Performance Contracts for Software Network Functions

Rishabh Iyer, Luis Pedrosa, Arseniy Zaostrovnykh,
Solal Pirelli, Katerina Argyraki, George Candea

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Software Network Functions – Pros and Cons

- Increased flexibility ✓
- Reduced capital and operating expenses ✓
- Programming errors ✗
- Unexpected performance behaviour ✗

Dealing with unexpected NF performance

- Goal: Comprehensive understanding of NF's performance profile
  - Operators – capacity planning and anticipate attacks
  - Developers – informed development decisions
- Previous work [NSDI'12, NSDI'18, SIGCOMM'18]
  - Focus on narrow subset of input workloads
  - Offer few completeness guarantees

Performance Contracts for NFs

- Abstraction for users to parameterize arbitrary input workloads
- Predict performance for workload spec without running NF
- Performance predicted as function of **Performance Critical Variables (PCVs)**
- Per-packet metrics: Instruction count, memory accesses, latency (cycles)

Outline

- What is a performance contract?
- How does Bolt generate contracts?
- Evaluation & Use-Case

Running example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Running example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Running example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Running example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

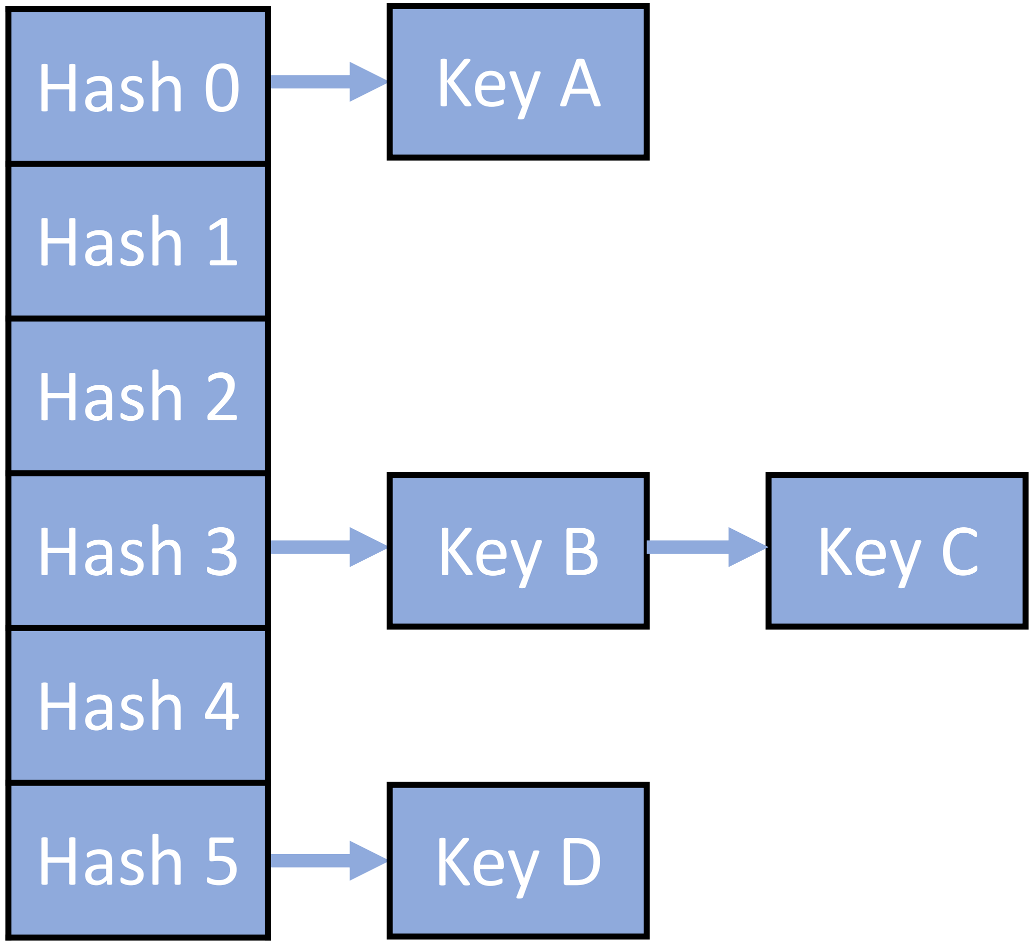
Running example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

MACtable implementation



Performance Contracts Example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance

Performance Contracts Example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	
Valid, DestMAC known	
Valid, DestMAC unknown	

Performance Contracts Example

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3C + 20$
Valid, DestMAC unknown	$3C + 100$

C = Number of hash collisions

Using performance contracts

Spec 1: Unconstrained traffic

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3\boldsymbol{C} + 20$
Valid, DestMAC unknown	$3\boldsymbol{C} + 100$

$\boldsymbol{C}$ = Number of hash collisions

Using performance contracts

Spec 1: Unconstrained traffic

$$\Rightarrow \boldsymbol{C} = \max\_collisions$$

Predicted performance:

$$3(\max\_collisions) + 100$$

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3\boldsymbol{C} + 20$
Valid, DestMAC unknown	$3\boldsymbol{C} + 100$

$\boldsymbol{C}$ = Number of hash collisions

Using performance contracts

Spec 2: No hash collisions

$\Rightarrow \boldsymbol{C} = 0$

Predicted performance:
100

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3\boldsymbol{C} + 20$
Valid, DestMAC unknown	$3\boldsymbol{C} + 100$

$\boldsymbol{C}$ = Number of hash collisions

Using performance contracts

Spec 3: Valid, no collisions,

DestMAC known

$\Rightarrow \boldsymbol{C} = 0$

Predicted performance:
20

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3\boldsymbol{C} + 20$
Valid, DestMAC unknown	$3\boldsymbol{C} + 100$

$\boldsymbol{C}$ = Number of hash collisions

Using performance contracts

Spec 3: Valid, no collisions, DestMAC known

$\Rightarrow \boldsymbol{C} = 0$

Predicted performance: 20

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3\boldsymbol{C} + 20$
Valid, DestMAC unknown	$3\boldsymbol{C} + 100$

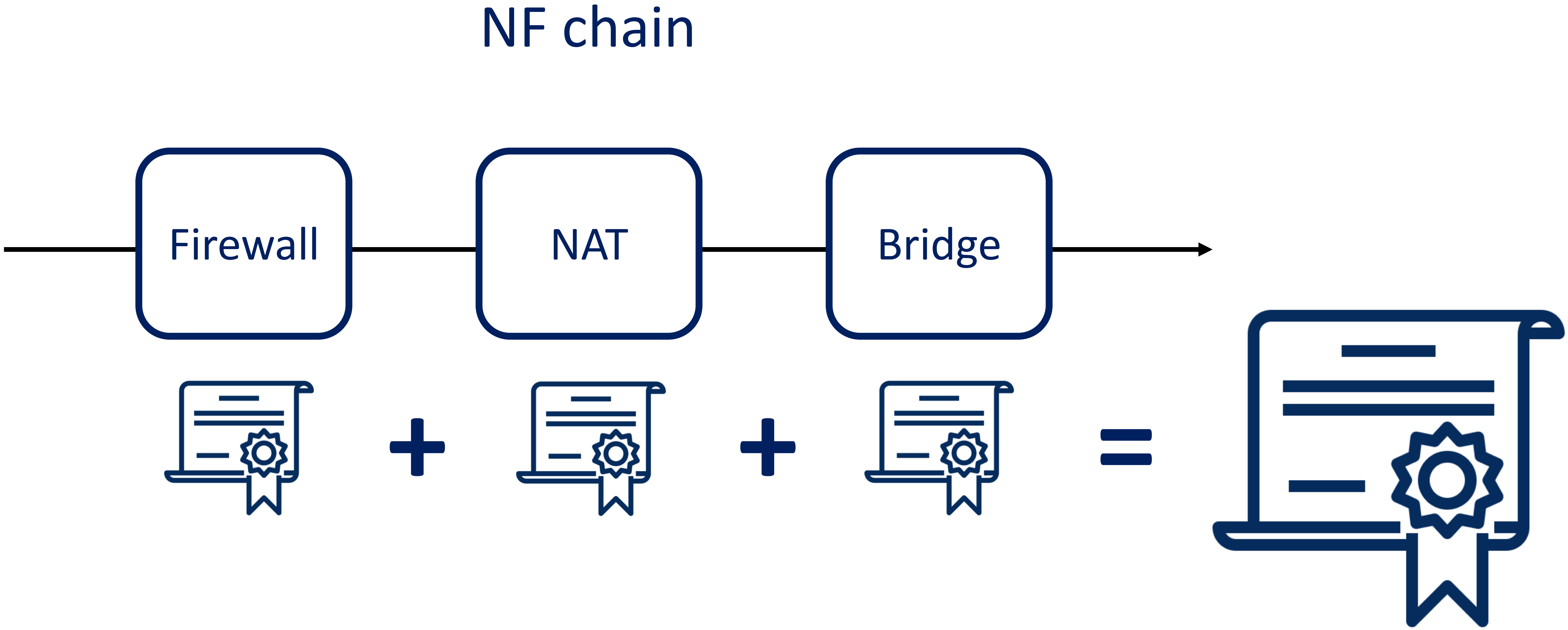
$\boldsymbol{C}$ = Number of hash collisions

Contracts quantify performance for all traffic classes of the NF
Users query contract for performance of specific input workloads

Outline

- What is a performance contract?
- How does Bolt generate contracts?
- Evaluation & Use-Case

Generating performance contracts recursively



Generating performance contracts recursively

Individual NF

Stateless Code

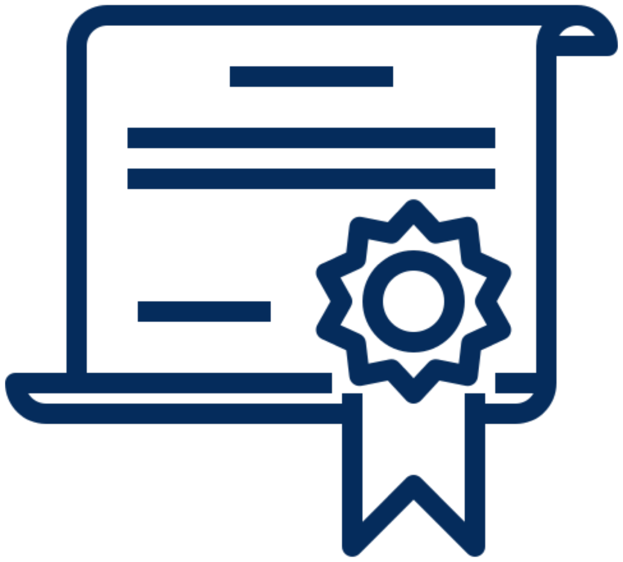
```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

Stateful NF
data structures

MACtable
put(), get()

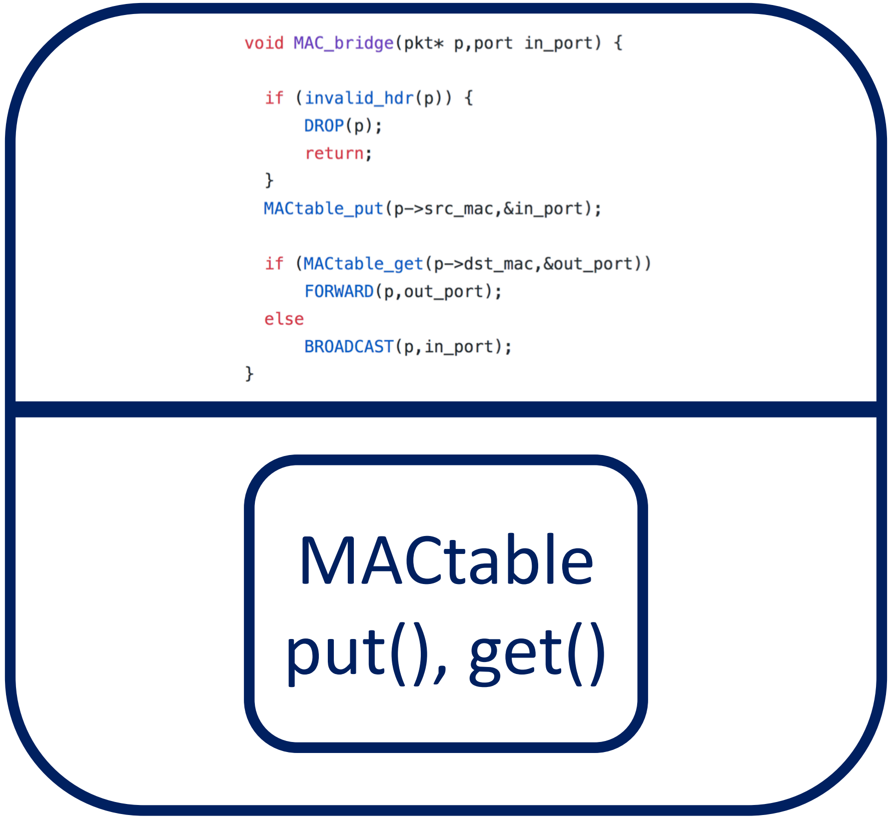


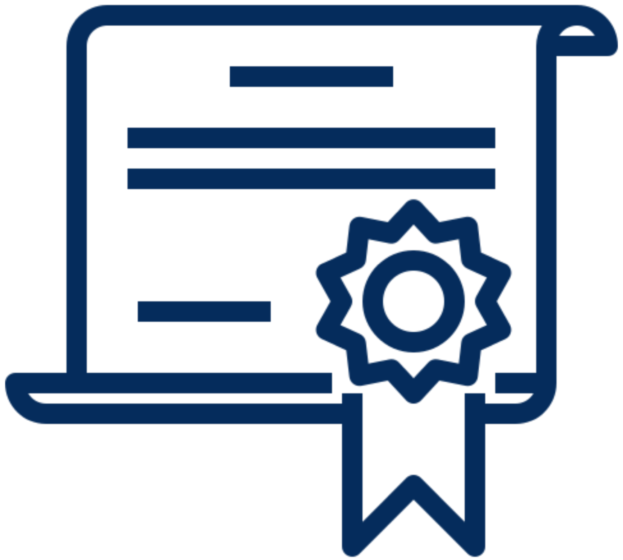
Generating performance contracts recursively

Individual NF

Stateless Code
(Simple to analyze)*

Stateful NF
data structures
(Hard to analyze)*





*A.Zaostrovnykh, S.Pirelli, L.Pedrosa, K.Argyraki, G.Candea "A Formally Verified NAT" SIGCOMM 2017

Generating performance contracts recursively

- Well defined separation between stateful and stateless NF code*
  - NFs typically have well defined, isolated state
- Encapsulate NF state using a library of data structures
- Stateful data structures – Base case of recursive process
  - Analyze once, reuse across NFs

*A.Zaostrovnykh, S.Pirelli, L.Pedrosa, K.Argyraki, G.Candea "A Formally Verified NAT" SIGCOMM 2017

Analyzing stateful data structures

$$Performance_{NF} = f(input\ packet, NF\ state, config, ..)$$

- Cannot account for all possible packet histories -> Path explosion
- BUT, performance of MACtable depends **ONLY** on number of hash collisions

Performance Critical Variables (PCVs)

- Abstract away NF state specificities
- Succinctly summarize impact of packet history, configuration on performance
- Tailor legibility and detail to audience

Contract for MACtable_put

Traffic Class	Performance
Unconstrained	$1\boldsymbol{C} + 2$

Contract for MACtable_get

Traffic Class	Performance
Key present	$2\boldsymbol{C} + 12$
Key absent	$2\boldsymbol{C} + 7$

$\boldsymbol{C}$ = Number of hash collisions

Only PCV required to summarize perf in terms of lines of pseudo-code

Generating Performance Contracts for NFs

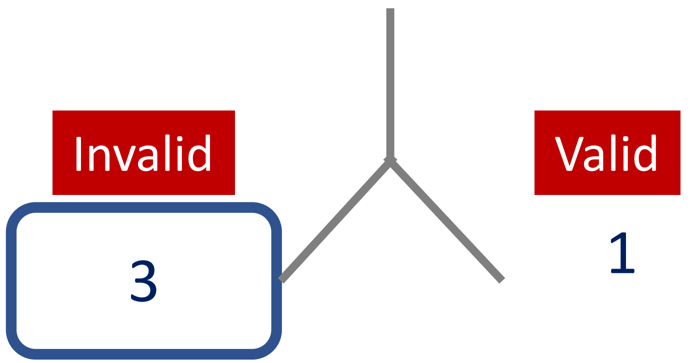
- Symbolically execute stateless code to traverse all execution paths
- While traversing each path
  - Keep track of performance metrics for stateless code
  - Plug in contracts for stateful code using path constraints

Generating Performance Contracts for NFs

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

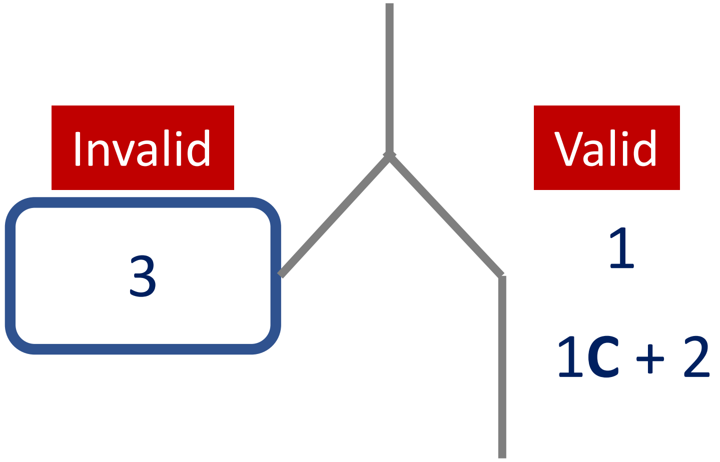


Generating Performance Contracts for NFs

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```



Traffic Class	Performance
Unconstrained	$1C + 2$

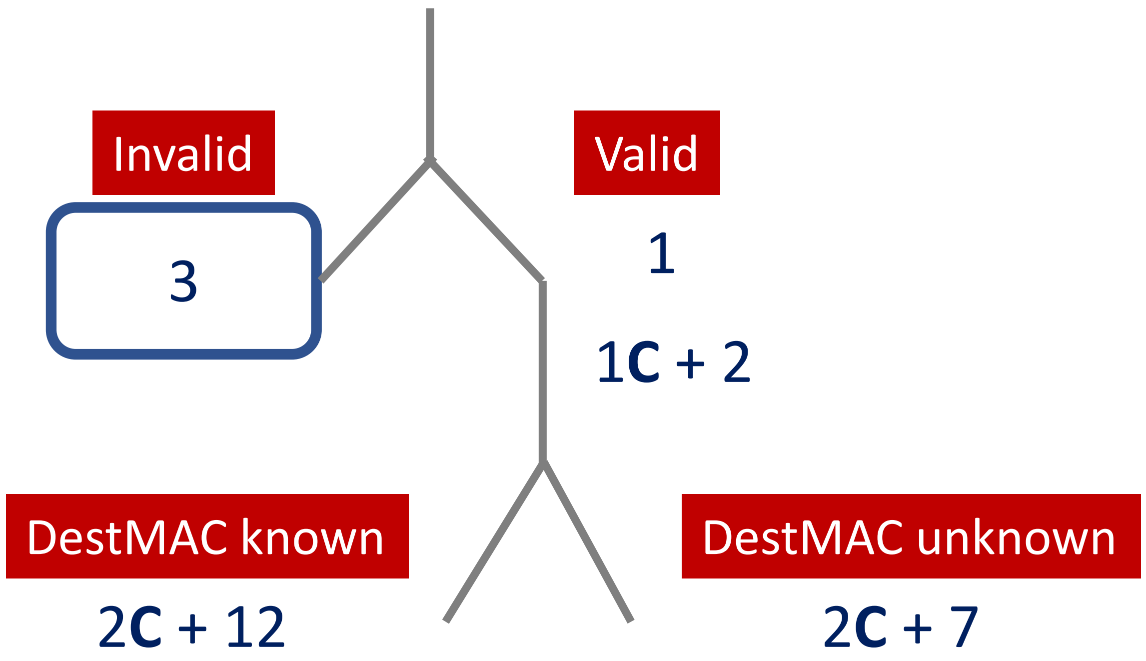
Contract for MACtable_put

Generating Performance Contracts for NFs

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```



Traffic Class	Performance
Key present	$2C + 12$
Key absent	$2C + 7$

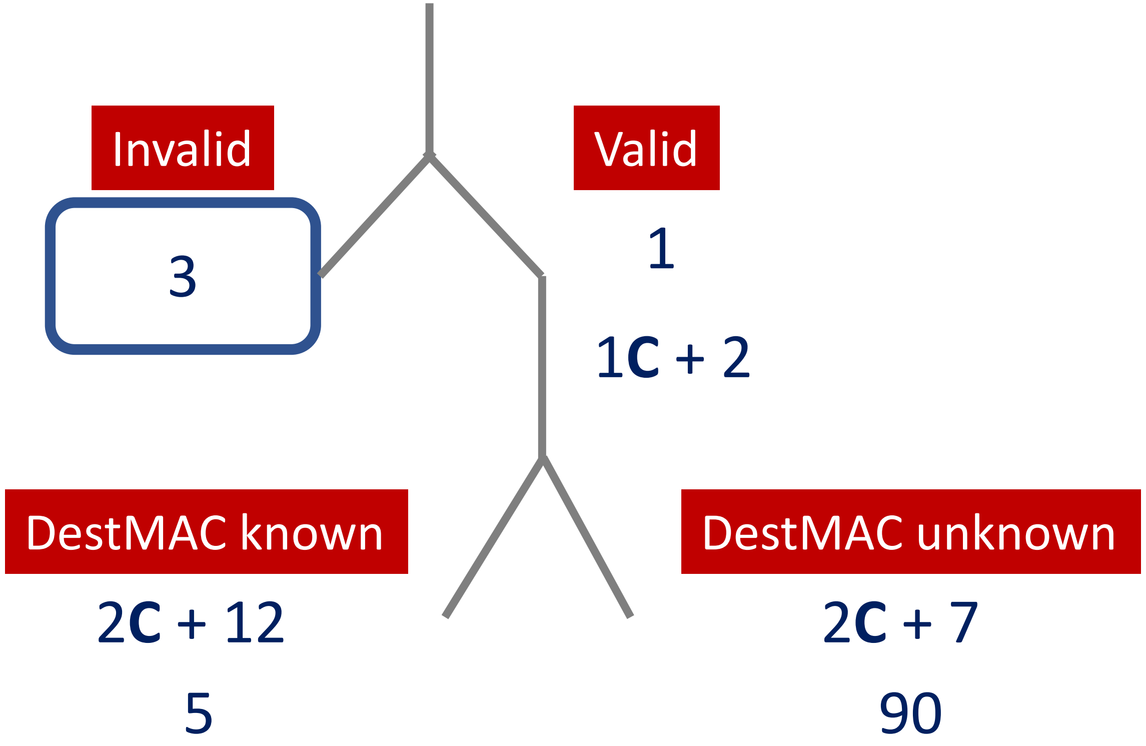
Contract for MACtable_get

Generating Performance Contracts for NFs

```
void MAC_bridge(pkt* p,port in_port) {

  if (invalid_hdr(p)) {
      DROP(p);
      return;
  }
  MACtable_put(p->src_mac,&in_port);

  if (MACtable_get(p->dst_mac,&out_port))
      FORWARD(p,out_port);
  else
      BROADCAST(p,in_port);
}
```

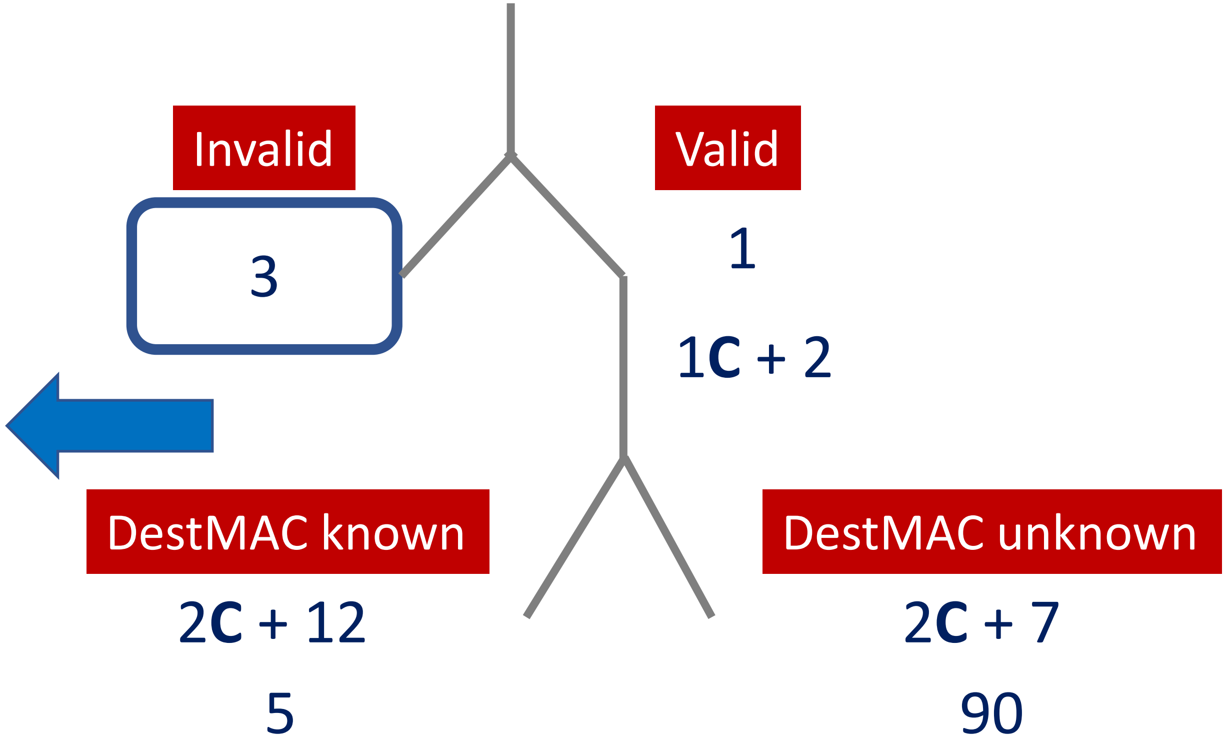


Generating Performance Contracts for NFs

Performance Contract for MAC_bridge

Traffic Class	Performance
Invalid Header	3
Valid, DestMAC known	$3\boldsymbol{C} + 20$
Valid, DestMAC unknown	$3\boldsymbol{C} + 100$

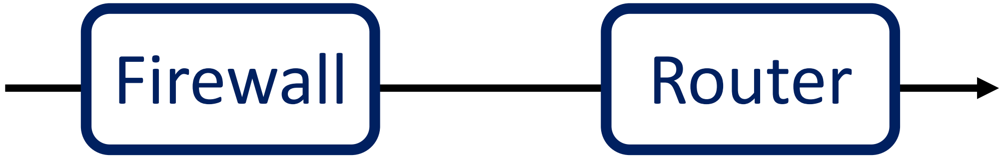
$\boldsymbol{C}$ = Number of hash collisions



Performance Contracts for NF chains

- Generate performance contracts for individual NFs in chain
- Pair together traffic classes from communicating NFs
- For each pair - AND respective constraints together
    - Equate packet sent by first NF to packet received by second

Performance Contract for NF chains - Example



Drops packets with IP options

Fast path – No IP options
Slow path – IP options

Firewall

	Traffic Class	Perf
F1	IP opt	20
F2	No IP opt	50

Router

Traffic Class	Perf	
IP opt	500	R1
No IP opt	60	R2

NF chain

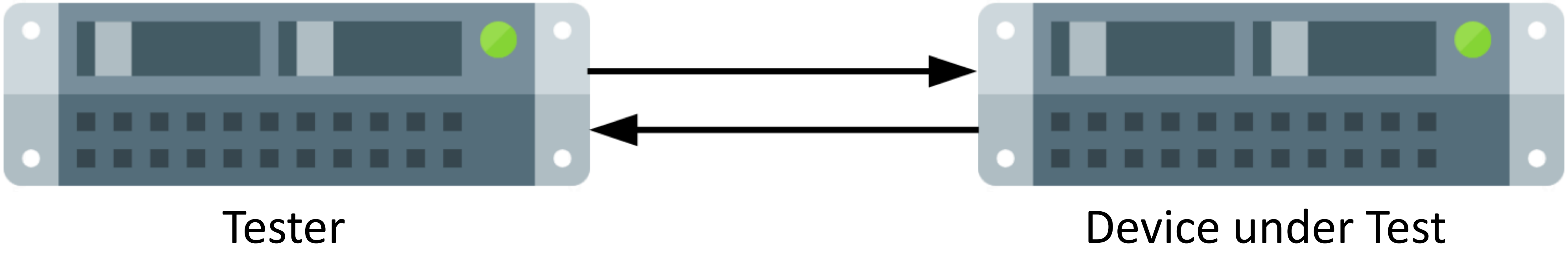
Traffic Class	Perf	
IP opt	20	<F1>
No IP opt	110	<F2, R2>

Outline

- What is a performance contract?
- How does Bolt generate contracts?
- Evaluation and Use-Case

Evaluation setup & methodology

- 4 NFs - NAT, Maglev-like LB, MAC bridge, LPM router
  - Analyze NF logic + DPDK + NIC driver*
- Metrics – instructions executed, memory accesses, execution cycles
- Testbed - Intel Xeon E5-2667v2 3.3GHz, 82599ES 10Gb NICs
- Compare predicted vs measured performance for various packet classes

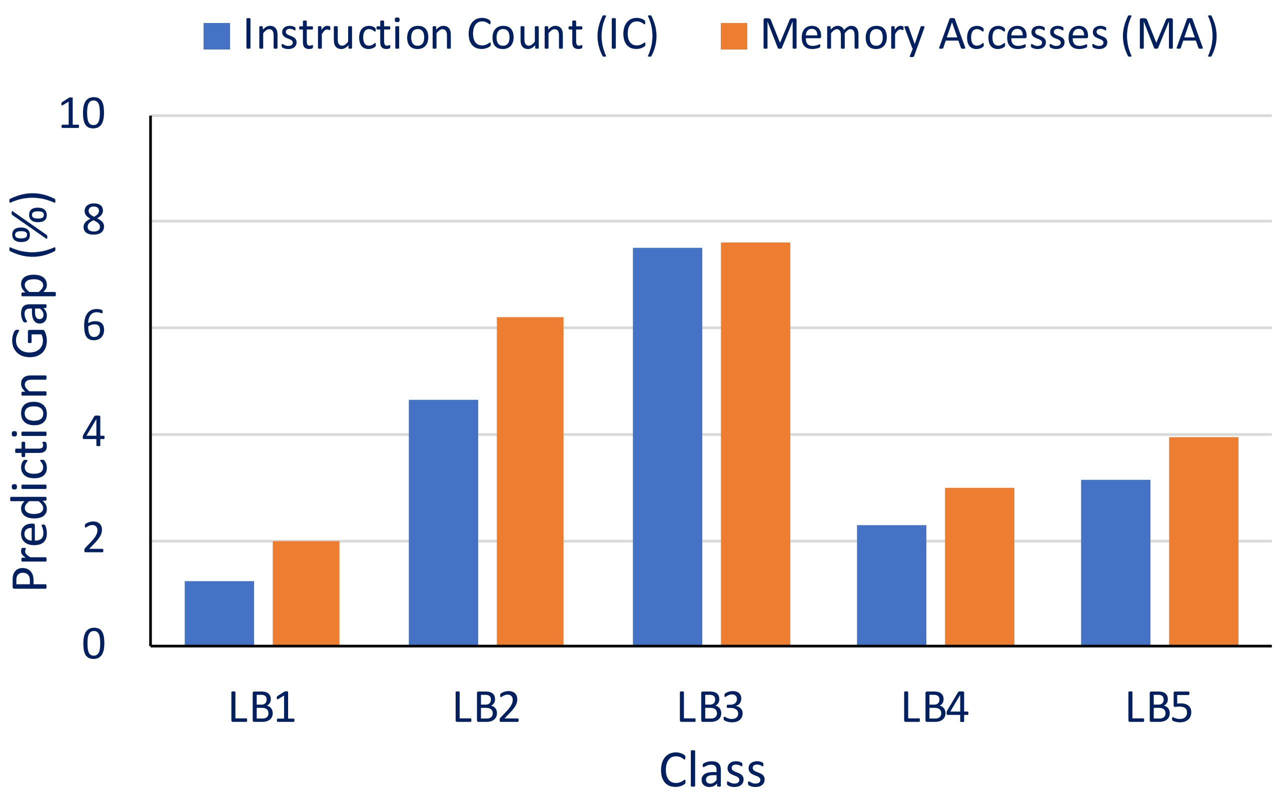


*S.Pirelli, A.Zaostrovnykh, G.Candea "A Formally Verified NAT Stack" KBNETS Workshop - SIGCOMM 2018

Predictions for Instruction Count, Memory Accesses

Results for Maglev-like Load Balancer

Class	Description
LB1	Unconstrained traffic
LB2	Client packet, new flow
LB3	Client packet, existing flow, unresponsive backend
LB4	Client packet, existing flow, existing backend
LB5	Heartbeat packets



Max prediction gap – 7.5% (IC) and 7.6% (MA)

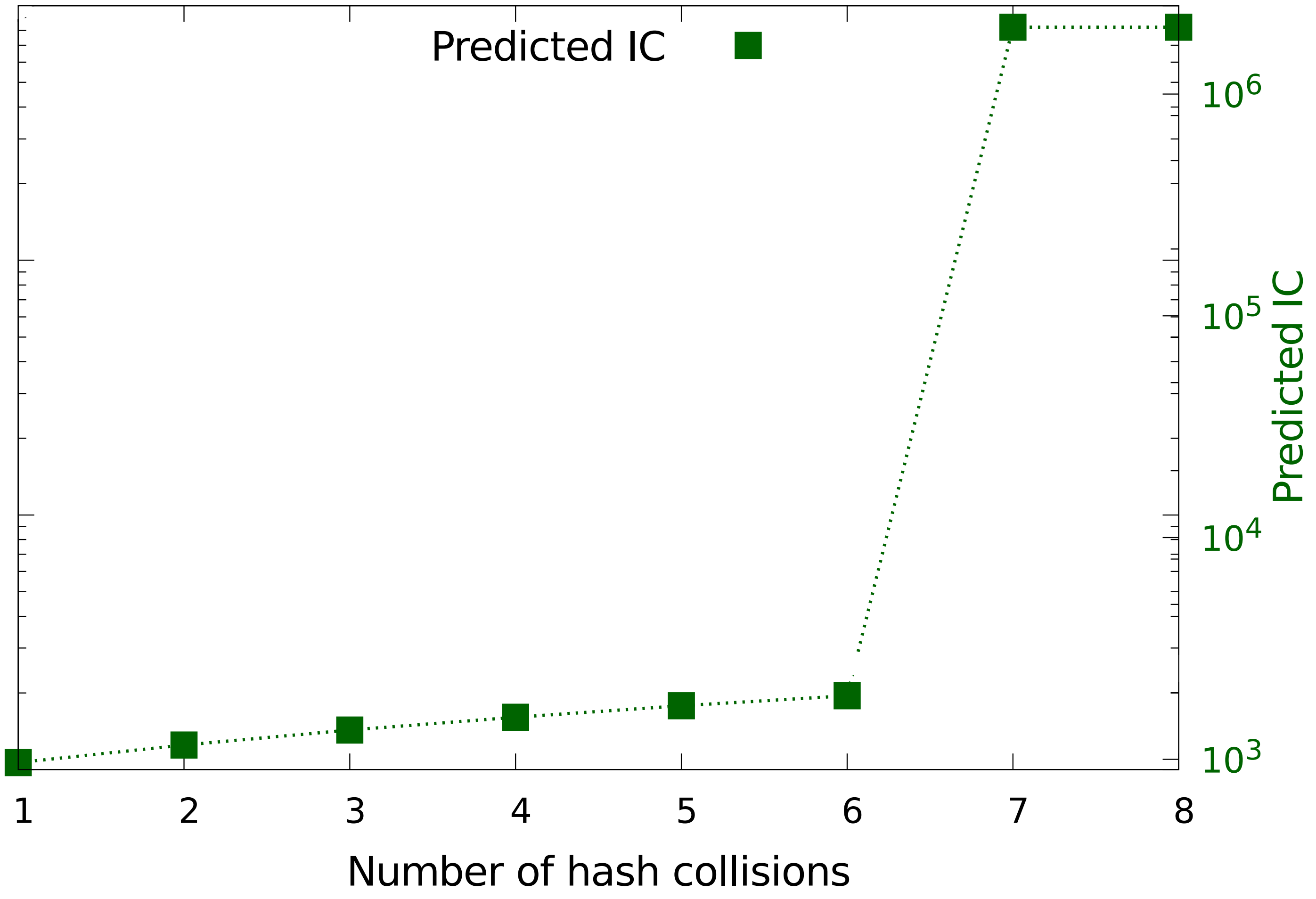
Why is there a prediction gap?

- Source 1: Trade-off between precision and legibility in PCVs
  - ❖ Can be overcome by exposing more detail
- Source 2: Differences between analyzed and production code
  - ❖ Disabled link time optimizations in analyzed code

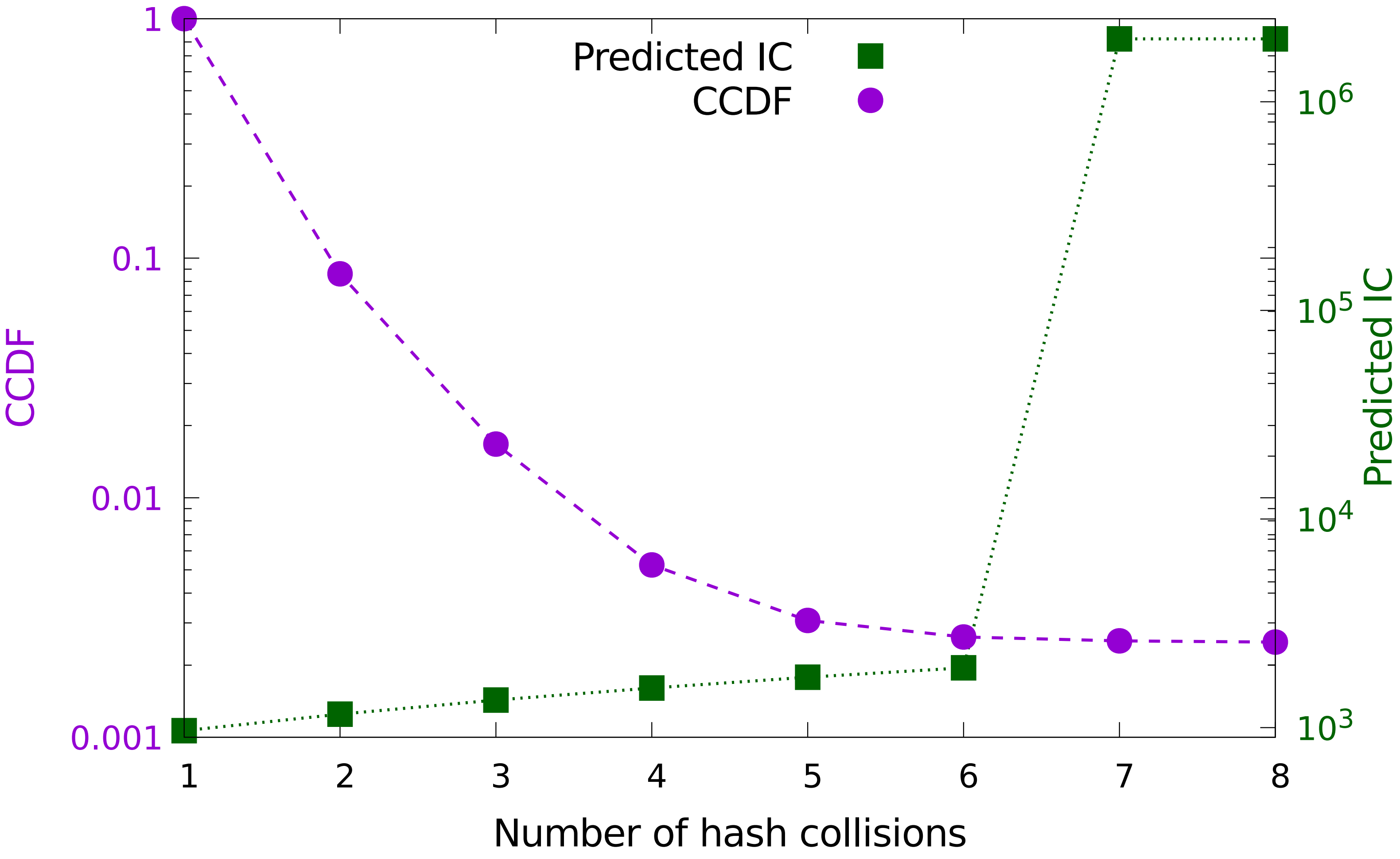
Use Case – Informed cost-benefit analysis

- Example: Bridge with randomized hash table
  - Incorporates random key into hash function
  - Rehashes all entries with a new key when collisions greater than a threshold
- Question: Where to place threshold?
  - Avoid rehashing under normal operation
  - Should rehash under attack

Use Case – Informed cost-benefit analysis



Use Case – Informed cost-benefit analysis



Bolt allows operators to visualize the consequences of their decisions

Performance Contracts for NFs

- Abstraction for users to parameterize arbitrary input workloads
- Predict performance for workload spec without running NF
- Performance predicted as function of **Performance Critical Variables (PCVs)**

bolt-perf-contracts.github.io

Backup Slides

- Distiller
- Results – IC, MA
- Results – NF chains
- Results – Latency
- Full Blown Contract

The Bolt Distiller

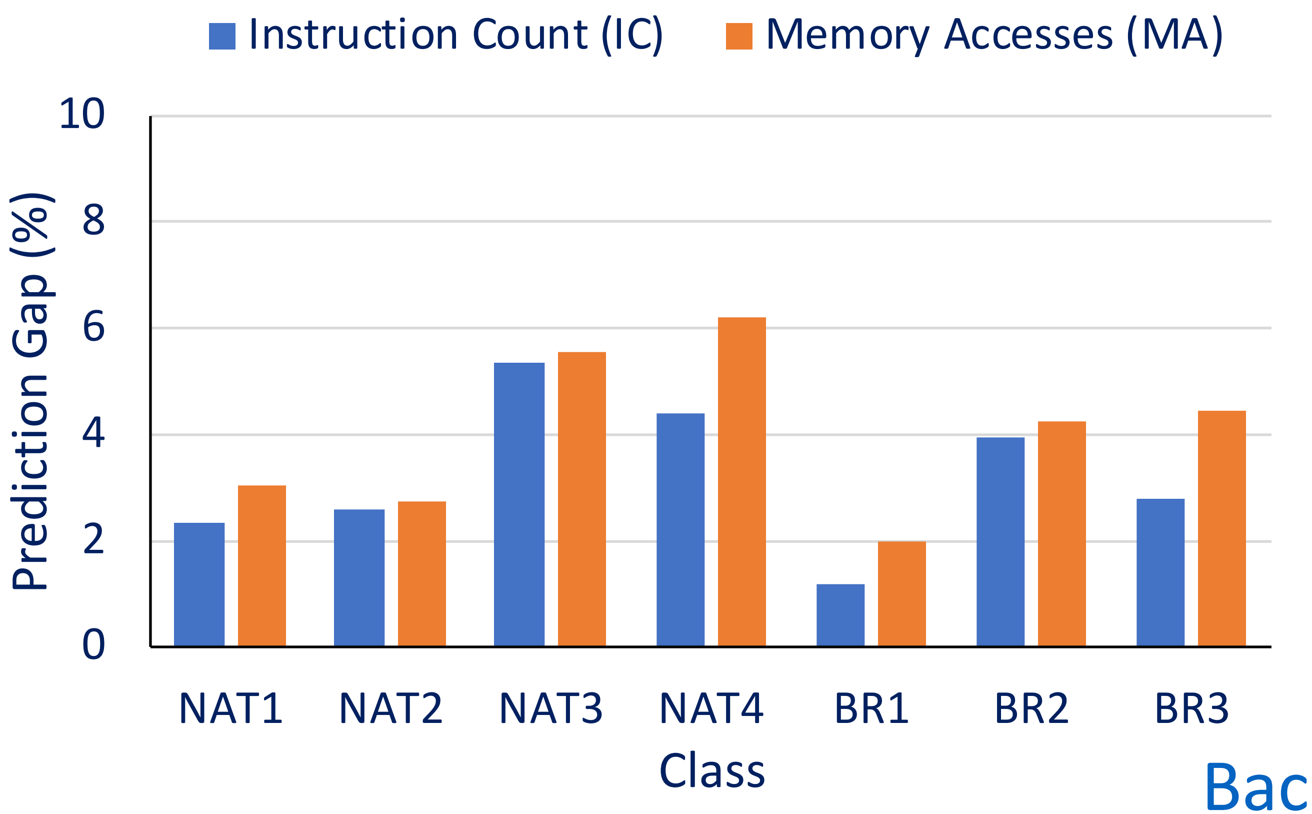
- Users need to know which traffic classes are likely
- Bolt is a static analysis tool, cannot know probabilities of each traffic class
- The Bolt Distiller
  - Input – A representative packet trace
  - Output - Execution path taken by each packet & values of PCVs
  - Users can then extrapolate the likelihood and query contract accordingly

Back

Predictions for Instruction Count, Memory Accesses

Results for NAT, Bridge

Class	Description
NAT1	Unconstrained traffic
NAT2	Client packet, new flow
NAT3	Existing flow
NAT4	External, dropped packet
BR1	Unconstrained traffic
BR2	Broadcast traffic
BR3	Unicast traffic



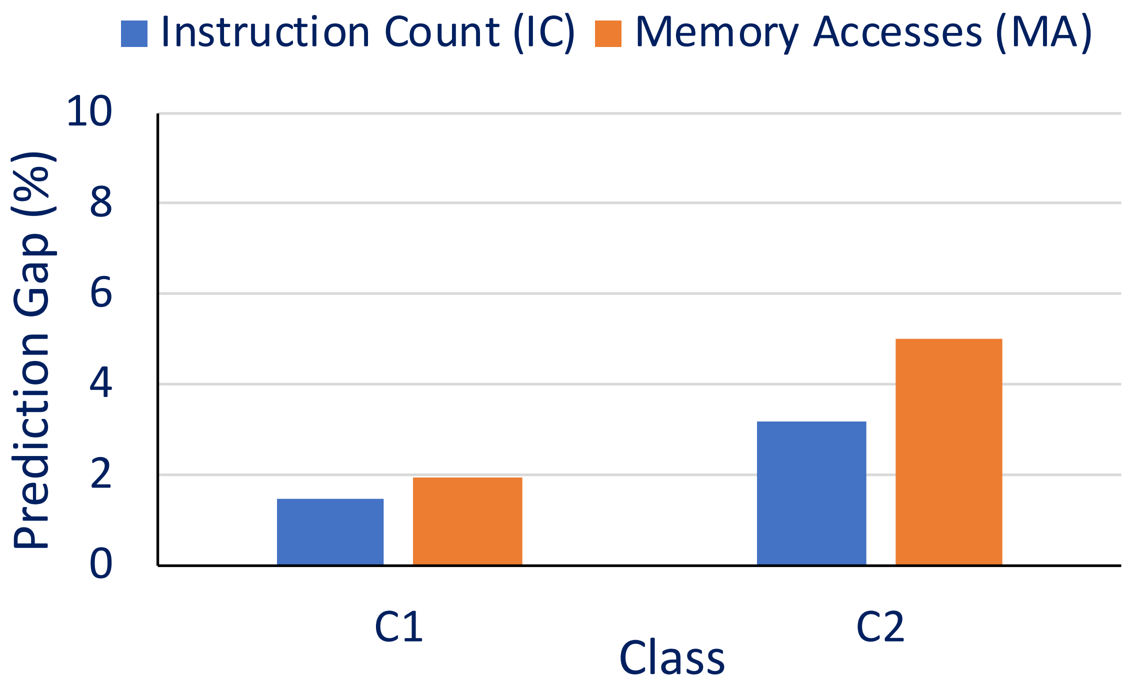
Back

Bolt predicts IC & MA accurately, irrespective of NF/Traffic Class

Predictions for NF chains

- NFs chained together
  - Firewall – drops packets with IP options
  - Router – Fast path (No IP options), Slow path (packets with IP options)

Class	Description
C1	Packets with IP options
C2	Packets without IP options

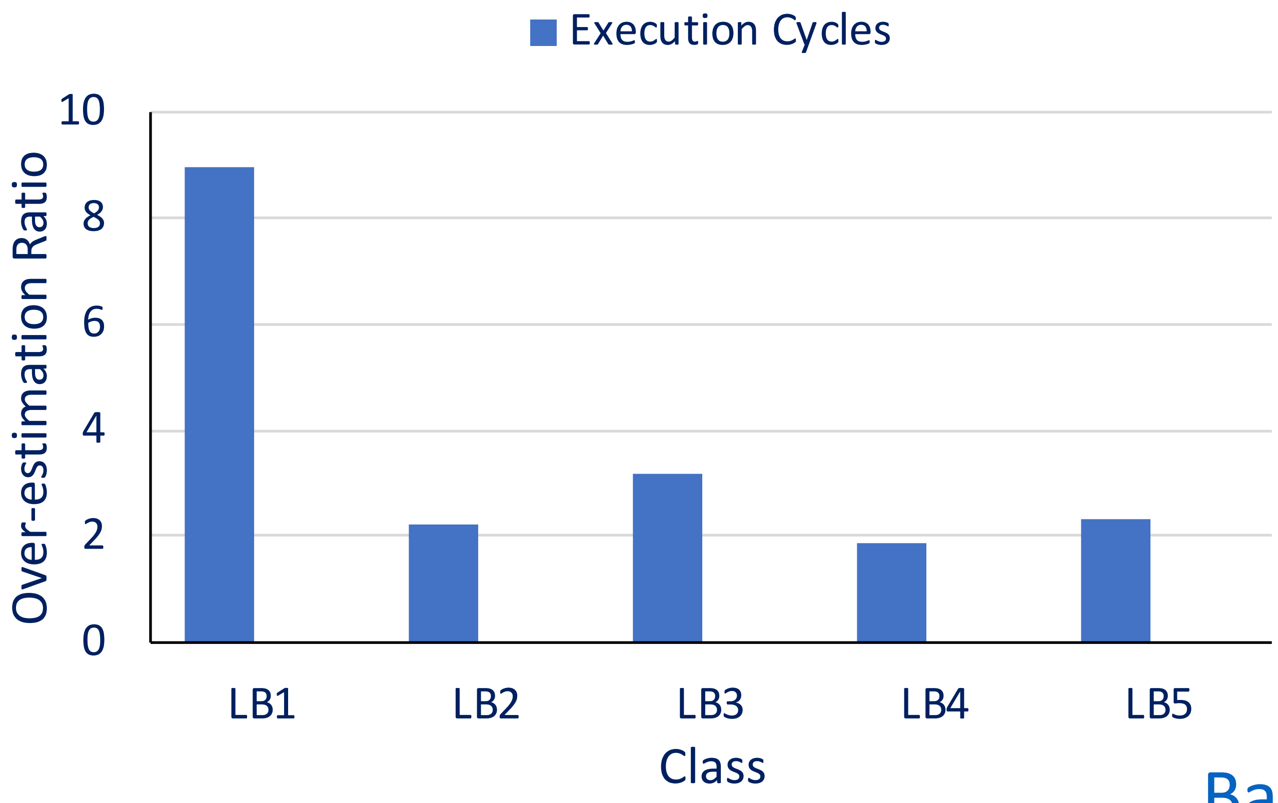


[Back](#)

Predictions for Latency (Execution Cycles)

Results for Maglev-like Load Balancer

Class	Description
LB1	Unconstrained traffic
LB2	Client packet, new flow
LB3	Client packet, existing flow, unresponsive backend
LB4	Client packet, existing flow, existing backend
LB5	Heartbeat packets



Back

9x for pathological traffic, 3x for typical traffic

Predictions for Execution Cycles

Results for LB,NAT, Bridge,LPM

NF+Class	Predicted Bound	Measured Cycles	Ratio
NAT1	591,948,908,371	65,217,699,390	9.08
NAT2	7,401	2,376	3.11
NAT3	5,142	1,789	2.87
NAT4	2,956	884	3.34
Br1	295,984,939,878	32,383,472,634	9.14
Br2	7,329	2,013	3.64
Br3	7,383	1,808	4.08
LB1	591,969,879,756	66,062,284,173	8.96
LB2	5,299	2,386	2.22
LB3	8,108	2,541	3.19
LB4	4,300	2,310	1.86
LB5	4,837	2,079	2.33
LPM1	1,419	967	1.46
LPM2	1,015	545	1.86

Table 3: Accuracy of execution cycle performance contracts for multiple NFs and packet classes.

Back

Full Blown Contract

Traffic Type	Instructions
Invalid packets (dropped)	$359 \cdot e + 80 \cdot e \cdot c + 38 \cdot e \cdot t + 425$
Known flows (forwarded)	$359 \cdot e + 30 \cdot c + 18 \cdot t + 80 \cdot e \cdot c + 38 \cdot e \cdot t + 1030$
New external flows (dropped)	$359 \cdot e + 30 \cdot c + 18 \cdot t + 80 \cdot e \cdot c + 38 \cdot e \cdot t + 528$
New internal flows; table full (dropped)	$359 \cdot e + 30 \cdot c + 18 \cdot t + 80 \cdot e \cdot c + 38 \cdot e \cdot t + 639$
New internal flows; table not full (forwarded)	$359 \cdot e + 30 \cdot c + 44 \cdot t + 80 \cdot e \cdot c + 38 \cdot e \cdot t + 1316$

Table 6: VigNAT performance contract. Instructions are described as a function of the number of expired flows (e) and the number of hash collisions (c) and bucket traversals (t) incurred in the hash table.

Back