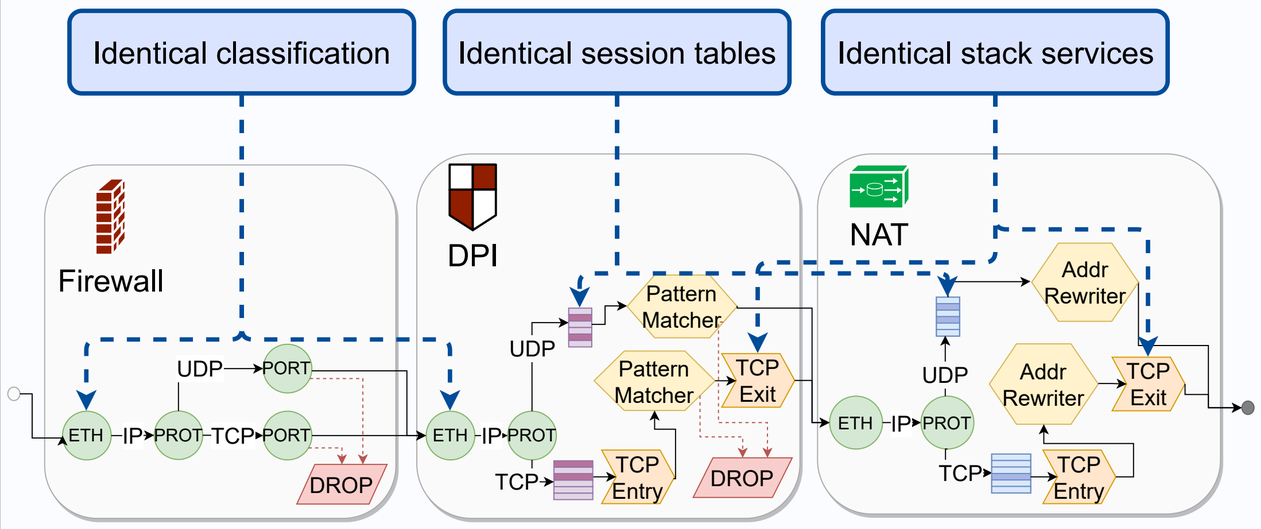
# A low-level dive into building a high-speed NFV dataplane for service chaining

Tom Barbette, Cyril Soldani, Romain Gaillard and Laurent Mathy
**University of Liege**

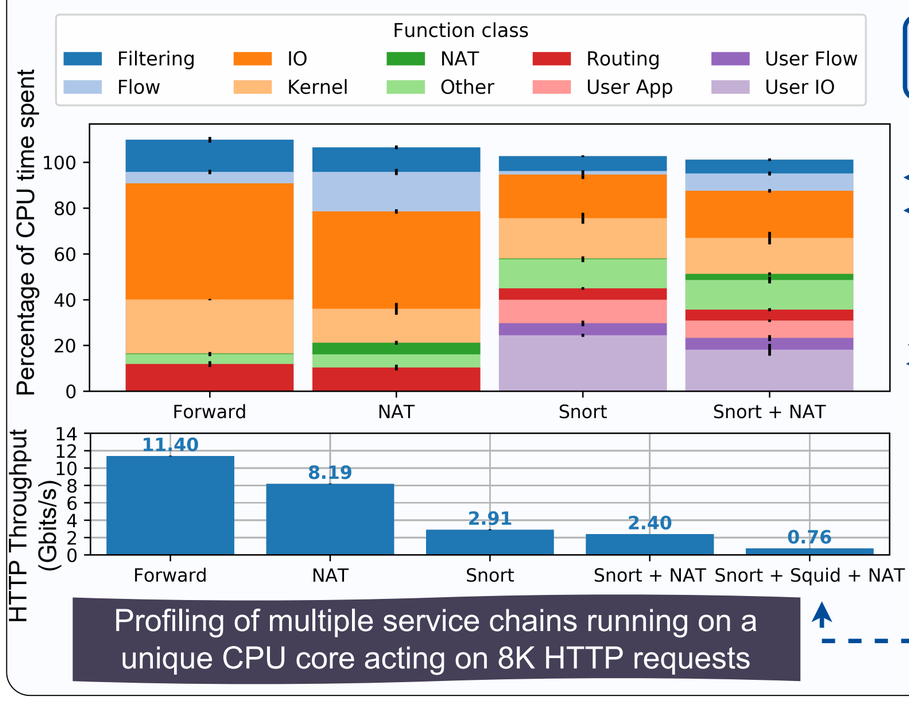
SUPERFLUIDITY EU H2020 PROJECT

## The problem

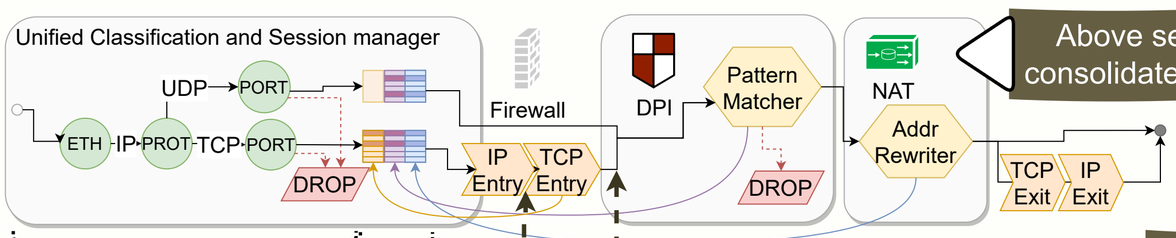
Example service chain, in which middleboxes perform redundant classification and session reconstruction operations




## Observation of its consequences




## MiddleClick architecture




Above service chain as consolidated by MiddleClick

Dynamic, minimized, once-for-all, session mappings with per-session space for each VNF

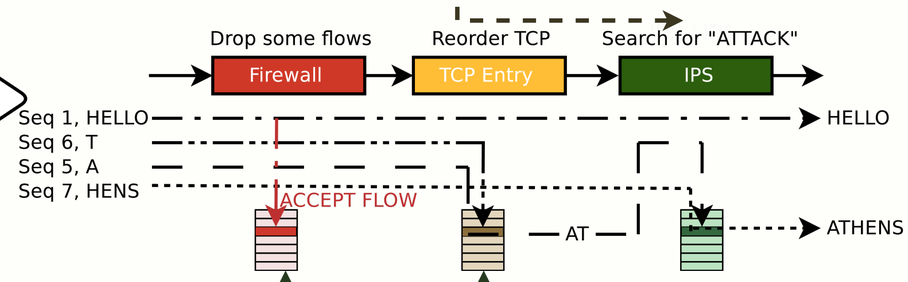
Protocol independent VNFs using generic "context" functions

Service-chain level classification (applied once)

A single TCP reconstruction for the entire service chain

Advantages of the run-to-completion-or-buffer model as permitted by the unified session table

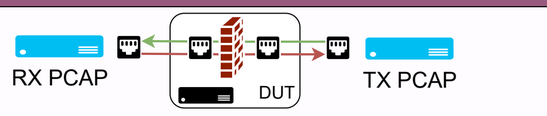
IPS can remember its DFA state while waiting for more data to make a decision. MiddleClick can send pro-active ACKs

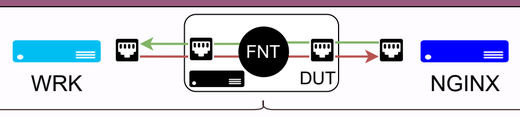



The firewall does a lookup only for the first packet of the session as it can remember the decision in the per-session space at no supplementary cost

The TCP reorderer has its space ready to put reference to out-of-order packets while waiting for the missing ones. No need for a flow table

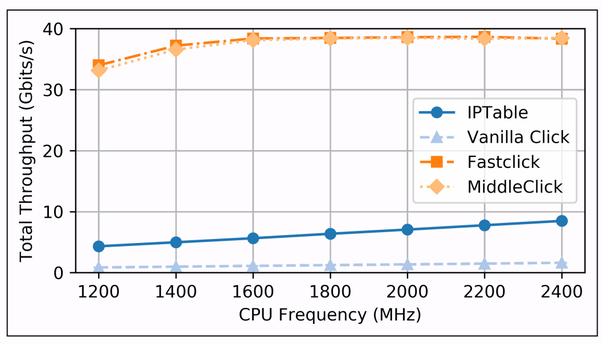
## Evaluation



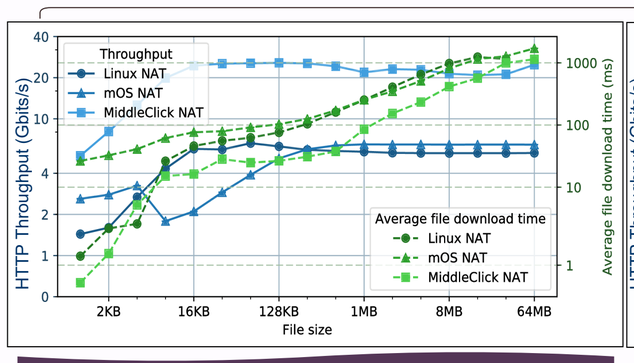




Machine A (Xeon E5-2630 v3 8*2.4GHz)
DUT (Xeon E5-2630 v3 8*2.4GHz)
Machine C (Xeon E5-2683 v4 16* 2.1GHz)

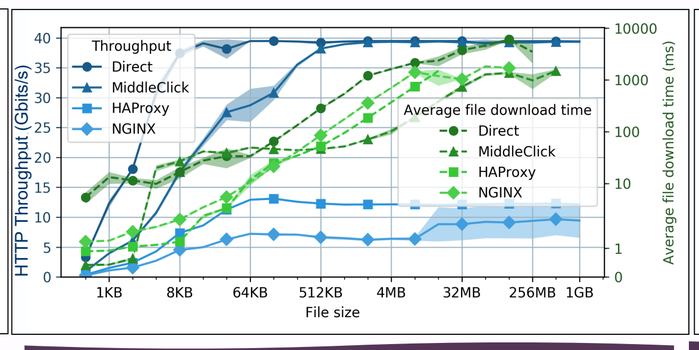
One 40G port of an Intel XL710 QDA2 NIC
DUT has two different cards



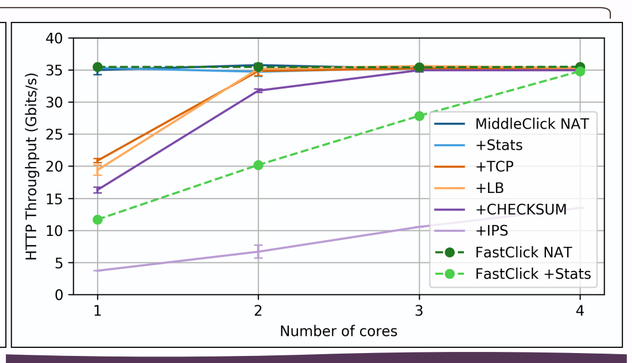

Total throughput for both sides of campus traces replayed through a **firewall** running on 2 CPU cores
**Impact of the flow system is minimal**




Throughput and latency of downloading files of varying size through a **NAT** running on 1 CPU core
**MiddleClick has better throughput and latency**

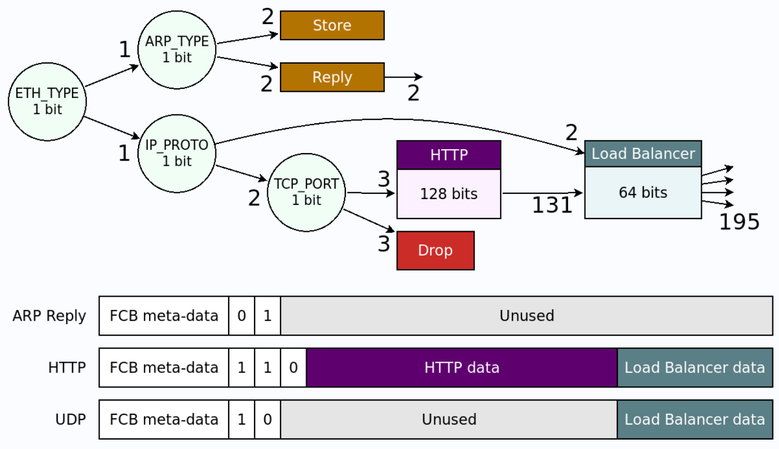



**TCP load balancing reverse proxy** running on 1 CPU core
**MiddleClick achieves 1M requests/s of 1KB files**




VNFs acting on 8K requests given an increasing number of cores
**MiddleClick exhibits minimal performance impact with increasing number of flow-based middleboxes**

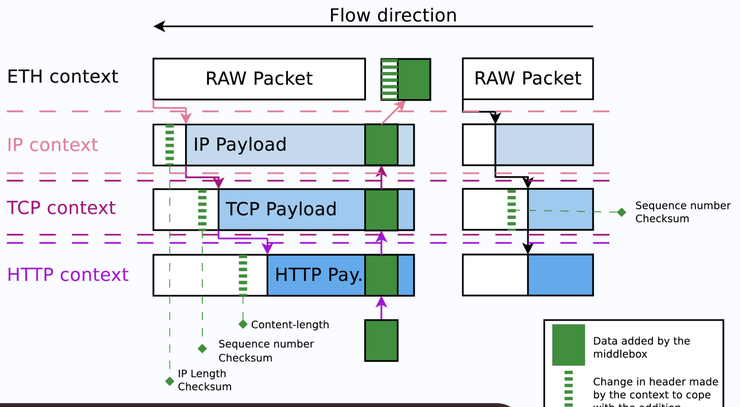
## Unified sessions




MiddleClick maintains only the necessary number of sessions by associating each flow with a flow control block (FCB), instanciated per session. The amount of needed session space is collected from all the VNFs at start-up.

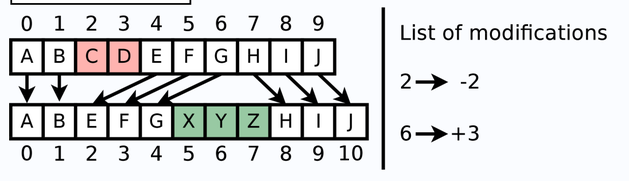
## Flow abstraction and tempering

Upon entry in a protocol context, the payload offset is moved forward according to the protocol. When the stream is modified, lower layers of context take care of the implications.




Each VNF can make requests to its context, such as determine if a packet is the last useful one of the flow, or close the connection. They can also add and remove bytes.

In an HTTP flow, the Content-Length may be changed, and then the request passed to the lower layer, likely TCP. *I.e.* the TCP context will change SEQ numbers and ACK numbers of the other side of the connection. Then, the request will pass to the lower layer, etc...




## A glimpse into state of the art solutions

### I/O Frameworks

Other works focus on providing a fast I/O platform with different layers of isolation that could replace FastClick[3]. NetBricks (Compiler-based isolation)[1], Mirage (unikernel), ClickOS (Xen-based), NetVM (KVM), OpenNetVM (container-based).

### Service chain consolidation

OpenBox[7], SNF. xOMB[2] and CoMb[5] optimize the graph to reuse some basic components and optimize the software classification. None of those work implements factorization of session management, or support flow tempering.

### NFV Dataplanes

E2[6], OpenBox[1], OpenMB and xOMB[2] focus less on low-level. They provide controllers and discuss more optimal placement, a problem not tackled in this work. While E2 provides "bytestream vports" their design is not mentioned.

### On-the-fly protocol tempering

NetBricks[4] and mOS[8] both implement a TCP stack with some similar monitoring abstractions but no support for modifications. They do not provide a generic non-TCP specific stream abstraction nor the session scratchpad facilities unified for all middleboxes.

[1] A. Bremler-Barr, Y. Harchol, and D. Hay. Openbox: a software-defined framework for developing, deploying, and managing network functions
[2] J. W. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat. xomb: extensible open middleboxes with commodity servers
[3] T. Barbette, C. Soldani, and L. Mathy. Fast userspace packet processing
[4] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker. Netbricks: Taking the v out of nfv
[5] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture
[6] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: a framework for nfv applications
[7] A. Bremler-Barr, Y. Harchol, and D. Hay. Openbox: software-defined framework for developing, deploying, and managing network functions.
[8] M. A. Jamshed, Y. Moon, D. Kim, D. Han, and K. Park. mos: A reusable networking stack for flow monitoring middleboxes