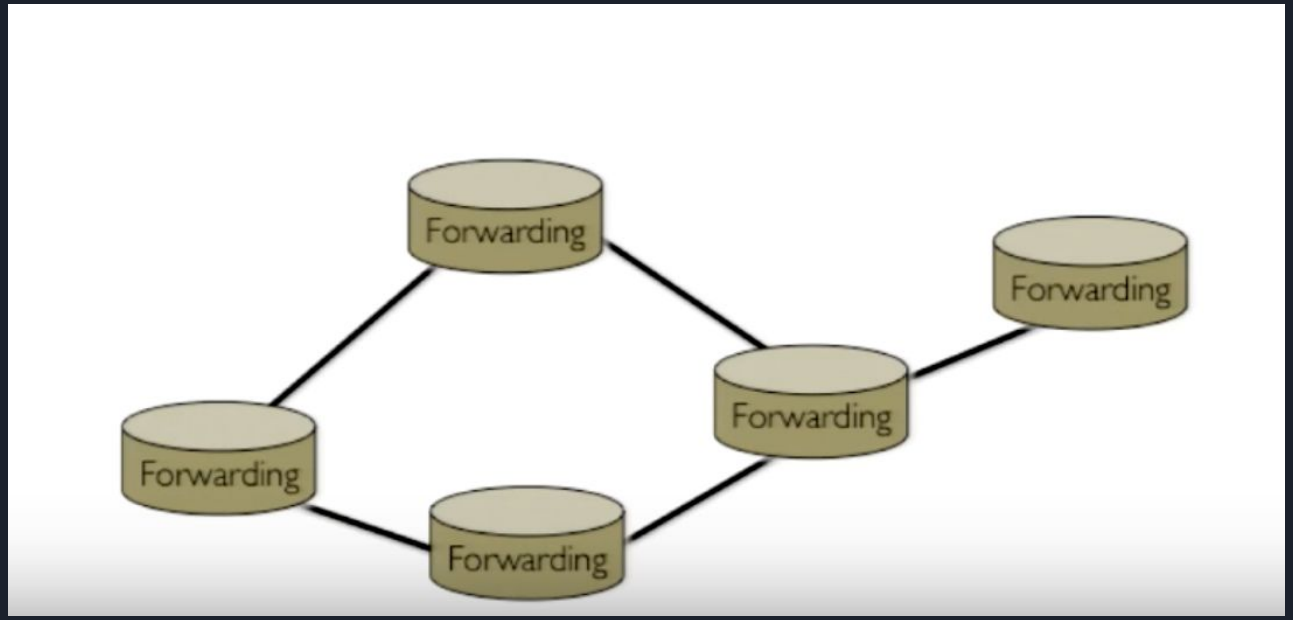
E2 A Framework for NFV applications

Presented by Paruyr Muradyan

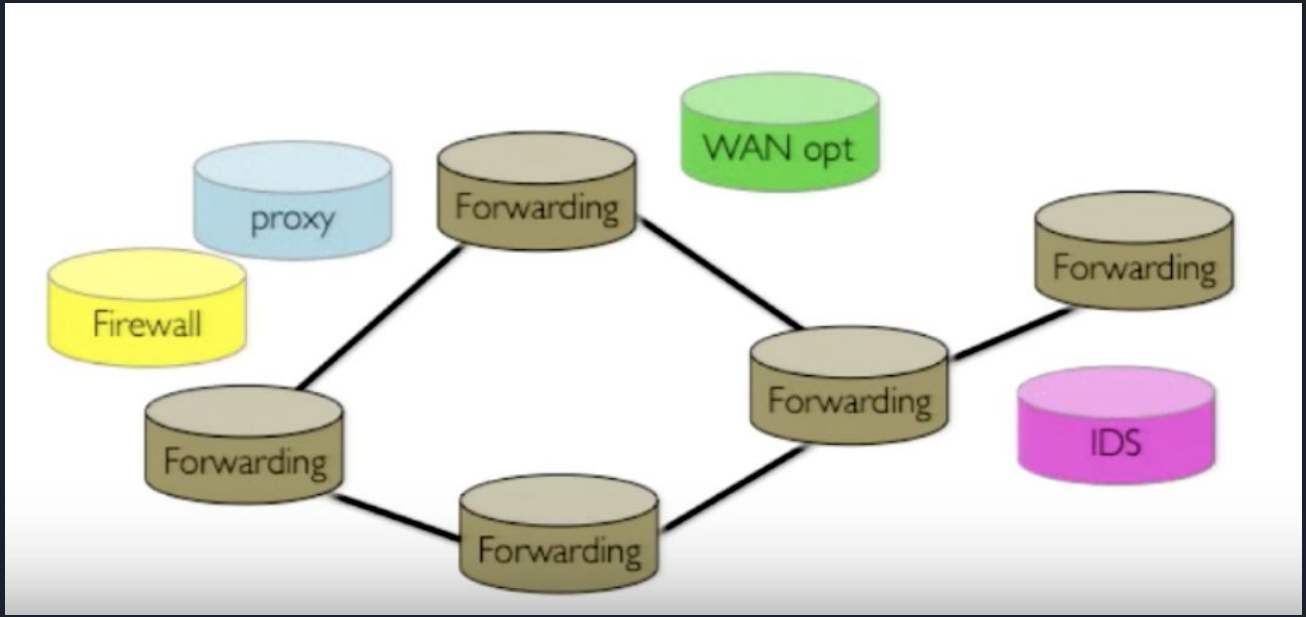
What network looks like

Switches and routers interconnected by links, that forward packets.



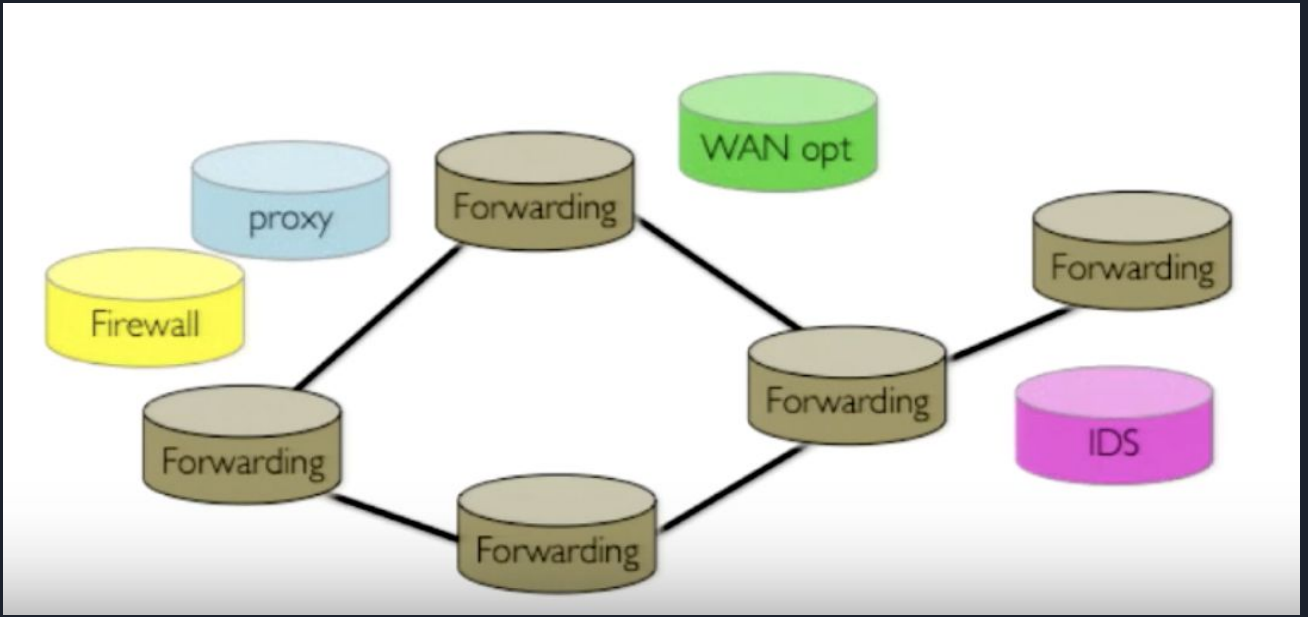
What network looks like

Middleboxes: There are a lot of specialized devices like firewalls, proxies, intrusion detection systems or WAN optimizers.



What network looks like

In many network solutions you have as many middleboxes as switches and routers.



Problems with middleboxes

Dedicated: impossible to shift unused resources from one to the other

Fix-function: there is no or very little programmability

Specialized hardware / software

Custom management APIs

High management costs.

Alternative approach

Middleboxes as software applications deployed on servers.

Alternative approach

Middleboxes as software applications deployed on servers.

Benefits

- Cost benefits
- Efficiency of statistical multiplexing
- Easier evolution and deployment

Alternative approach

Middleboxes as software applications deployed on servers.

Benefits

- Cost benefits
- Efficiency of statistical multiplexing
- Easier evolution and deployment

This approach is called Network Function Virtualization (NFV)

The Benefits of Network Functions Virtualization

reducing the need to purchase purpose-built hardware and supporting pay-as-you-grow models

The Benefits of Network Functions Virtualization

reducing the need to purchase purpose-built hardware and supporting pay-as-you-grow models

reducing space, power and cooling requirements of equipment and simplifying the roll out and management of network services

The Benefits of Network Functions Virtualization

reducing the need to purchase purpose-built hardware and supporting pay-as-you-grow models

reducing space, power and cooling requirements of equipment and simplifying the roll out and management of network services

reducing the time to deploy new networking services

The Benefits of Network Functions Virtualization

reducing the need to purchase purpose-built hardware and supporting pay-as-you-grow models

reducing space, power and cooling requirements of equipment and simplifying the roll out and management of network services

reducing the time to deploy new networking services

quickly scale up or down services to address changing demands

NFV: A closer look

Monolithic hardware -> Monolithic software

Custom per-app solutions to scaling, failover, etc

Custom per-app management API

No end-to-end system architecture

Inspiration of Elastic Edge (E2)

Modern data analytics systems (Hadoop, Spark, etc.)

Framework coordinates end to end execution of job and takes care of routine functions

E2 is a framework for NFV

- End to end management of network functions
- Provide general solutions for common tasks.

Benefits

- Simpler development of NFs
- Automated management of operators

What is E2 ?

A software environment for packet processing applications that implements general techniques for common issues.

What is E2 ?

A software environment for packet processing applications that implements general techniques for common issues.

- Placement, (which NF runs where)

What is E2 ?

A software environment for packet processing applications that implements general techniques for common issues.

- Placement, (which NF runs where)
- Elastic scaling (adapting number of NF instances and balancing load across them)

What is E2 ?

A software environment for packet processing applications that implements general techniques for common issues.

- Placement, (which NF runs where)
- Elastic scaling (adapting number of NF instances and balancing load across them)
- Fault tolerance

What is E2 ?

A software environment for packet processing applications that implements general techniques for common issues.

- Placement, (which NF runs where)
- Elastic scaling (adapting number of NF instances and balancing load across them)
- Fault tolerance
- Energy management
- Monitoring

E2 Interface

E2 provides a declarative interface for SDN (Software defined network) controller to tell each E2 cluster how traffic should be processed.

E2 Interface

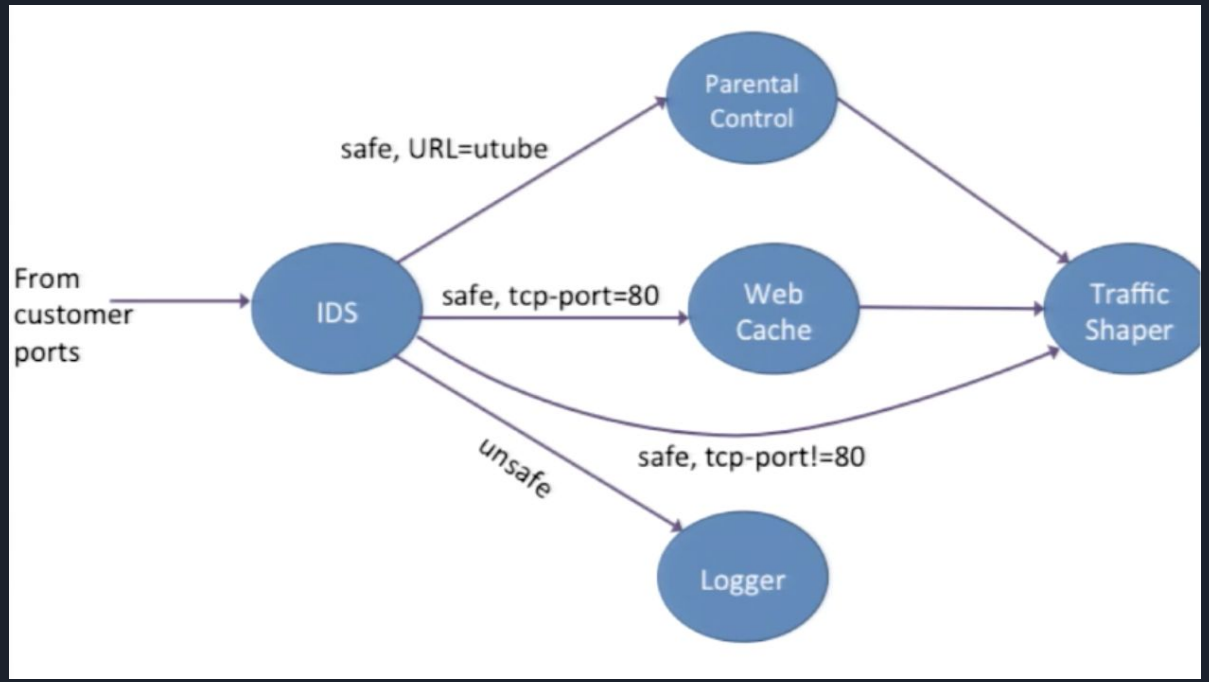
E2 provides a declarative interface for SDN (Software defined network) controller to tell each E2 cluster how traffic should be processed.

It does so by using policy statements called pipelets.

Pipelets use directed acyclic graphs (DAG) that describe traffic processing rout.

DAG

DAG is composed of nodes (NFs) and edges that define what type of traffic should be routed to a specific NF.



DAG

Pipelet: a traffic class -> policy DAG statement

- traffic class: defined on packet header and ports
- Policy DAG: defines how traffic is process by NFs
  - Nodes are NFs
  - Edges describe type of traffic by traffic classes
  - Optional: estimate of the traffic load on an edge for better placement and management of NF

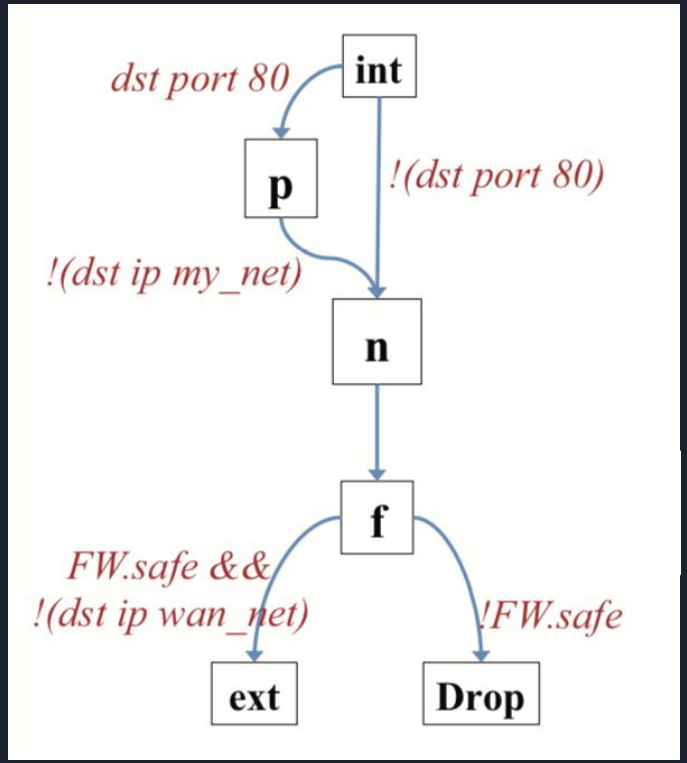
A Pipelet example

```
// First, instantiate NFs from application types.
Proxy p;
NAT   n;
FW    f;
Port<0-7>  int;  // internal customer-facing ports
Port<8-15> ext;  // external WAN-facing ports
```

A Pipelet example

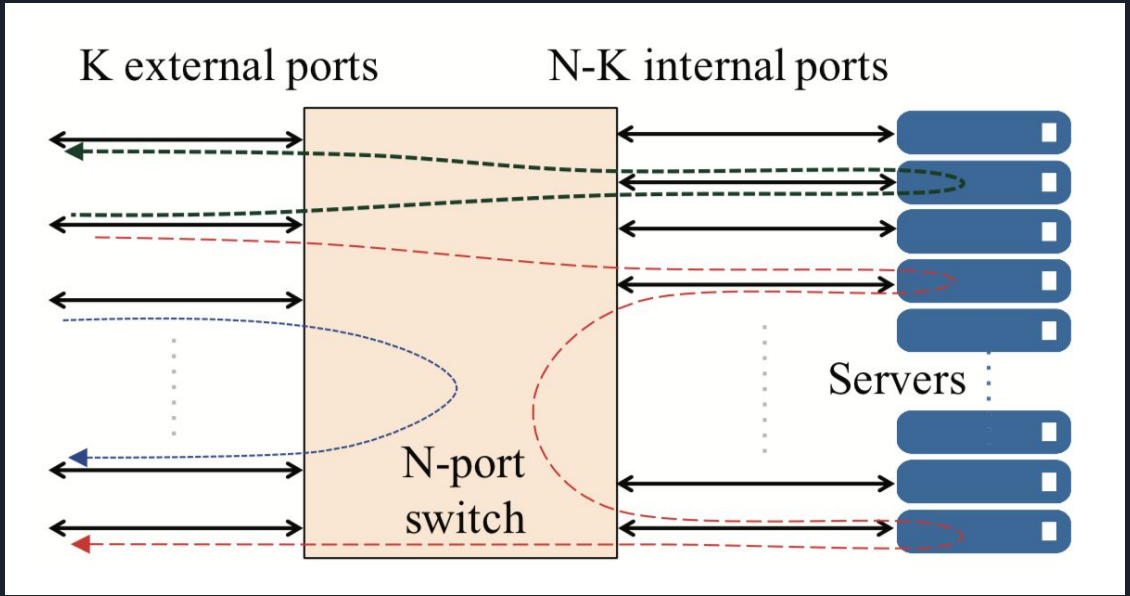
```
pipelet {  // outbound traffic
  int [dst port 80] -> p;
  int [!(dst port 80)] -> n;
  p [!(dst ip my_net)] -> n;
  n -> f;
  f [FW.safe && !(dst ip wan_net)] -> ext;
  f [!FW.safe] -> Drop;
}
```

A Pipelet example

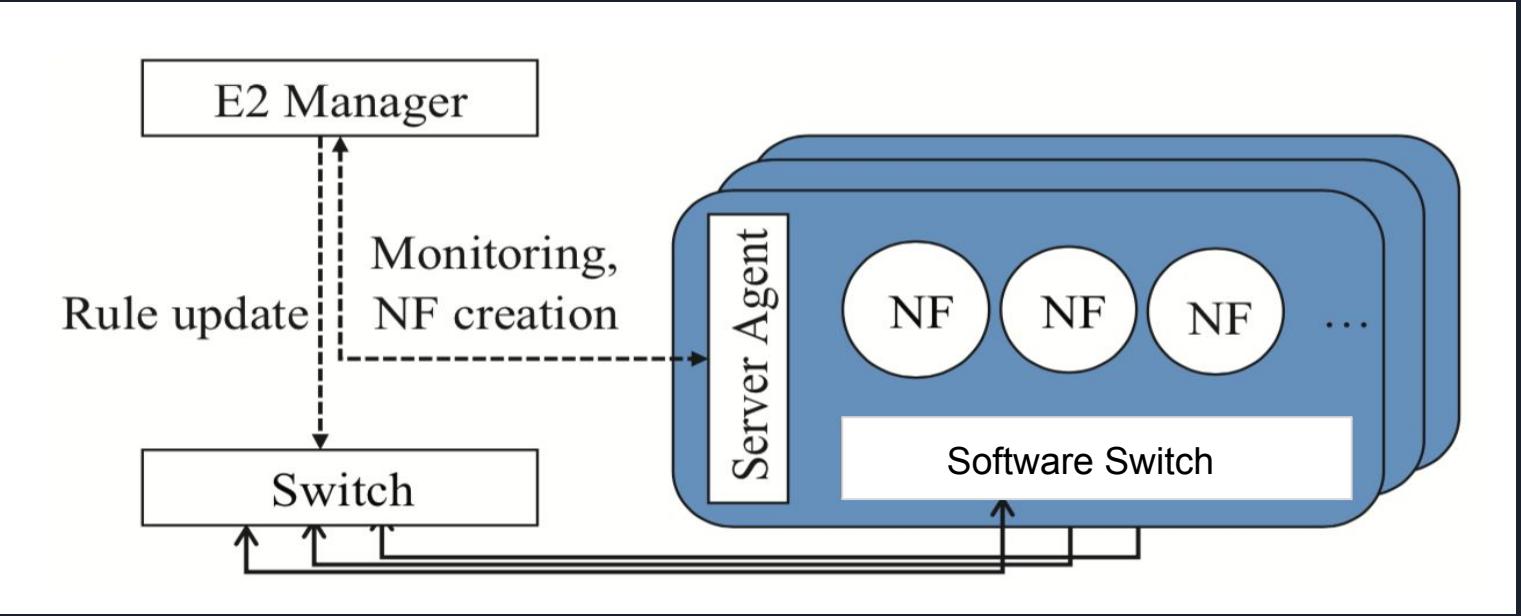


Hardware infrastructure

E2 is designed for general-purpose servers interconnected by commodity switches.

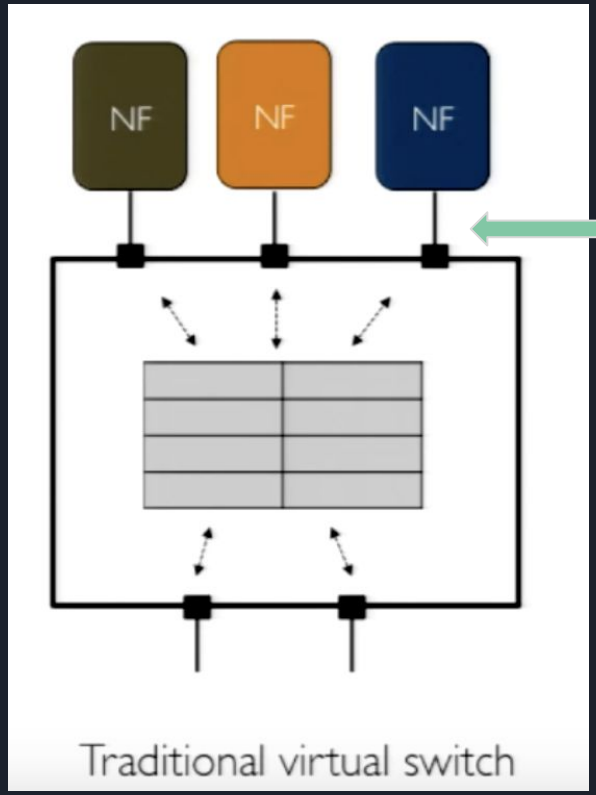


Hardware infrastructure



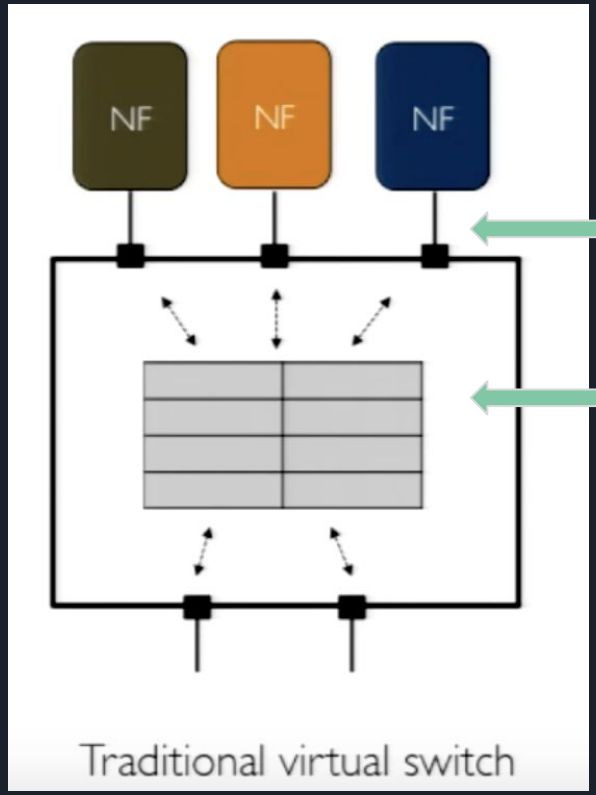
E2 Manager takes as an input policy pipelet and configures all components of the system.

Software switch architecture



Virtual ports (vports): NF connects to switch

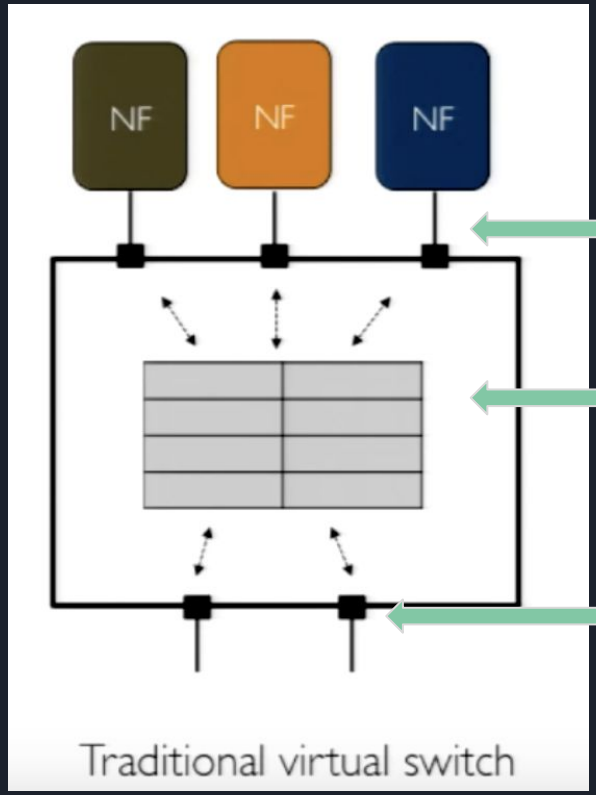
Software switch architecture



Virtual ports (vports): NF connects to switch

Flow table: match actions rules

Software switch architecture



Virtual ports (vports): NF connects to switch

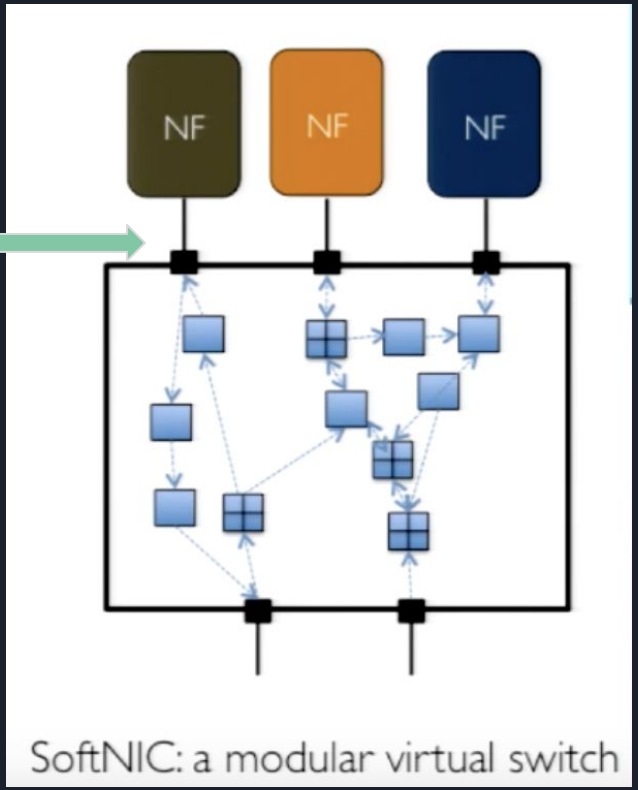
Flow table: match actions rules

Physical ports (pports)

Software switch architecture based on SoftNIC

vports
bytestream vports

per-packet metadata

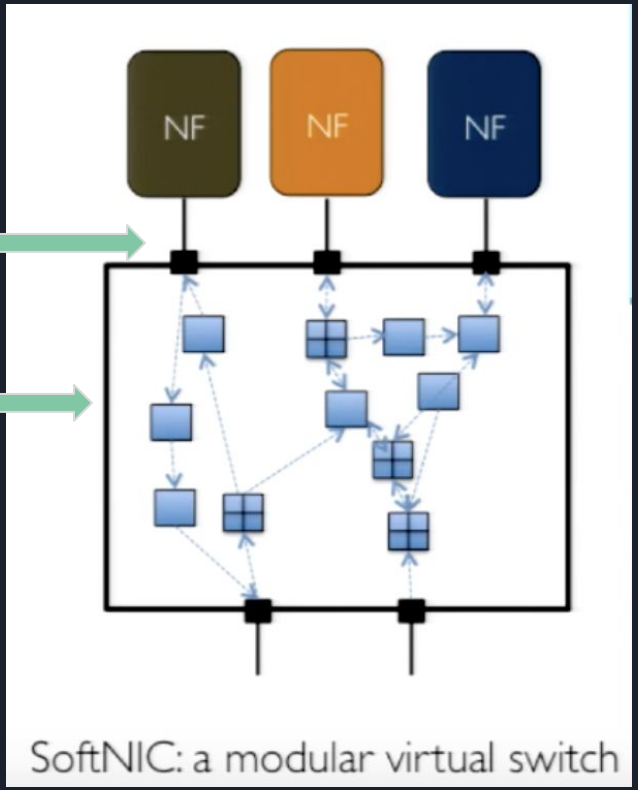


Software switch architecture based on SoftNIC

vports
bytestream vports

per-packet metadata

E2 API



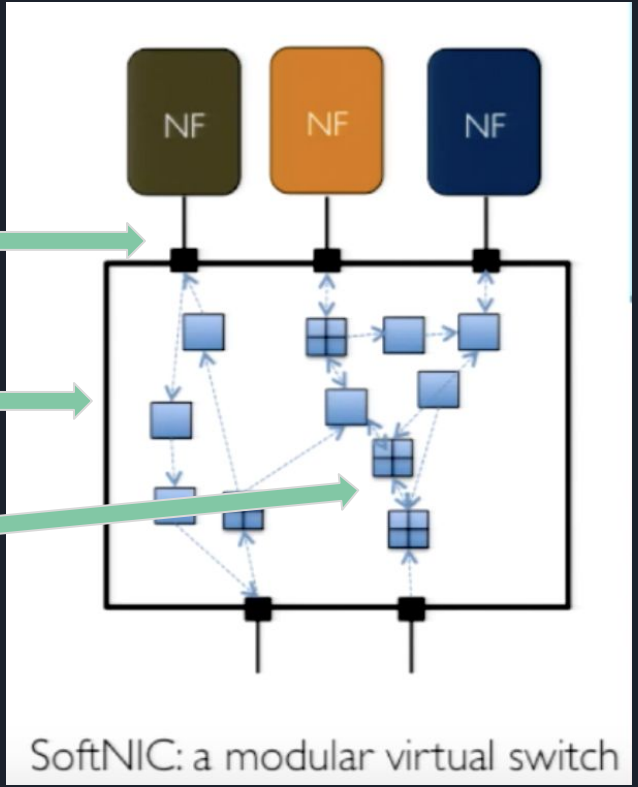
Software switch architecture based on SoftNIC

vports
bytestream vports

per-packet metadata

E2 API

E2 modules for dynamic scaling, composition, etc.



E2 operates on three inputs

Policy pipelets

Network Function description

Hardware description

E2 operates on three inputs

Policy pipelets

Network Function description

- Attributes: which attribute an NF exports and how
- (Attribute: Value) -> method binding
- Methods: vports (packet or bytestream based) and metadata
  - (is_safe, True) -> vport 4
  - (URL, *) -> metadata 2
- Performance information (optional)

E2 operates on three inputs

Policy pipelets

Network Function description

Hardware description

- Capabilities of hardware
- Number of cores, bandwidth, number of ports ...

E2 operations

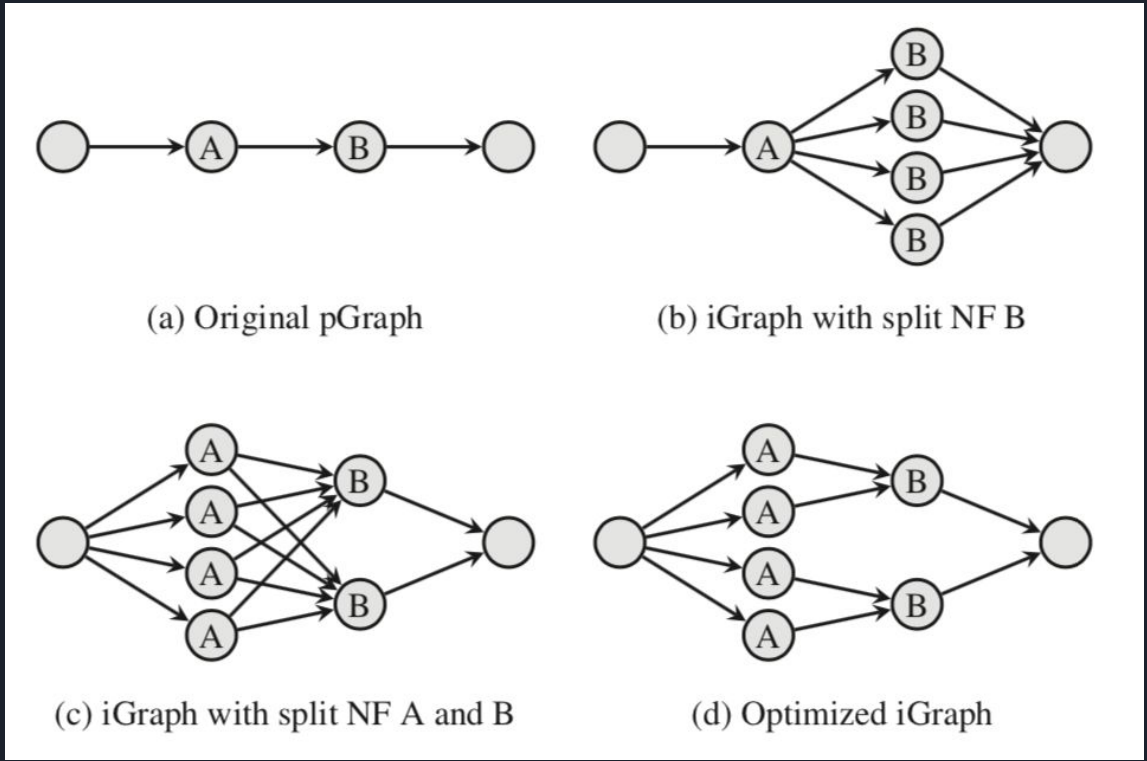
Policy pipelets

Network Function description

Hardware description

E2 operations

The process of Sizing



E2 operations

Policy pipelets

Network Function description

Hardware description

Goal: Minimize switch traffic
A graph partitioning problem: E2 uses an algorithm based on the Kernighan-Lin heuristic
For n nodes in instance graph

- Initial placement: $O(n^2 \log n)$
- Incremental placement: $O(n)$

E2 operations

Policy pipelets

Network Function description

Hardware description

Direct traffic between the NFs

Composition

What traffic should be sent to NF: vports vs metadata vs header matching

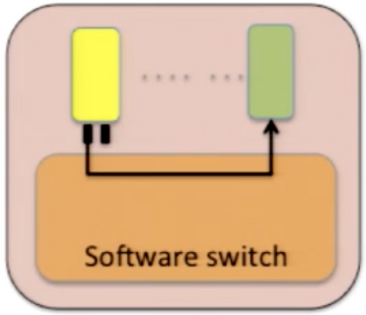
How traffic is sent: packet vs bytestream

Which NF instance is traffic sent to: load balancing

Performance Overview

Overheads E2 adds

The metadata does not come for free



- Intel E5-2680
- SoftNIC runs on one core

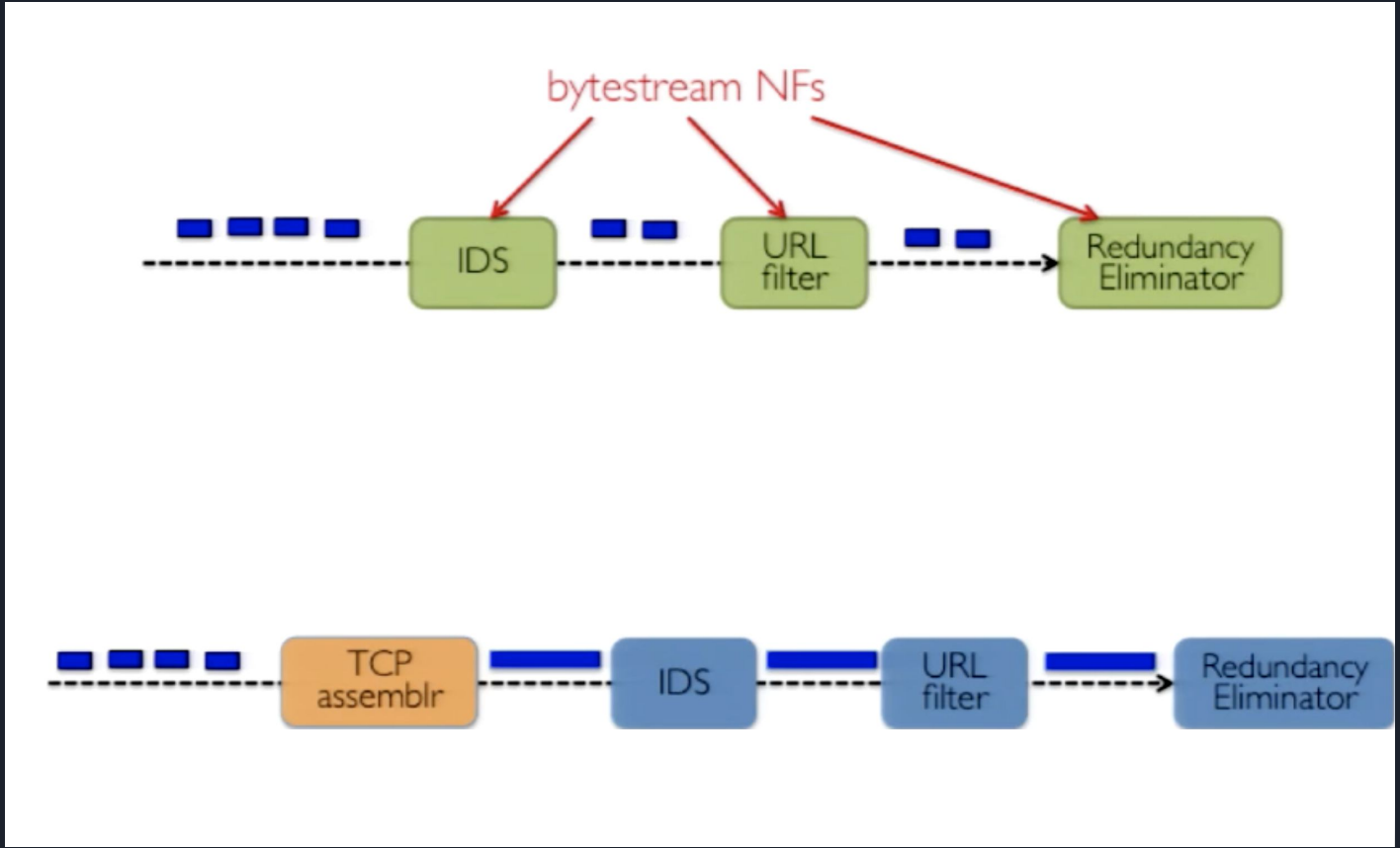
Path NF→ softnic→ NF	Latency (µs)	Gbps (1500B)	Mpps (64B)	
Baseline SoftNIC (vport→vport)	1.214	187.5	15.24	
Header-Match	1.56	152.5	12.76	↓16%
Metadata-Match	1.695	145.826	11.96	↓6%

Overheads E2 adds

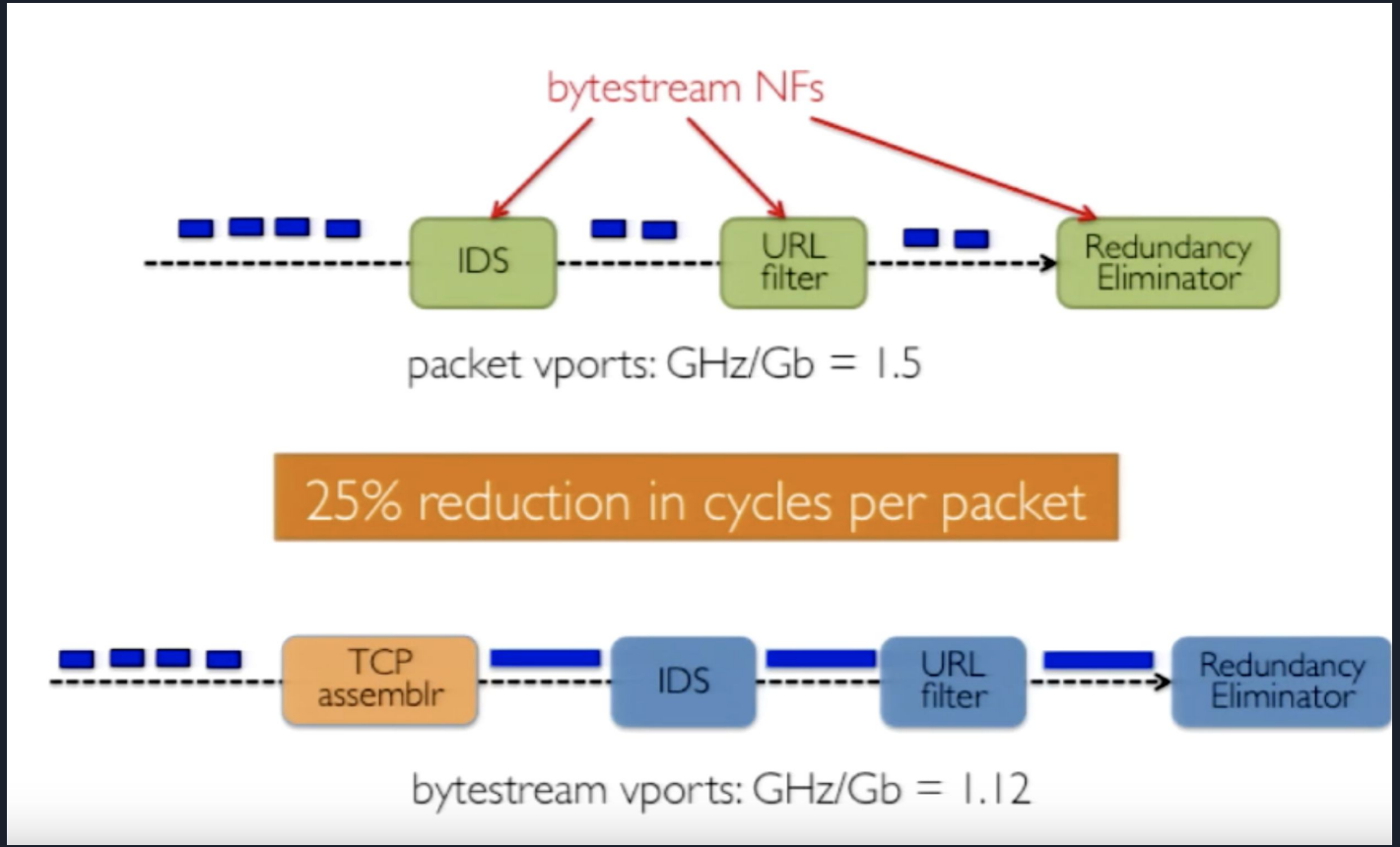
The metadata does not come for free

- It reduces CPU consumption per packet by 41%

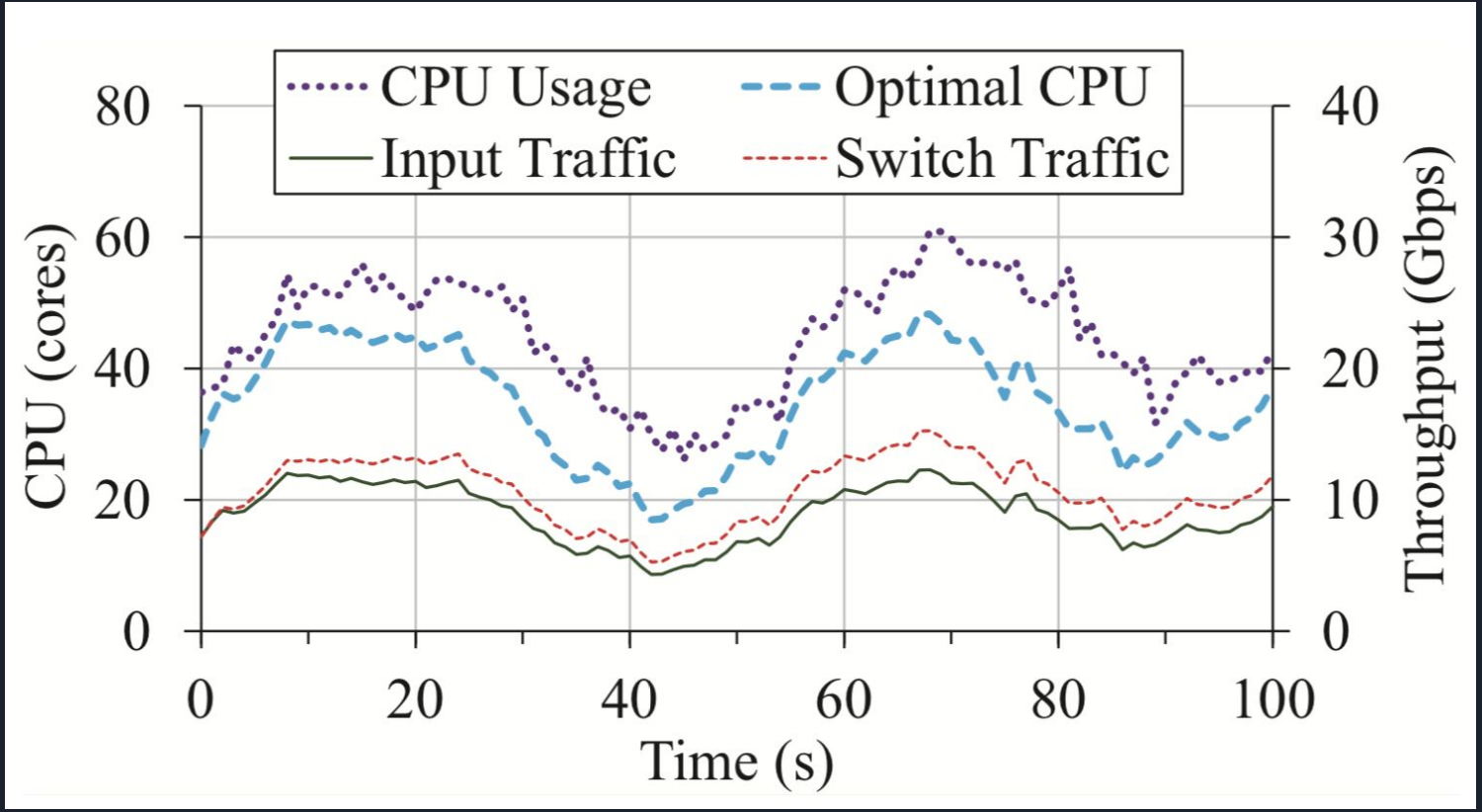
Bytestream Virtual Ports



Bytestream NFs



End to End Operation: CPU and Traffic



Conclusion

E2 is a runtime framework for NFV applications

Interface: traffic-class -> NF pipeline

Separation of concerns

- NFs implement application-specific processing
- Framework implements common functions
- Framework manages cluster's internal network

Simplifies NF development and management

The End.