Tsinghua University

INFOCOM 2021

NFD: Using Behavior Models to Develop Cross-Platform Network Functions

Hongyi Huang, Wenfei Wu, Yongchao He, Bangwen Deng, Ying Zhang, Yongqiang Xiong, Guo Chen, Yong Cui, Peng Cheng

Tsinghua University, Facebook, Microsoft Research, Hunan University

Content

01 Background

02 Goal&Challenge

03 Design

04 Evaluation

05 Conclusion

Content

01 **Background**

02 Goal&Challenge

03 Design

04 Evaluation

05 Conclusion

Network Function (NF) / Middlebox

Various network functions are widely deployed in network & between hosts in addition to switches & routers

-- hard-coded

-- wired

Wan optimizer

Proxy

IDS

……

CDN

Firewall

NAT

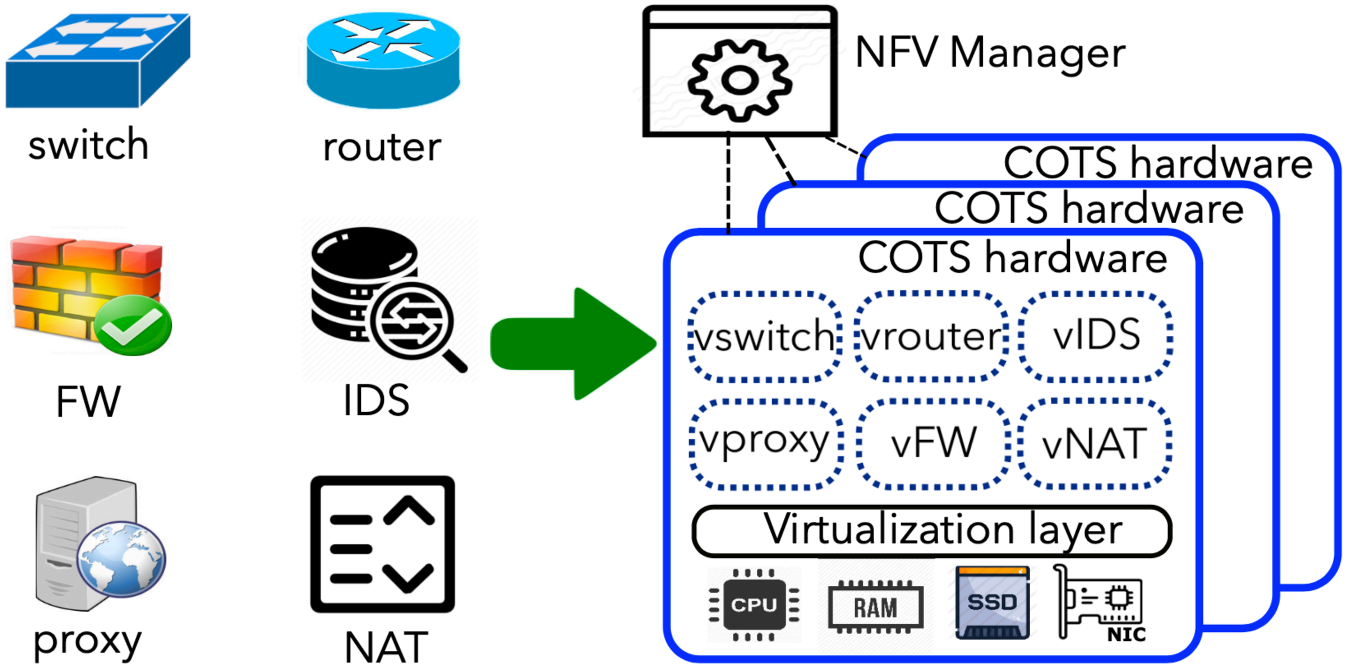
BRAS

NFV

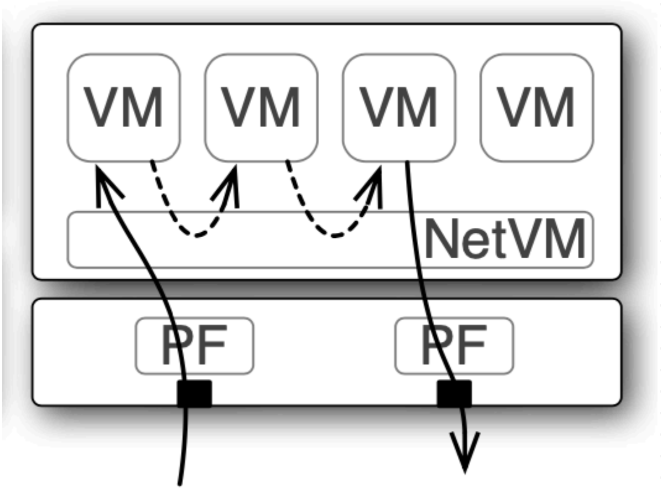
NFV replaces specialized middleboxes with software Virtual Network Functions (VNFs) consolidated on Commodity Off-The-Shelf (COTS) hardware.

-- Flexible deployment

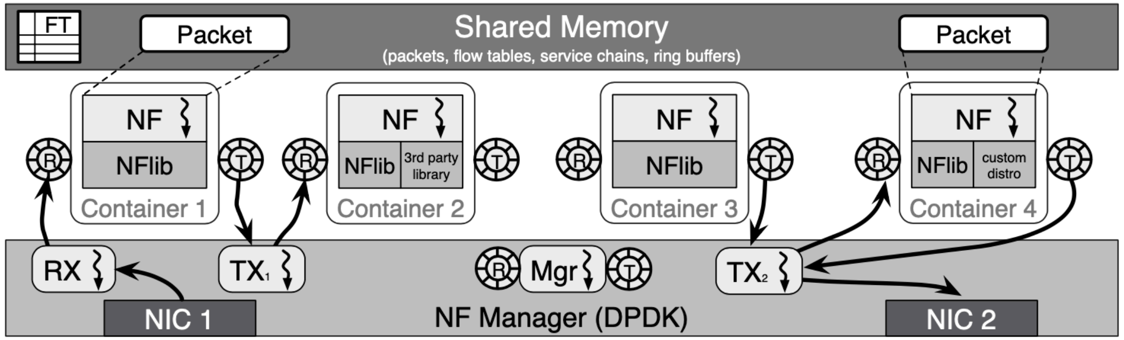
-- Quick evolution



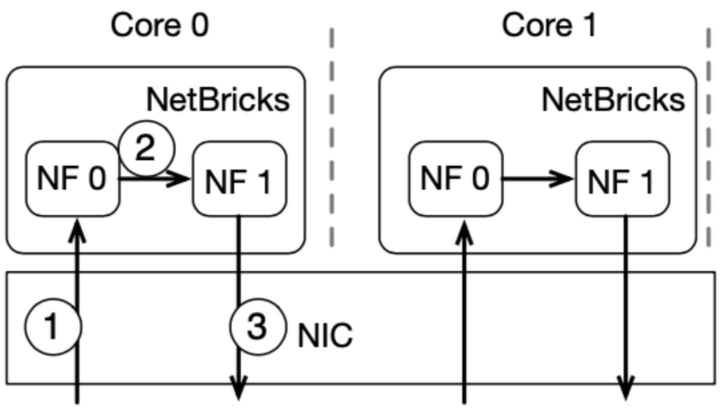
Deployment Way



Virtual Machine: NetVM, ClickOS

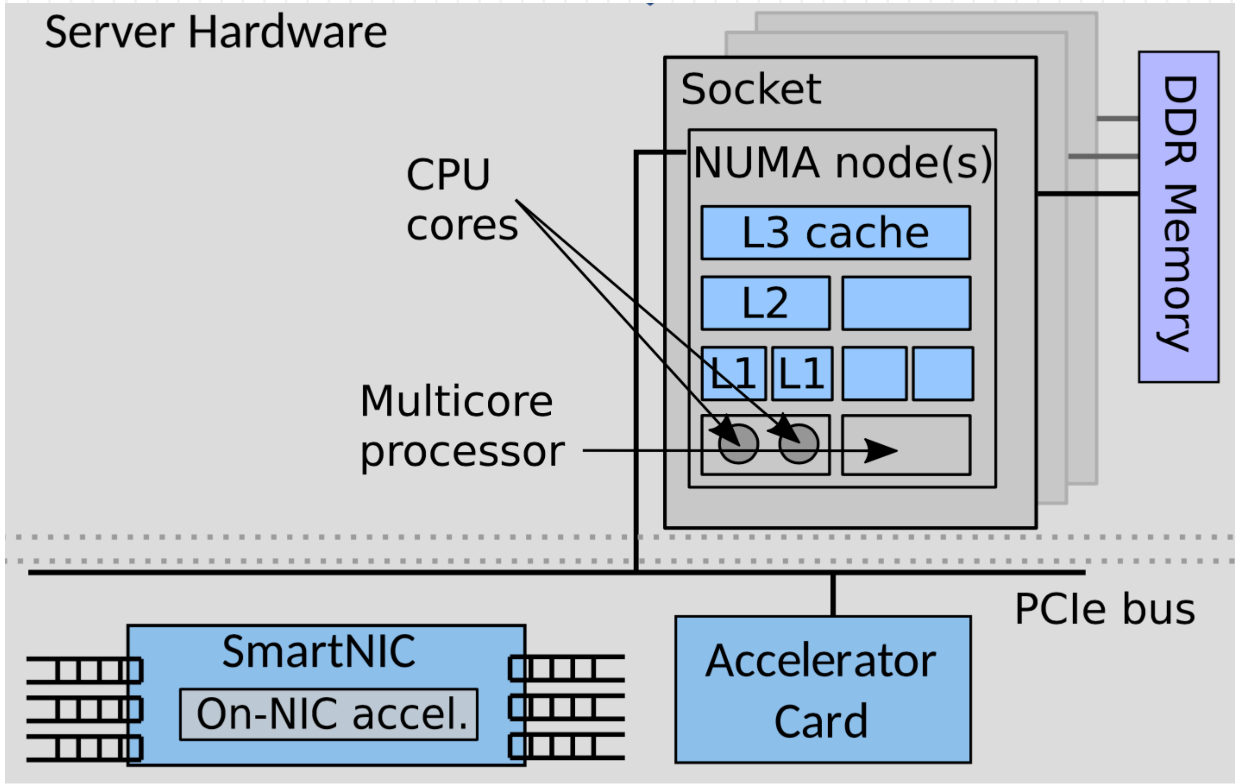


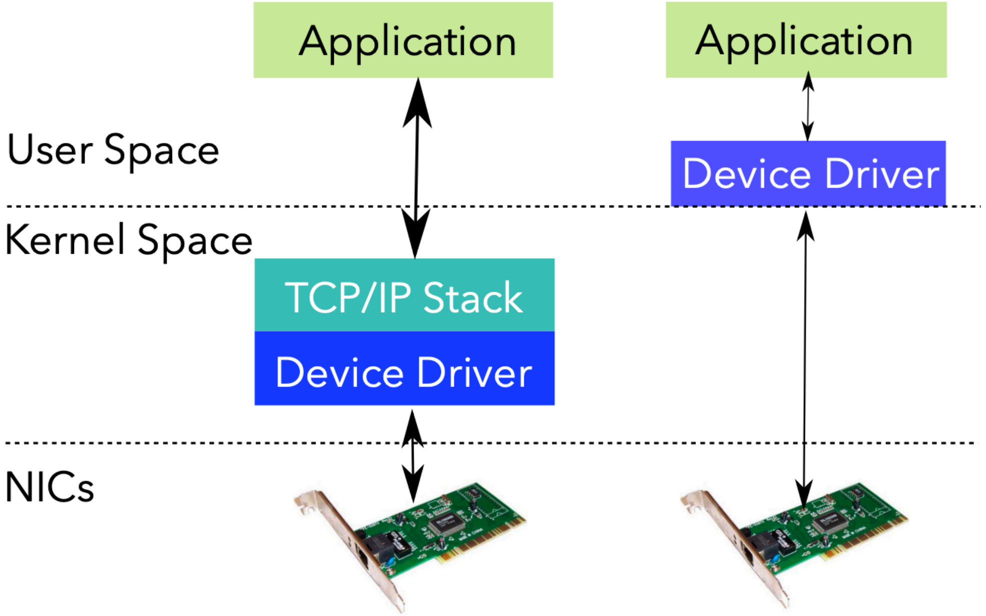
Container: OpenNetVM



Process/Function: Click, NetBricks

Acceleration Technique





Hardware-assisted: Floem, ClickNP, ResQ, SGX(Sgx-box), DPDK(many), GPU(apunet)

Software: OpenNF, S6

Integration: Cross-Product

NF Developer

IDS(Intrusion Detection System)
NAT(Network Address Translator)
Load Balancer
Rate Limiter
Cache
Monitor
…
…

m

X

Platform Provider /Developer

NetVM
OpenNetVM
Click
NetBricks
OpenNF
…
…

n

=

Network Operator

Explosion

m*n

IF m+n Possible?

m NFs + n Platforms(Compiler)

➔ m * n (integrations)

m NF models + n Platform(Abstraction + Compiler)

➔ m * n (models) + n compilers
auto integration

m NF models + 1 Framework(Abstraction + Compiler) + n Platform Plugins

➔ m (models) + n (plugins) + 1 Framework
auto integration

Content

01 Background

02 **Goal&Challenge**

03 Design

04 Evaluation

05 Conclusion

Example 1

Kernel bypassing I/O

Change packet I/O with DPDK-enabled NICs

original

```
//=== Snort.c with libpcap ===
int main(int argc, char* argv[]){
 ... // initialization
 pcap_loop(phandle,-1,pcap_handle,NULL);
}
```

modified

```
//=== Snort.c with DPDK ===
int main(int argc, char* argv[]){
 ... // initialization
 for (;;) {
  struct rte_mbuf *bufs[SIZE];
  rte_eth_rx_burst(port, 0, bufs, SIZE);
  ...
}}
```

Example2

State migration and management

Modifying PRADS and Snort to integrate with OpenNF takes more than 100 man-hours [OpenNF, StateAlyzr]

```
//=== Snort.c without OpenNF ===
int main(int argc, char* argv[]){
 ... // initialization
 pcap_loop(phandle,-1,pcap_handle,NULL);
 }
```

```
//=== Snort.c with OpenNF ===

int main(int argc, char* argv[]){
 ... // initialization
 locals.put_allflows = &put_allflows;
 sdmbn_init(&locals); // start agent
 pcap_loop(phandle,-1,pcap_handle,NULL);
}
```

Example3

secure VNFs from memory reading attacks

2.5k extra lines of code in the modification when porting an IDS to Intel SGX [SGX-box]

state

state

```
//=== PRADS.c ===
void check_vlan (packetinfo *pi) {
 config.pr_s.vlan_recv++; // a state
 ... }
void prepare_ip4 (packetinfo *pi){
 config.pr_s.ip4_recv++;  // a state
      }
```

```
//=== SGX Config ===
enclave {
 ...
 trusted{ public void check_vlan(...);
          public void prepare_ip4(...);
};};
```

Intuitions to Design New Framework

a). Most integration targets a specific piece of logic(e.g., IO in DPDK, states in OpenNF)

b). The framework should provide interfaces for *logic identification* in NFs.

c). Integration operations are tedious but regular.

Goal: Build a cross-platform development framework for NFs

Challenge

1. Expressiveness.

   Expressive to describe the packet processing logic in various NFs

2. EasyDev.

   How to save the development workload for platform provider

3. Performance.

   The outcome NFs should be logically correct and have comparable performance compared with existing legacy NFs.

Content

01 Background

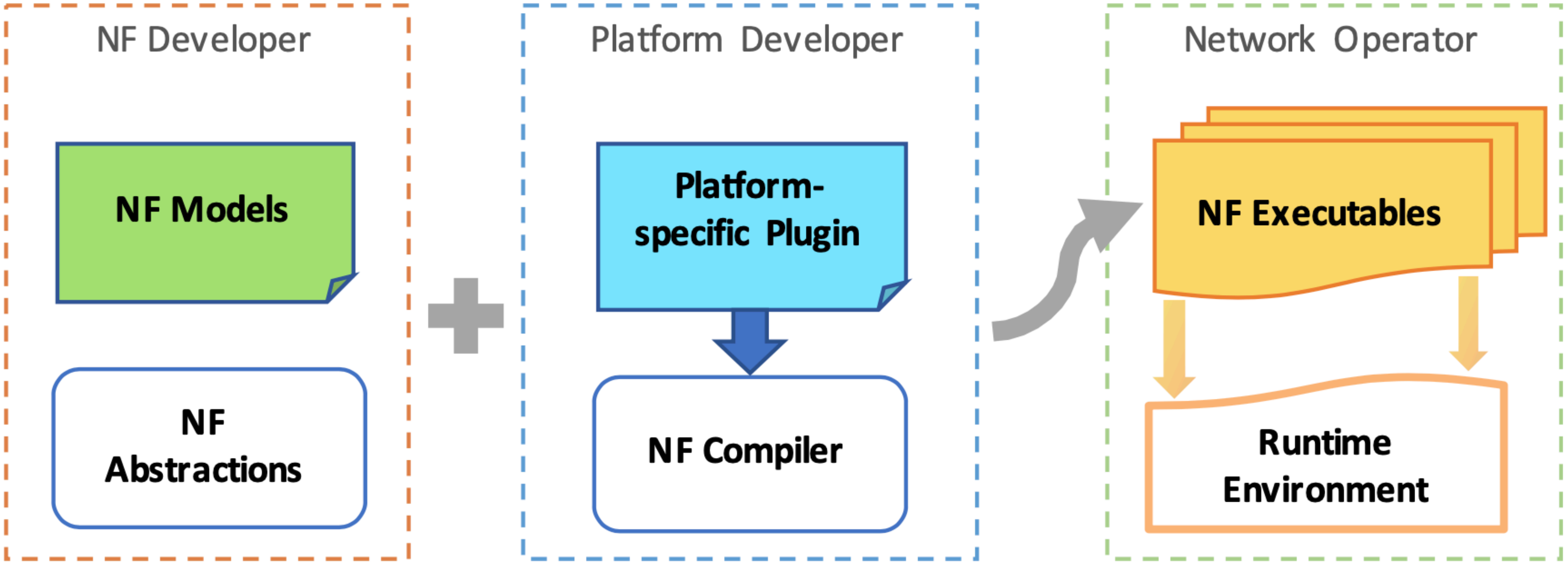
02 Goal&Challenge

03 **Design**

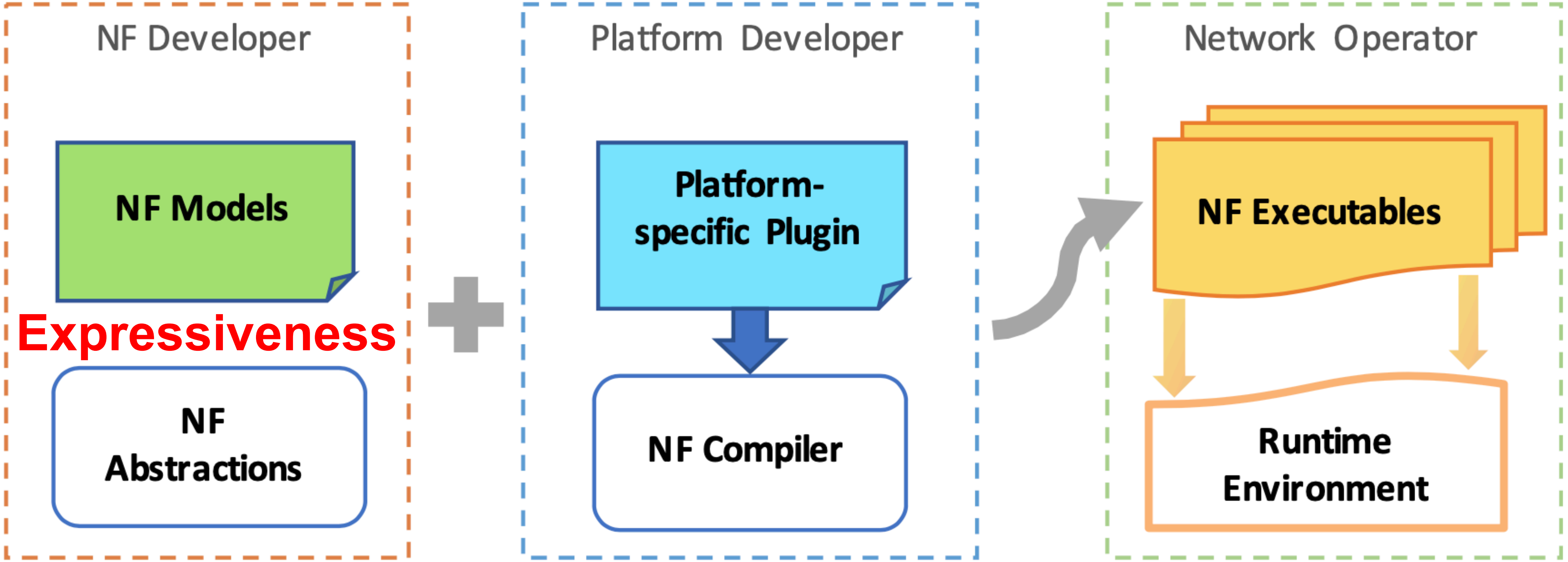
04 Evaluation

05 Conclusion

Overview


NF Developer
NF Models
NF Abstractions
Platform Developer
Platform-specific Plugin
NF Compiler
Network Operator
NF Executables
Runtime Environment

Overview


NF Developer
NF Models
Expressiveness
NF Abstractions
Platform Developer
Platform-specific Plugin
NF Compiler
Network Operator
NF Executables
Runtime Environment

Expressive Language

Existing programming frameworks:

NetCore, SNAP, SDN, Click, …

Basic types and expression

$$
\begin{array}{rcll}
\text{const} & c & ::= & (0|1)^{+} \\
\text{header field} & h & ::= & sip|dip|sport|dport|proto|\ldots \\
\text{state} & s & & \\
\text{variable} & var & & \\
\text{expression} & e & ::= & c|h|s|var|e|Expr\_Op(e_1, e_2, \ldots)
\end{array}
$$

Predicates

$$
\begin{array}{rcll}
\text{flow predicate} & x_f, y_f & ::= & \epsilon|*|h = c|\neg x_f|x_f \wedge y_f|x_f \vee y_f \\
\text{state predicate} & x_s, y_s & ::= & *|Rel\_Op(s, e)|\neg x_s|x_s \wedge y_s|x_s \vee y_s \\
\text{general predicate} & x, y & ::= & Rel\_Op(e_1, e_2, \ldots)|\neg x|x \wedge y|x \vee y
\end{array}
$$

Policies and Statements

$$
\begin{array}{rcll}
\text{flow policy} & p_f, q_f & ::= & h := e|p_f; q_f \\
\text{state policy} & p_s, q_s & ::= & s := e|p_s; q_s \\
\text{general policy} & p, q & ::= & q := e|p; q
\end{array}
$$

Model

$$
\begin{array}{rcll}
\text{model} & model & ::= & stmts \\
\text{statements} & stmts & ::= & stmt|stmt; stmts \\
\text{statement} & stmt & ::= & p|if|loop \\
\text{if statement} & if & ::= & \text{if } (x)\{(stmts\} \text{ else } \{stmts\} \\
\text{loop statement} & loop & ::= & \text{while}(x) \text{ then } \{stmts\}
\end{array}
$$

Semantics

symbols	meaning
$f[h]$	h is a header field (Figure 4 does not list all fields), and $f[h]$ is the field h in packet f.
$f[TAG]$	We append tags to each packet for flexible processing[34], which can be viewed fields of a packet.
$f[output]$	Record the output ports of a packet. $f[output] := \{p_1, p_2\}$ means sending packet f to port p_1 and p_2. $f[output] := \epsilon$ means dropping the packet.
r	Abbreviation for A rule: $h_1 = v_1 \wedge h_2 = v_2 \wedge \ldots$
$f \sqsubseteq r$	Abbr. for a flow-rule match: $f[h_1] = v_1 \wedge f[h_2] = v_2 \wedge \ldots$
R	Abbreviation for a rule set: $\{r_1, r_2, \ldots\}$
$f \sqsubseteq R$	Abbreviation for a flow-ruleset match (f match one of rules in R): $f \sqsubseteq r_1 \vee f \sqsubseteq r_2 \vee \ldots$

Programming Abstractions

Packet processing abstraction: parse, deparse, transform

Bytestream processing abstraction: TCP flow

User-defined abstraction: custom abstractions

State abstraction: manage state in granularity

```
  string type="int";  Value value=0;
  int granularity=sip&dip&sport&dport&proto;
  map<unsigned, Value> instances;
  State_Counter& operator++(){
    key=hash(pkt&MaskOf(granularity));
    if(instances.find(key)==instances.end())
      instances.put(key, value);
    instances[key]++;
} };
```

Time-driven logic abstraction: timer

Uniform Structure

Stateful Match Action Table

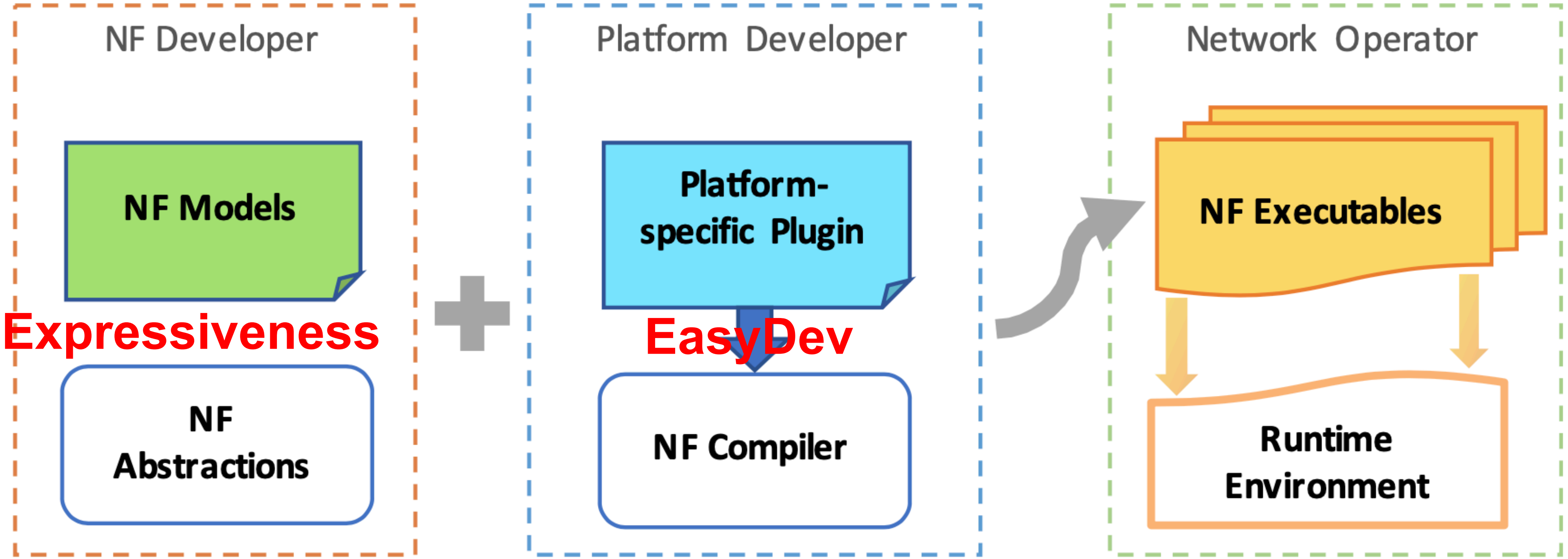
entry	*entry*	::=	if $(x_f \wedge x_s)$ then $(p_f; p_s)$ else $\perp$
SMAT	*smat*	::=	*entry*|*entry*; *model*

Rationality:

1. Existing practices in Microsoft Azure [VFP]
2. Stateless -> compatible with switch policy [SDN, P4]

	Match		Action	
	Flow	**State**	**Flow**	**State**
Stateful Firewall	**Configuration:** OK={r1, r2, ...}			
	f∈OK	-	f[output]:=IFACE	seen:=seen∪{f}
	f	f∈seen	f[output]:=IFACE	-
	f∉ OK	f∉seen	f[output]:=ϵ	-

Overview

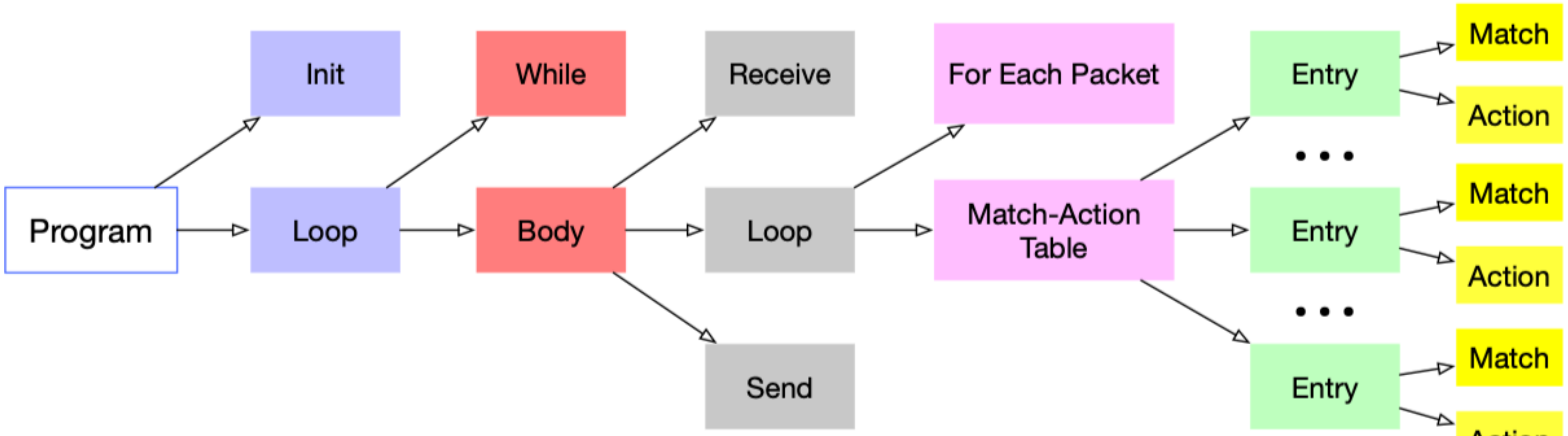

NF Developer
NF Models
Expressiveness
NF Abstractions
Platform Developer
Platform-specific Plugin
EasyDev
NF Compiler
Network Operator
NF Executables
Runtime Environment

Syntax Tree

Intermediate Representation (IR)

leaf nodes

non-leaf nodes : derived to leaf nodes



EasyDev.

Interface1: Override (DPDK, GPU)

-- replace a piece of logic by a platform enhanced implementation

(replace pcap_loop)

```
//=== Snort.c with libpcap ===
int main(int argc, char* argv[]){
  ... // initialization
  pcap_loop(phandle,-1,pcap_handle,NULL);
}
```

```
//=== Snort.c with DPDK ===
int main(int argc, char* argv[]){
      // initialization
  for (;;) {
   struct rte_mbuf *bufs[SIZE];
   rte_eth_rx_burst(port, 0, bufs, SIZE);
   ...
}}
```

EasyDev.

Interface2: Modification (OpenNF)

-- insert/delete/modify a node on the syntax tree using IR callback function (add initialization)

```
    new OpenNFVisitor.visit(syntax_tree);
  }
public class OpenNFVisitor implements NFDCompiler{
  @Override public T visitInit(...){
    AddAgentCode(...)
    InsertCode("List<State>␣allStates")
    super.visitInit(...) // orig. compilation }
  @Override public T visitStateDeclaration(...){
    super.visitStateDeclaration(...)
    stateName = ... // get the state name
    InsertCode(String.format("allStates.add(%s)",stateName))
} }
```

EasyDev.

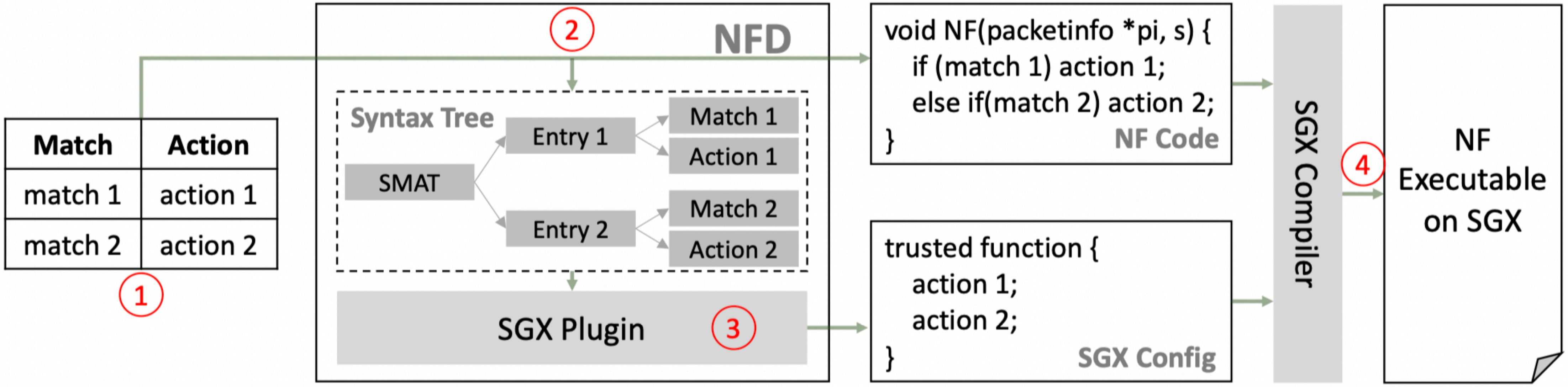
Interface3: Retrieval (SGX)

-- collect extra information (sensitive state and function)

-- use the information for platform integration (SGX config)

```
    new SGXVisitor.visit(syntax_tree);
  }
public class SGXVisitor implements NFDCompiler{
  List<String> sensitiveFunc;
  List<String> sensitiveData;
  @Override public T visitStateDeclaration(...){
    stateName = ...
    sensitiveData.add(stateName);}
  @Override public T visitStateMatch(...){
    FuncName = ...
    sensitiveFunc.add(FuncName);}
  @Override public T visitStateAction(...){
    FuncName = ...
    sensitiveFunc.add(FuncName);}
}
```

Example Workflow



Content

01 Background

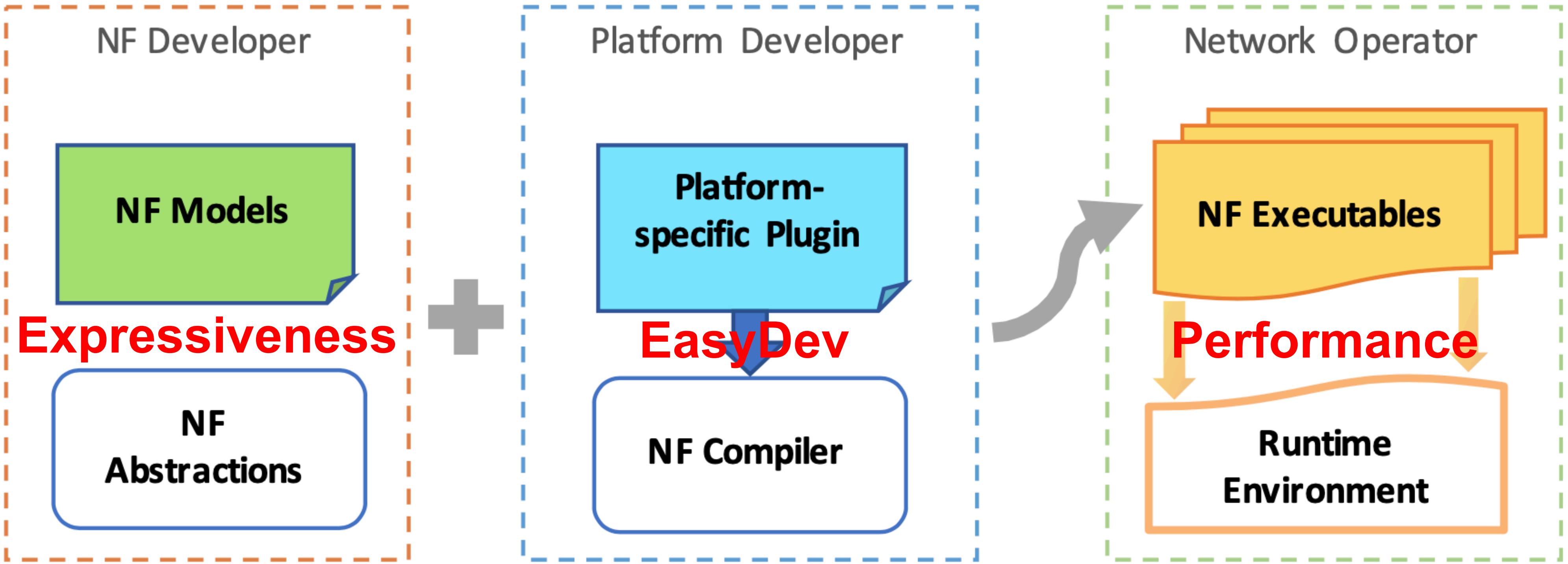
02 Goal&Challenge

03 Design

04 **Evaluation**

05 Conclusion

Overview


NF Developer
NF Models
Expressiveness
NF
Abstractions
Platform Developer
Platform-
specific Plugin
EasyDev
NF Compiler
Network Operator
NF Executables
Performance
Runtime
Environment

Prototype

Source: DSL

Compiler: ANTLR4

Target: C++

14 NFs + 6 Platforms

Component of NFD	Lines of Code
NFD model grammar	234 (g4)
compiler frontend (automatically derived by Antlr)	4.3k (Java)
compiler backend (generate C++ NF programs)	1137 (Java)
C++ template (program structure, operators) for NFs	752 (C++)
extension for OpenNF	489 (C++)
extension for GPU	668 (C++)
extension for DPDK	167 (C++)
extension for SGX	273 (C++)

Comparing Workload (LOC)

NFD: models + framework + plugins = ~ 4k

manual: ~700k

Testbed

Server:

- Intel i9 CPU (10-core, 20-thread)
- 128GB memory
- 10Gbps NIC
- three NVIDIA GTX1080 Ti graphics cards
- 1TB SSD

Trace: [IMC10]

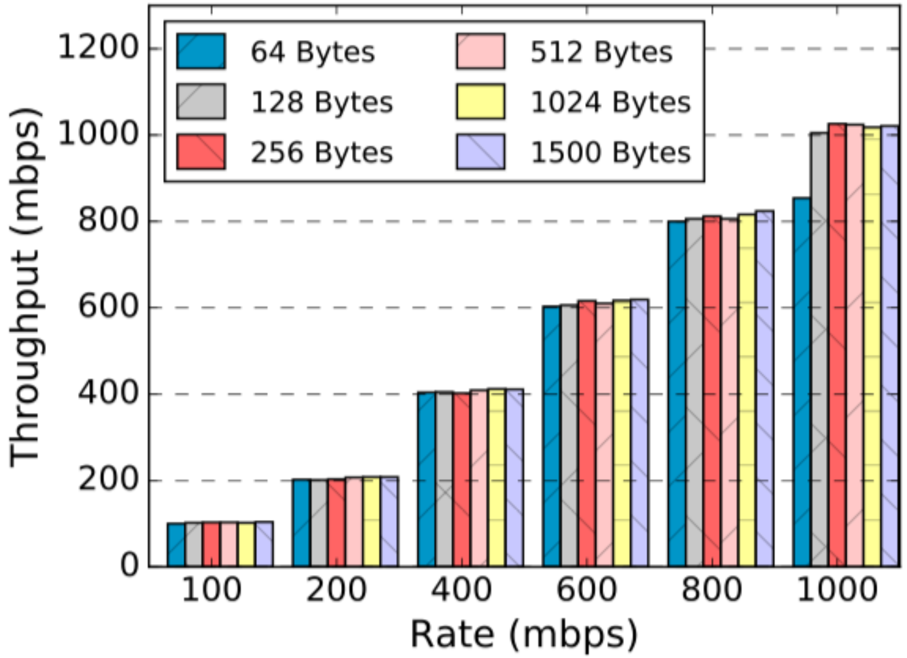
Opensource NFs: Snort, PRADS, Balance, HAProxy, Click NAT

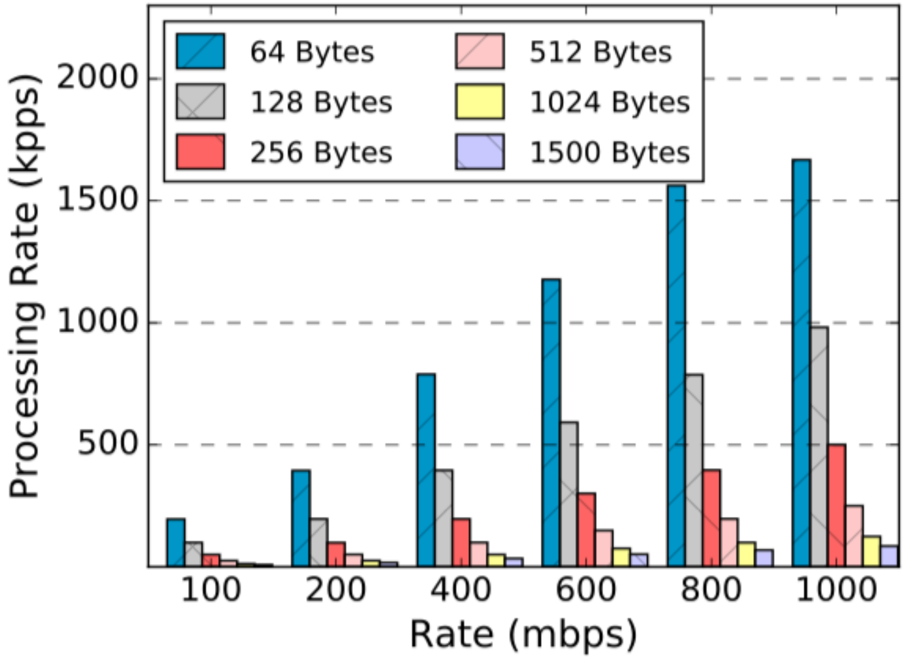
Network Traffic Characteristics of Data Centers in the Wild. IMC 10

Correctness

Rate Limiter: tuning rate

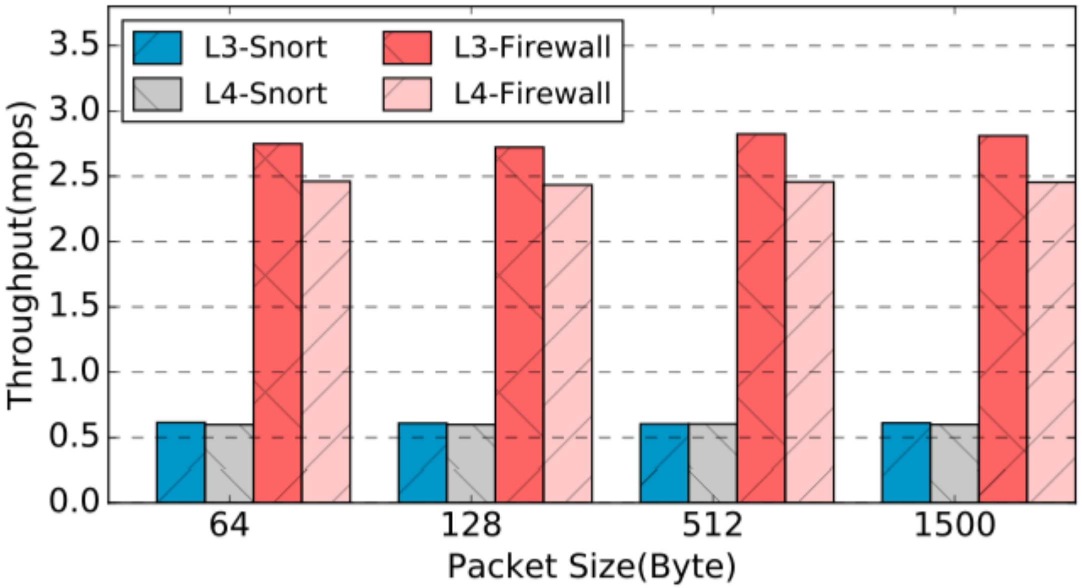
-- control the sending rate accurately as configuration



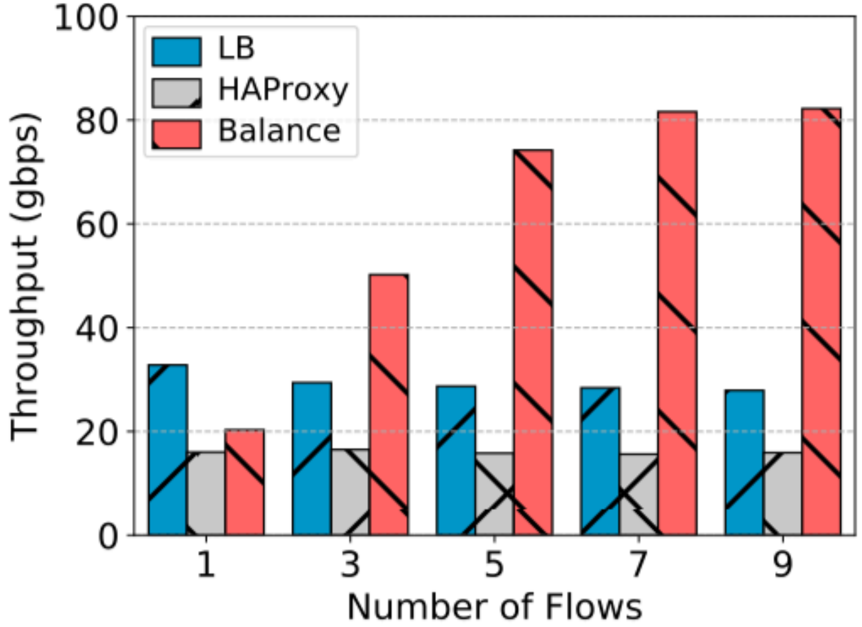


Performance

Firewall



Load Balancer

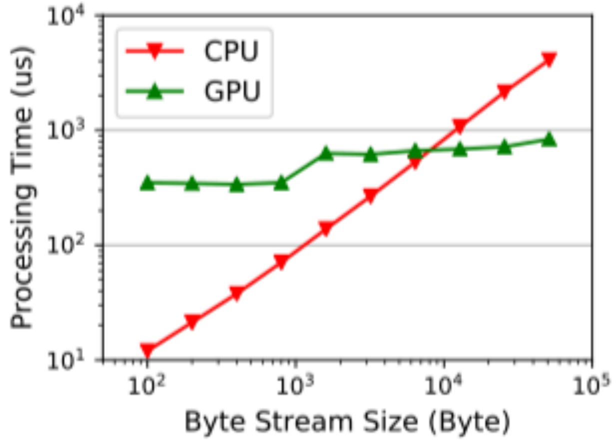


Optimization: reduce redundant logic

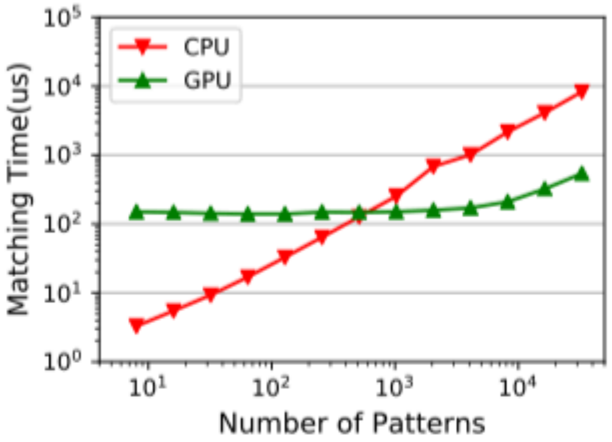
Integration

GPU

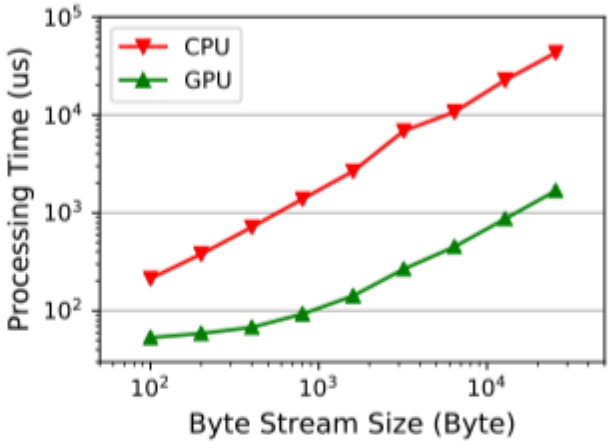
+ parallelism
- data transfer



(a) Encryption scaling up byte stream size



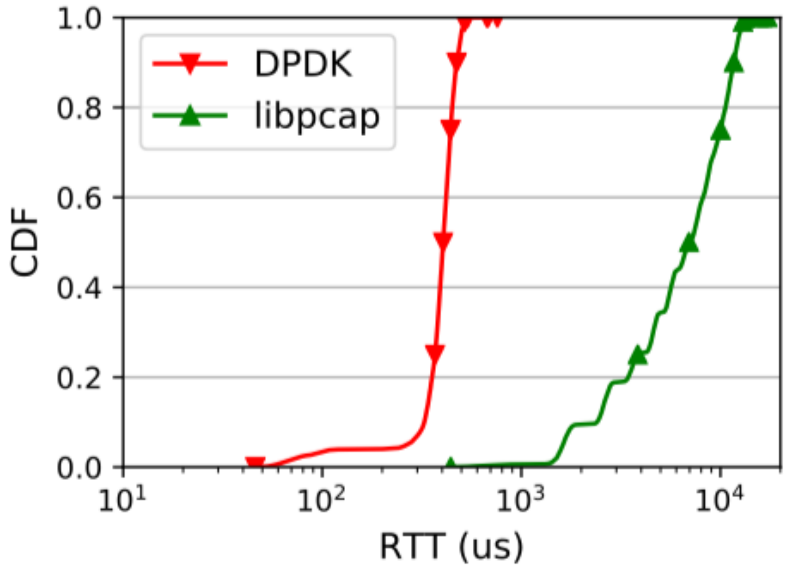
(b) Pattern matching scaling up number of patterns



(c) Pattern matching scaling up byte stream size

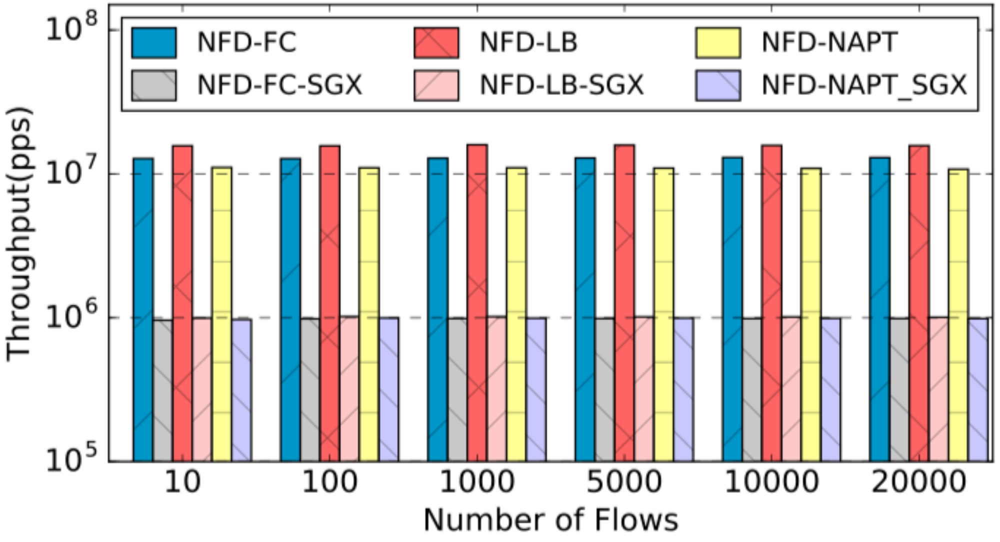
Integration

DPDK



405us v.s. 6952us

SGX



Content

01 Background

02 Goal&Challenge

03 Design

04 Evaluation

05 **Conclusion**

Conclusion

We built a cross-platform NF development framework NFD

- Platform-independent language
- Reconfigurable compiler
- Develop 14 NFs with 6 platforms
- Less workload, valid logic and performance, platform compatibility, and commodity-equivalent complex logic

清華大學
Tsinghua University

Thank You