Platform for programmable heterogeneous virtual middleboxes

The team

Laurent Mathy (Supervisor)
Cyril Soldani (Language / code analysis / private NFV)
Emmanouil Psanis (Synchronization)
Me, Tom Barbette (System / Packet Processing)

We develop a platform for programmable heterogeneous virtual middleboxes

Programmable

Stir network innovation

Deploy middleboxes in the Cloud

We develop a platform for programmable heterogeneous virtual middleboxes

Platform

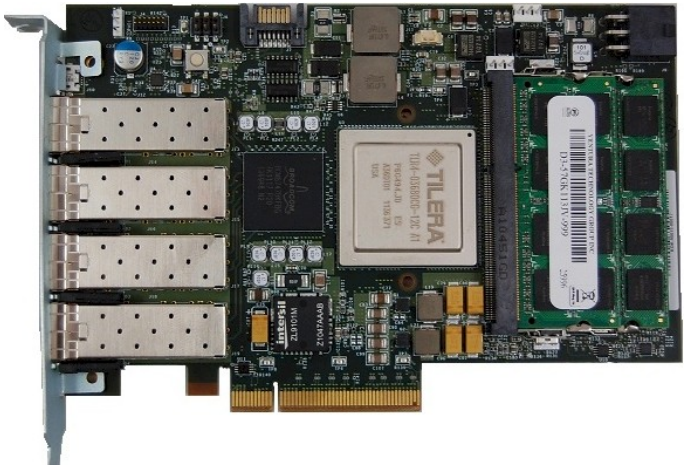
It needs to be

flexible (for various current and future middleboxes)
secure (isolation between tenants w.r.t data *and* resources)
fast (low latency, high throughput)
easy to use

We develop a platform for programmable heterogeneous virtual middleboxes

Heterogeneous

GPU



Tilera

NetFPGA

CPU

We develop a platform for programmable heterogeneous virtual middleboxes

Virtual

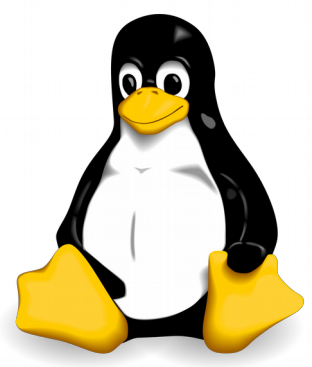
Consolidate middleboxes

- Full-VM not the right abstraction for performance
- Fine grained (eg. Tailtrie : multiple FIB on GPU with partial similar data)

Migrate them between the underlying hardware

- Virtualize data structures for heterogeneous hardware (Tile, NetFPGA, GPU, x86)

What will be the platform ?

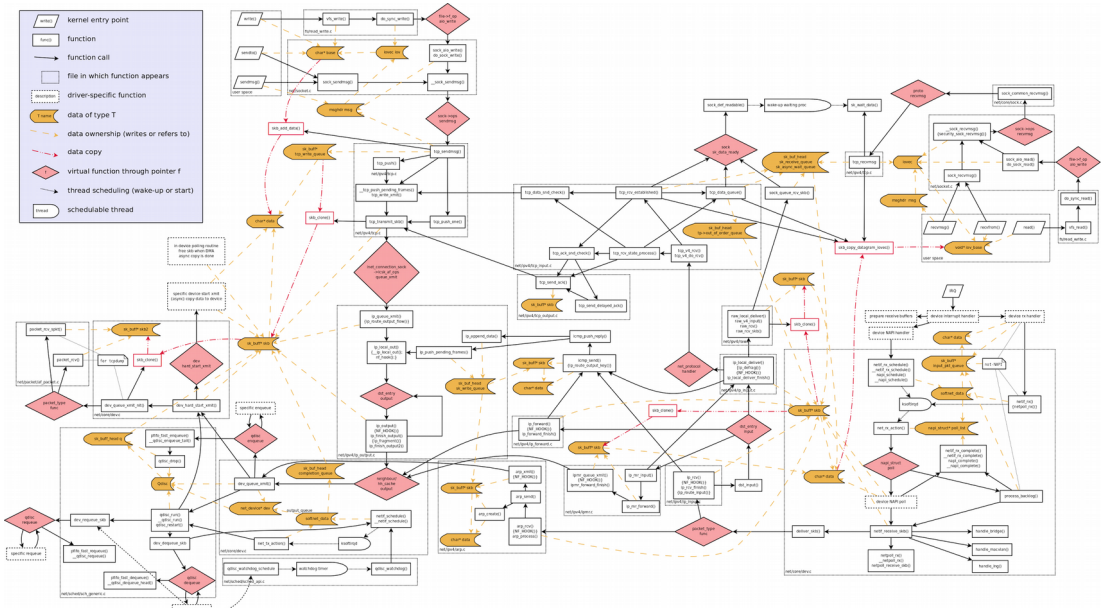


?

Why not?

- General purpose
- Have all the needed softwares
- Provide isolation through process mechanism
- Paper about running things on GPU, FPGA, ...

Kernel Network Stack is Slow



"Too much" general purpose

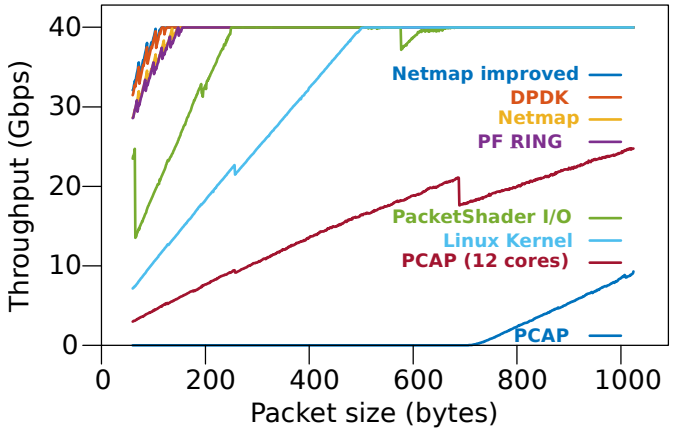
Lot of systems calls to receive and send packets

User-space <-> Kernel Space copy

First work

Lot of I/O frameworks available

Framework	Packet_mmap	PacketShader I/O	NetSlices	Netmap	PF_RING ZC	DPDK	OpenOnload
No userspace copy	X	-	-	X	X	X	X
Multiqueue	-	X	X	X	X	X	X
I/O Batching	X	X	X	X	X	X	X
Kernel bypass	-	X	X	X	X	X	X
Zero-copy forwarding	-	X	X	X	X	X	X
Zero-copy buffering	X	X	X	?	?	X	?
Devices family supported	ALL	1	ALL	8	4 ZC / ALL (non-ZC)	11	All SolarFlare
Pcap library	X	-	-	Header change	X	X	X
Socket library	X	-	-	-	X	-	X
Last kernel version supported	Last	Last	2.6.35	Last	Last	Last	Last
License	GPLv2	GPLv2	BSD	BSD	Proprietary	BSD	Proprietary
IXGBE version	Last	2.6.28	Last	Last	Last	Last	N/A

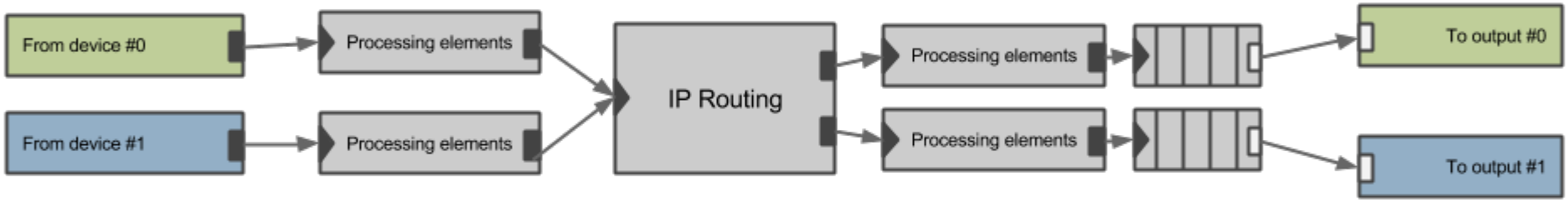


Those frameworks deliver
RAW packets
quickly to user-space

but we need to do something with them …

Click modular router

Flexible, easy-to-use, component-based configuration
Existing reusable **elements** for common networking functions
Available in **user-space** with some of the above frameworks
Possibility of **hardware offloading** of some functions

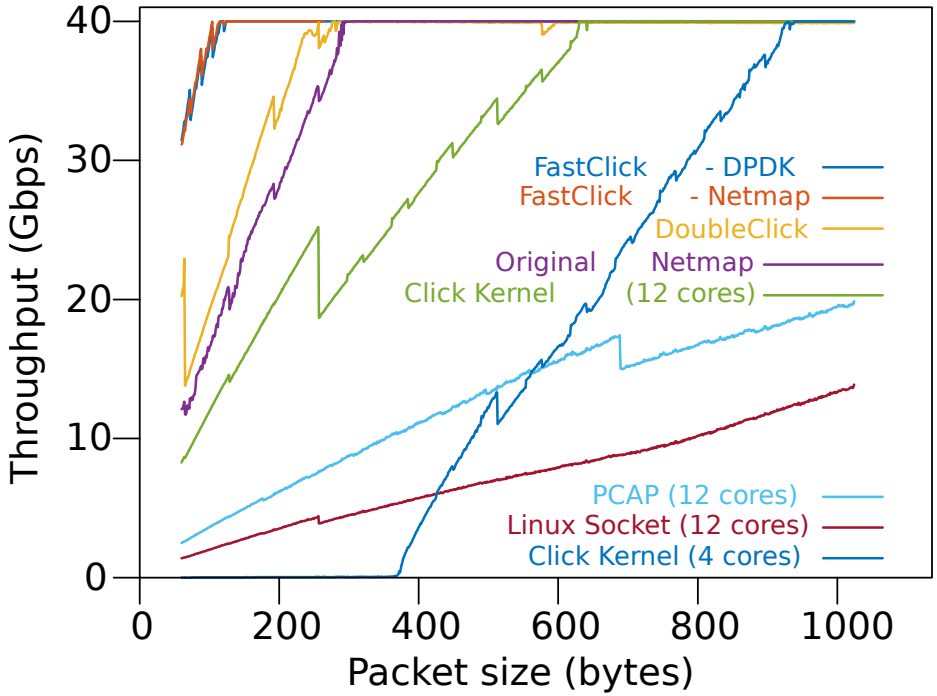


Second work

Enhance click

Multiple I/O Framework **integrations**, but perfectible

Globally enhance Click with multiqueue support, batching, zero-copy, better multithreading...



Forwarding test case

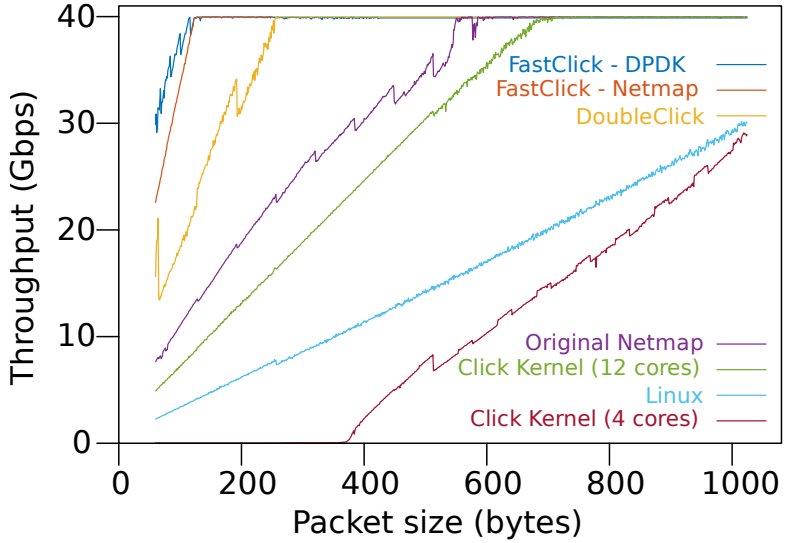
FastClick

ANCS'15

- Review the need for high speed I/O frameworks more suited than kernel API and compared them

- Using proposed ideas and new ideas of our own, we showed
that FastClick was fit for purpose as a high speed userspace packet processor and opens the door for implementation of middleboxes and NFV

- FastClick is available at
http://fastclick.run.montefiore.ulg.ac.be/



Routing test case

Current work

Add flow support to FastClick

Middlebox functionalities needs it :

Attack spitted across multiple fragmented packets
HTTP Reconstruction for ad-removal
defacing detection
Proxy-caching
DPI
…

Flows

Classification of packet into micro-flows

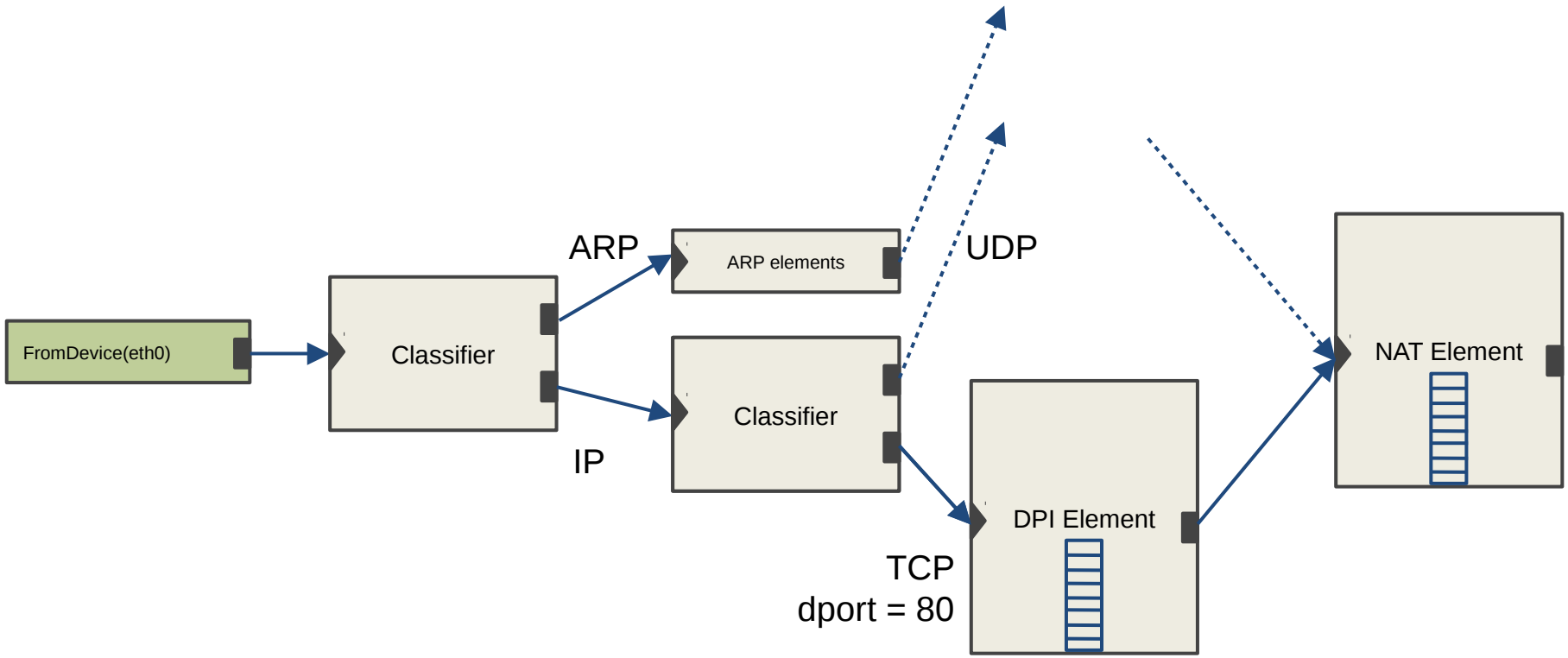
Eg. :

- TCP micro-flows
- IP Pair
- Packets from one AS to another AS
- ?

Wait for more packets mechanism

Flow-local storage

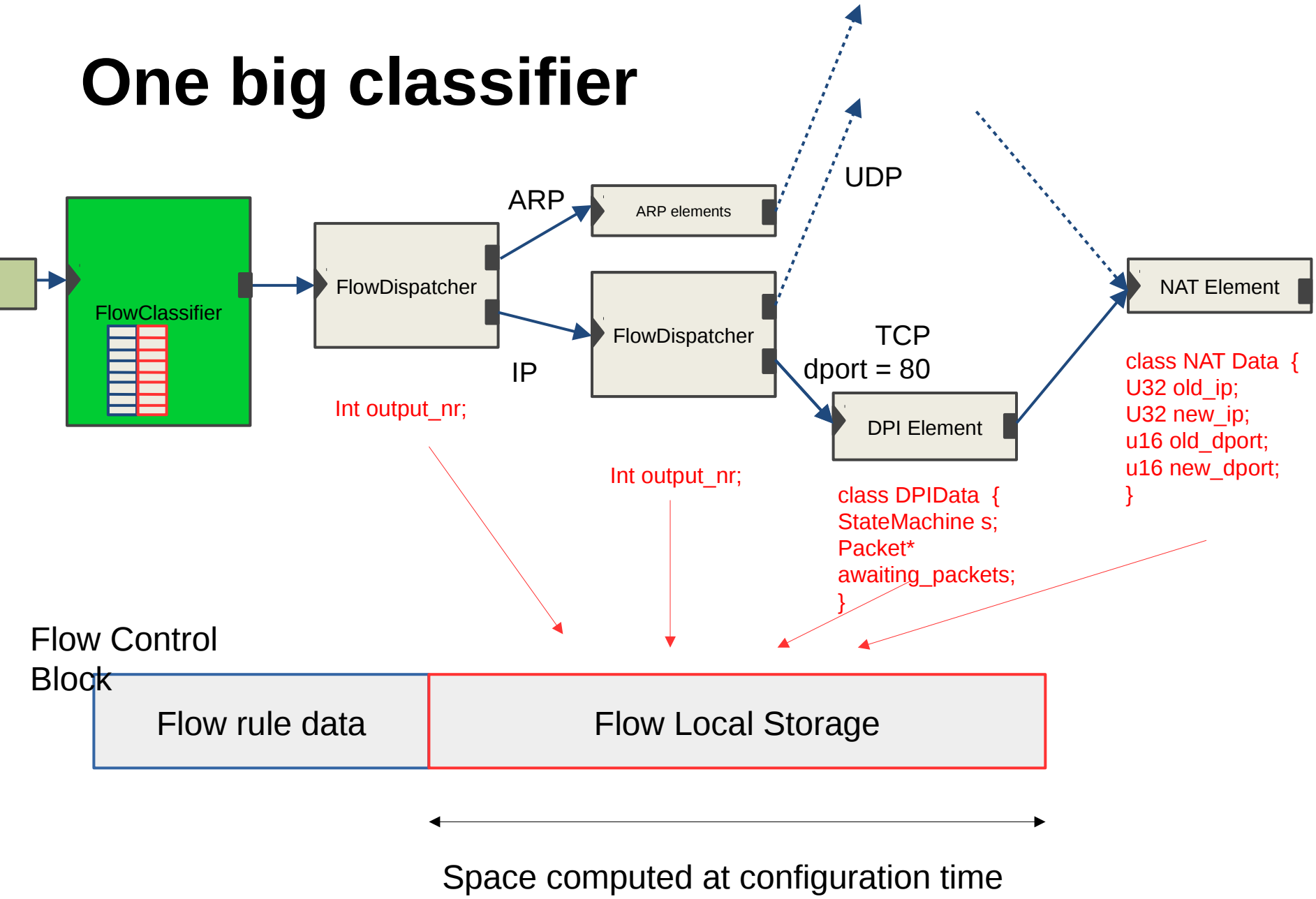
The Click way



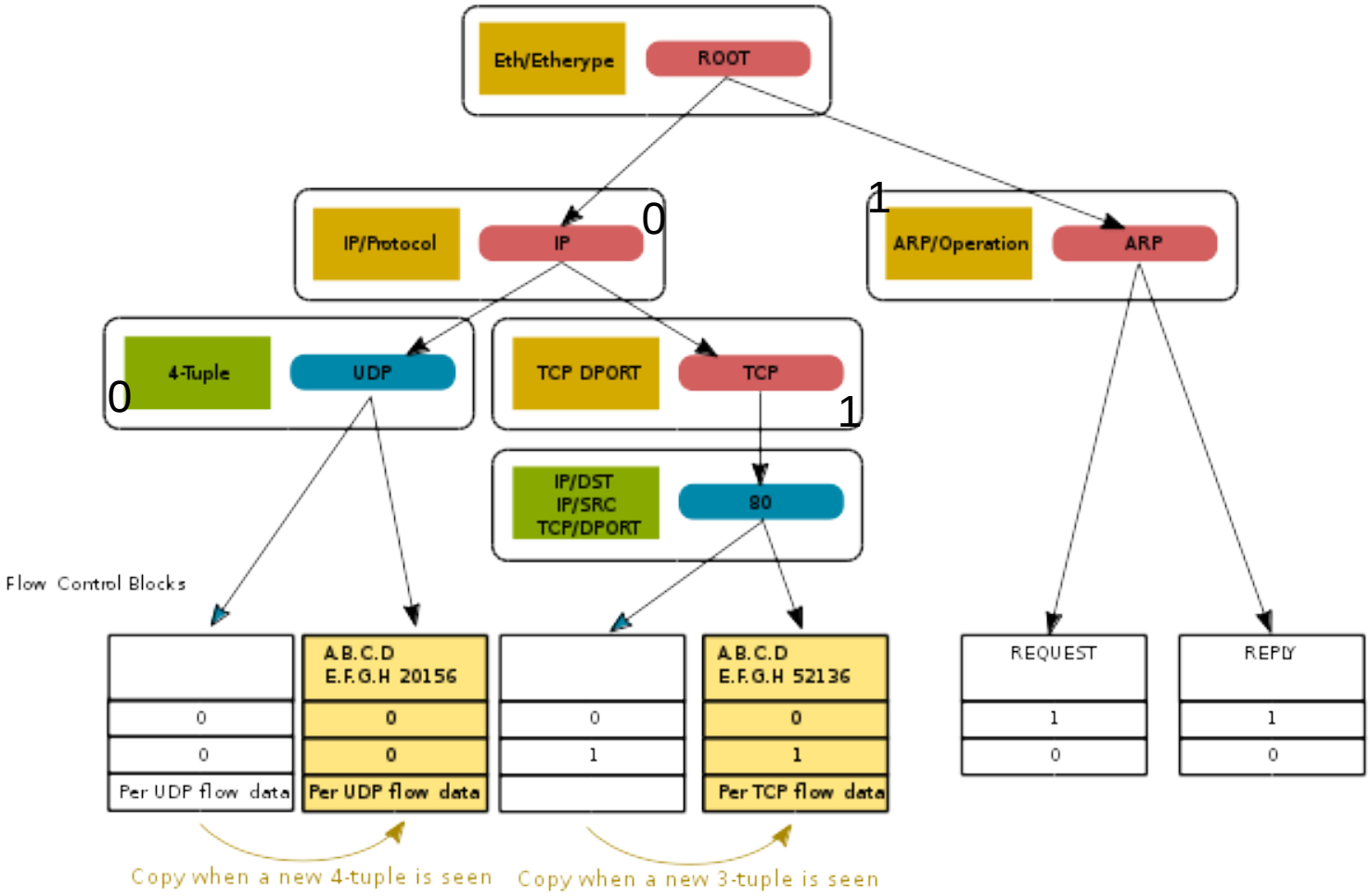
- Multiple classifiers
- Multiple flow table (each element has a hash table to access a per-flow space)

Also “the industry” way : multiple VM, reclassifying each time

One big classifier



Flow detection
One big classifier



Wait for more packets mechanism

“**Run to completion|buffer**” execution model

Try to process one packet through all elements in the pipeline.

Either :

- Execution finish in an output element
- Element asks for more packets, keeping that one in a buffer (per sub-flow, per element)

Process next packet

→ Avoid the cost of a context switch we would have with a socket

Current work

- Flow system (nearly finished)
- Quick packet classification
  - Fast rule specialization
  - Multi-dimensional
  - Partially offloadable

After that :

Make click elements compatible with some hardware devices (GPU, NetFPGA, …)

Handle the migration of those elements

→ Optimisation problem, data consistency, ...

Questions, (possibly) answers and discussion

A programmable middlebox platform that can run on top of "what's available", with fine grained virtualization and efficient consolidation.

?