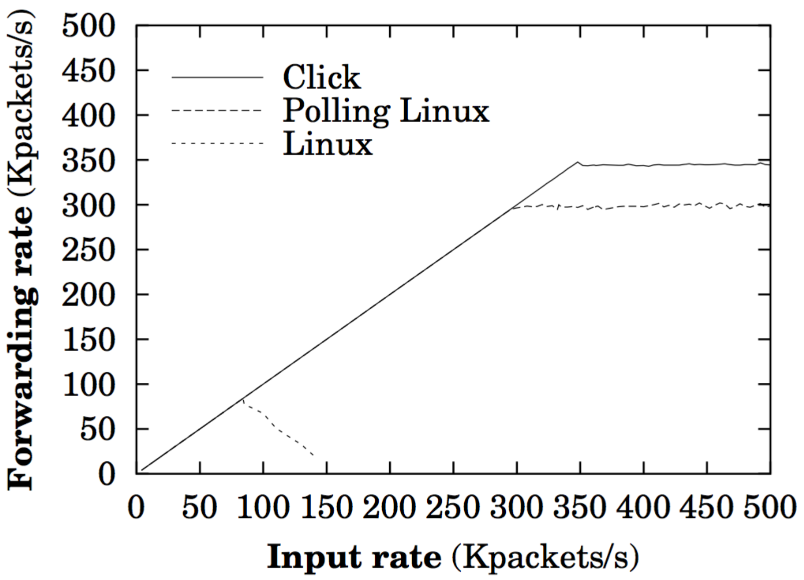
- Userlevel
  - Run as normal thread
  - `FromDevice`: TUN/TAP, `libpcap`
- Kernel module
  - Run as kernel thread
  - Loops over the task queue and run each task
  - Only interrupts can preempt this thread
  - Voluntarily gives up CPU to Linux
- ClickOS [NSDI'14]
  - Run in MiniOS on Xen
  - Use netmap [ATC'12] to get packets

Maximum Loss-Free Forwarding Rate

- Click IP router
  - 8 interfaces
  - 161 elements
  - 64 bytes packets size

- Forwarding rate versus input rate

- MLFFR reflects the router's behavior under load



Adding Network Functions

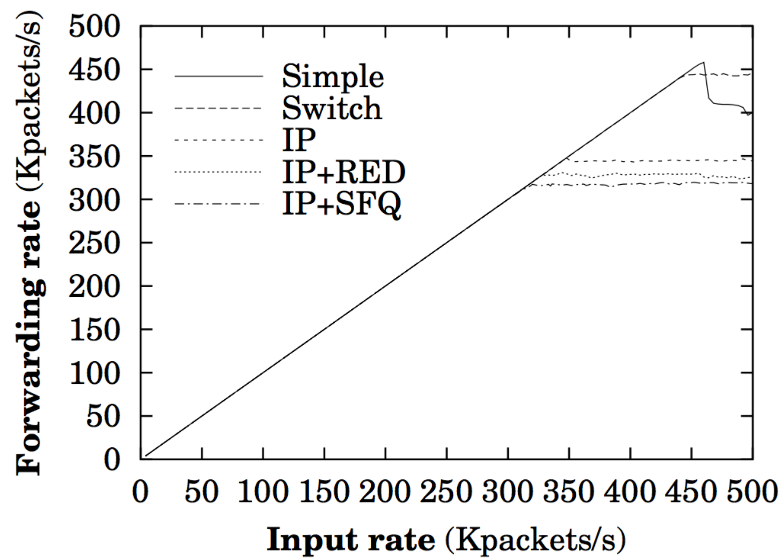
Simple
- Passes packets from one interface to another

Switch
- L2 switch

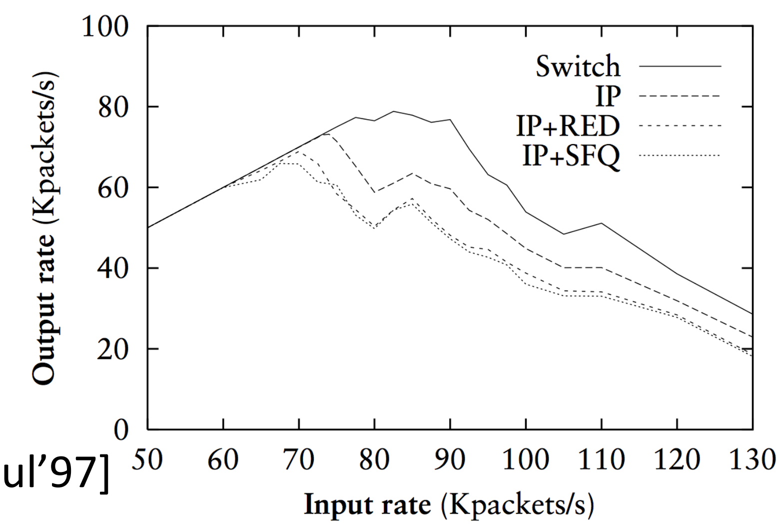
IP
- IP router

IP+RED
- Router + random early detection dropping policy

IP+SFQ
- Router + stochastic



Polling



Linux IRQ

Livelock! [Mogul'97]

Summary

- NFV: everything old is new again
- Cloud networks need
  - Flexible operational model
  - State-of-the-art infrastructure services
  - Programmable network functions
- NFV is changing the networking industry