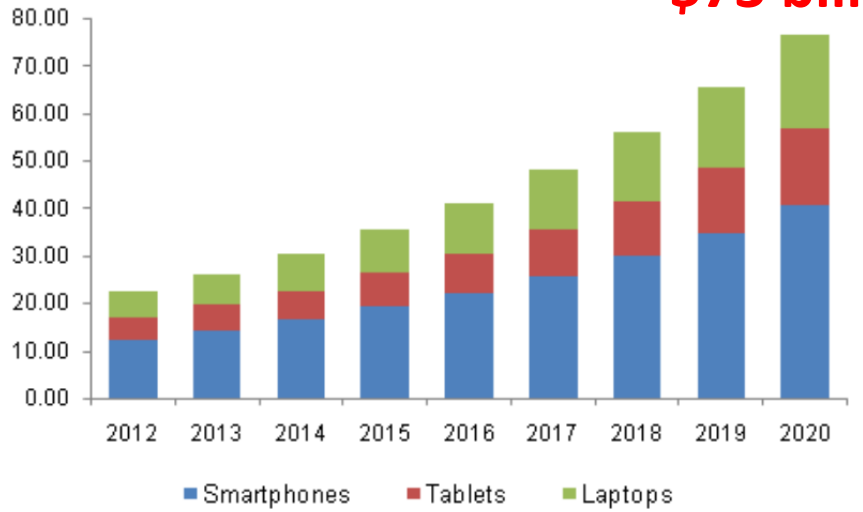
Programmable In-Network Security for Context-aware BYOD Policies

Qiao Kang

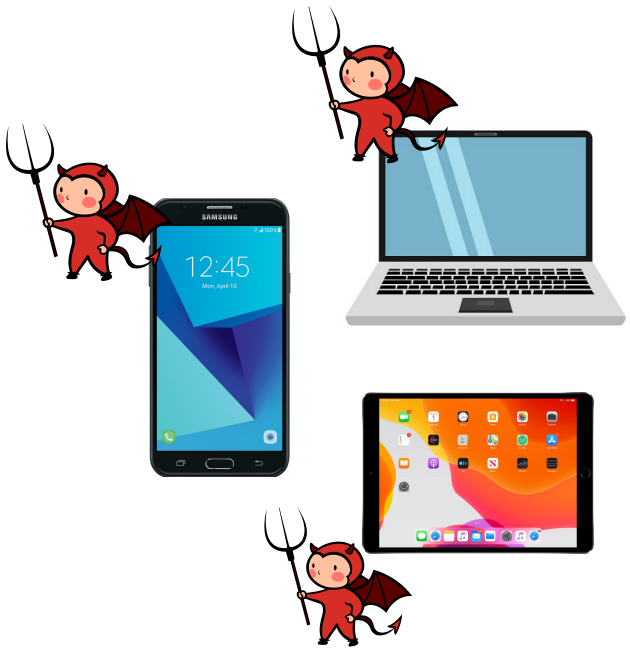
Rice University

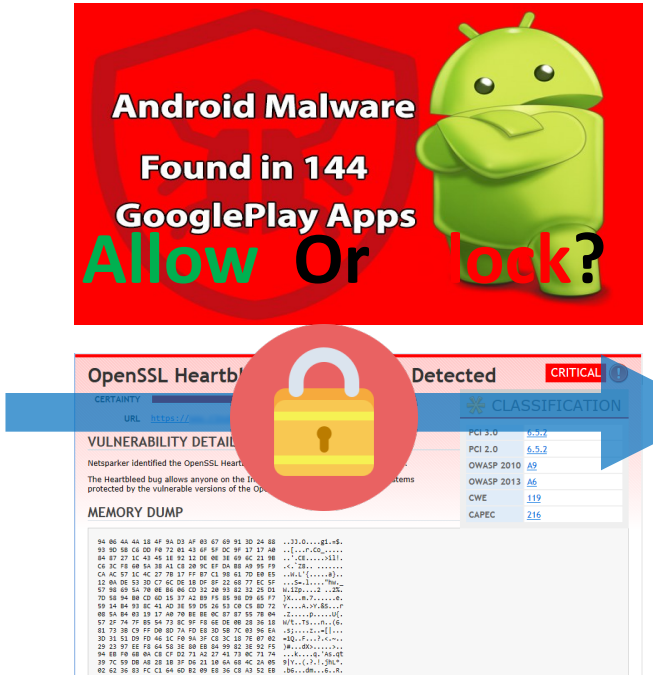
BYOD: Bring Your Own Device (to work)

$73 billion



Problem: Bring Your Own ~~Device~~ Risks





- BYOD devices are generally **less well managed** than their enterprise counterparts – easier to be compromised
- We need to enforce **access control** for BYOD clients

What policies does BYOD need?

Block access if *SSL version <= 6.5.2*

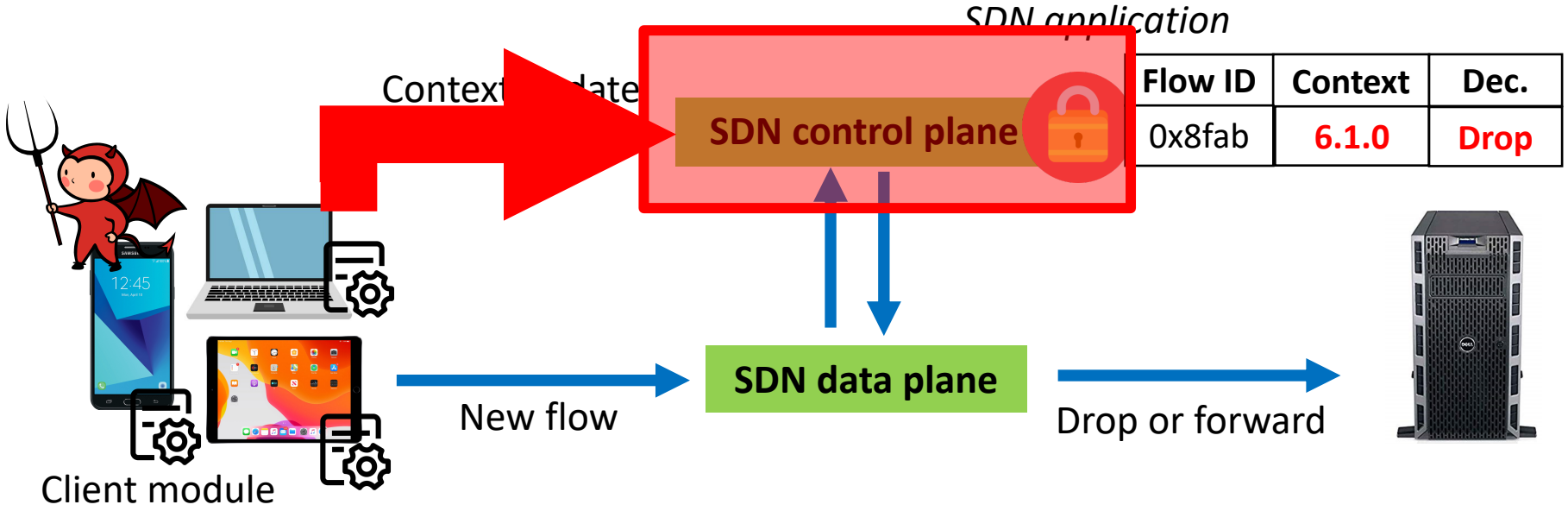
Block access if *administrator is offline*

Block access if *the recorder is on*

Block access if *client is outside of the company building*

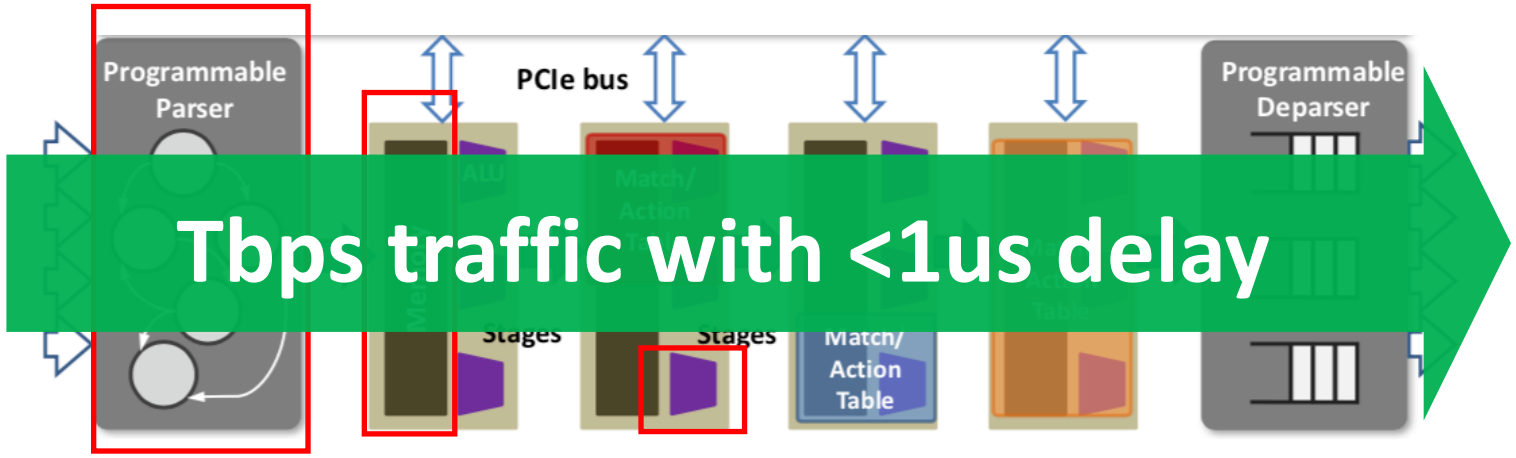
- **Context-aware policies**: More effective for BYOD than traditional access control
- Dynamic decisions based on “contexts”
- Challenge: How to enforce these policies?

What's is the state of the art?



- SDN-based BYOD policy enforcement [PBS – NDSS'16]
- Performance **bottleneck**: SDN control plane
- DoS attacks against the SDN control plane [AvantGuard - CCS'13]

Opportunity: Programmable data planes



P4 Language

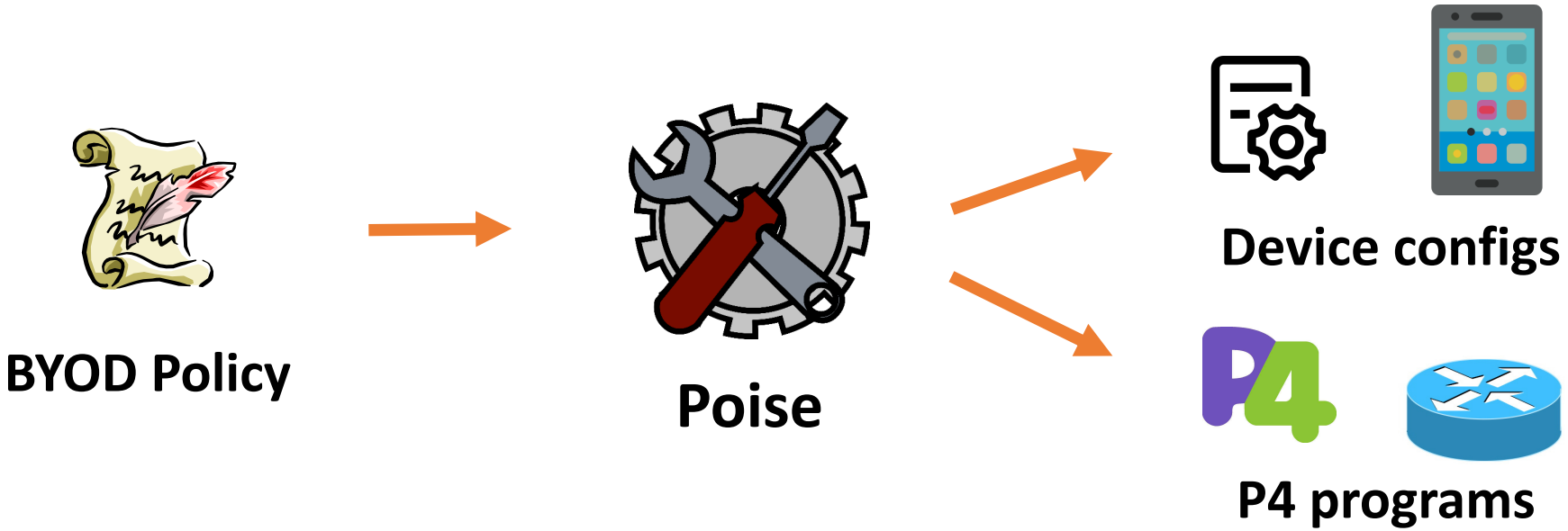
```
apply {
    if (hdr.ipv4.isValid() && hdr.ipv4.ttl > 0) {
        ecmp_group.apply();
        ecmp_nhop.apply();
    }
}
```

- New hardware features:
  - Programmable parser: Customized protocols
  - ALU: Arithmetic computations
  - Memory: Stateful processing
- High performance : <1us delay for Tbps traffic
- Programmable using the P4 language!

Research question

Can we leverage programmable switches to address the limitations of SDN-based solution?

Poise at 1000 feet



- **Language**: An expressive language for defining BYOD policies
- **Compiler**: Generates device configurations and switch programs
- **P4 data plane design**: A dynamic and efficient security primitive

Outline

- Motivation

- Poise Design
  - The Poise language
  - Compiling Poise policies
  - Data plane design
- Evaluation
- Conclusion

Outline

- Motivation
- Poise Design

  - The Poise language
  - Compiling Poise policies
  - Data plane design
- Evaluation
- Conclusion

The Poise language

Primitive Actions

A ::= drop | fwd(port) | flood | log

Expressions

E ::= v | $e_1 + e_2$ | $e_1 - e_2$ | $e_1 * e_2$ | M

Constant Lists

L ::= nil | v, L

Predicates

P ::= match($e_1 \circ e_2$) | match(h ∘ e) | match(h in l) | P&P | (P|P) | !P

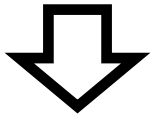
Monitors

M ::= count(P)

Policies

C ::= A | if P then C else C | (C|C)

Block access if *SSL version <= 6.5.2*



Predicate

```
if match (sslver <= 6.5)
then drop
```

Policy

Primitive Action

- An expressive language for writing context-aware policies
  - Predicates on customized client contexts
  - Support pre-defined primitive actions

Compiling Poise policies

```
if match (sslver <= 6.5)
then drop
```

```
header ctx_t {
  sslver: 16
}
```

```
table decision_tab
{
   key = {ctx.sslver: exact}

   entries = {
      <= 6.5.0: dec = DROP
      >  6.5.0: dec = ALLOW
   }
}
```

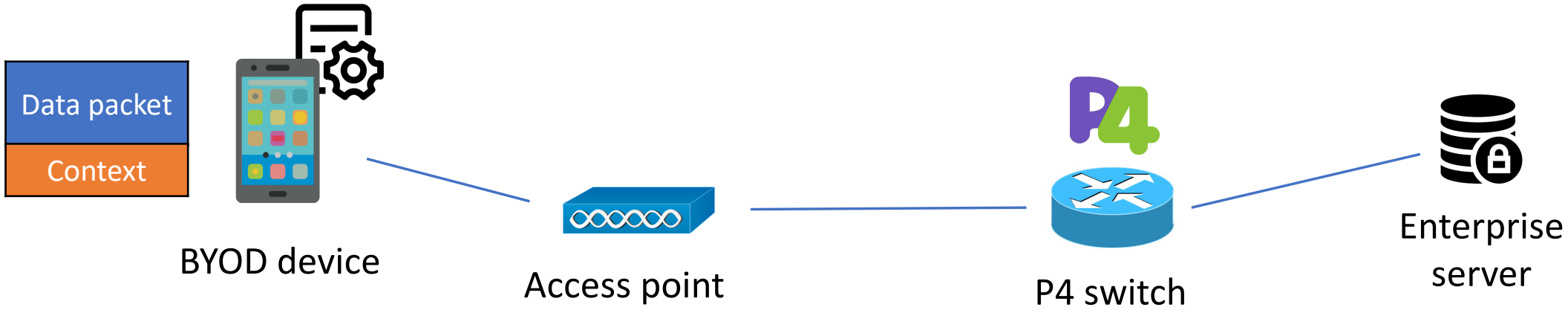
- Contexts (sslver) are compiled to customized header fields
- Security actions (if-else) are compiled to match/action table entries

Outline

- Motivation
- Poise Design
  - The Poise language
  - Compiling Poise policies

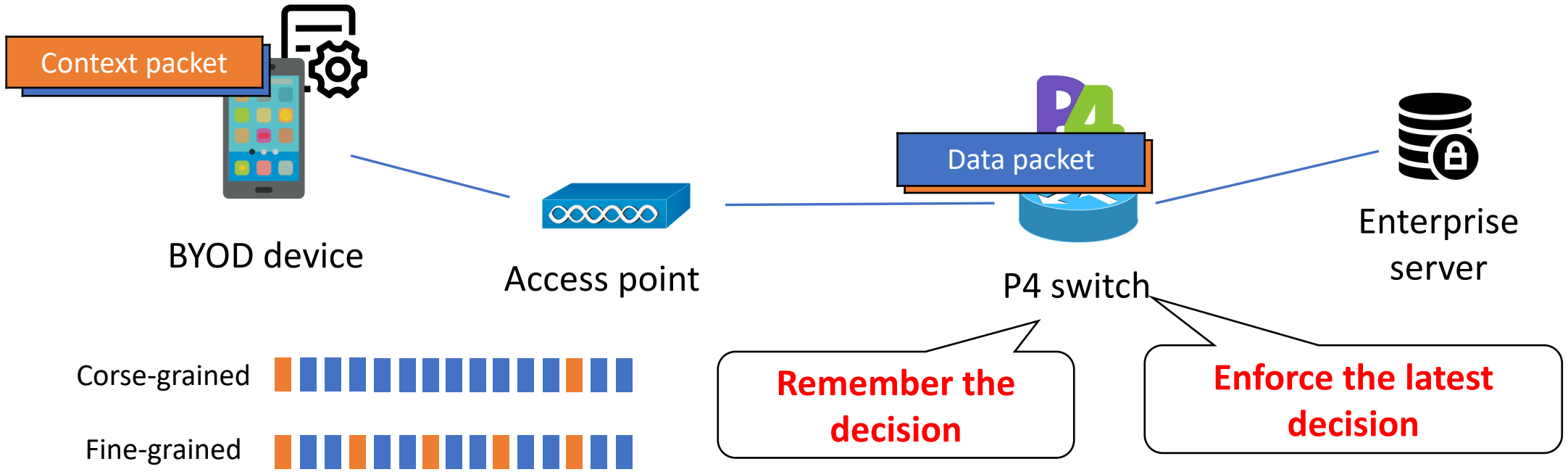
  - Data plane design
- Evaluation
- Conclusion

Starting basis: Tagging every packet



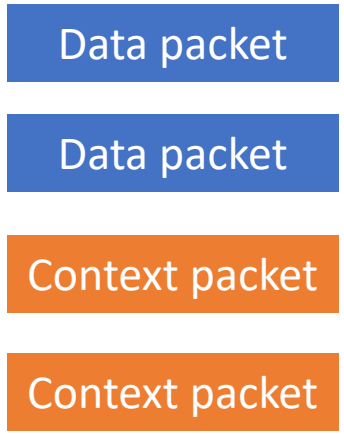
- Problem: Too much traffic overhead
  - Suppose 20 contexts each 4 bytes; average packet size is 500 bytes
  - 20×4/500 = **16% extra traffic**!
- Solution: Send context using dedicated “context packets” **occasionally**
  - Keep data packets **unmodified**

Poise runtime: A novel in-network primitive



- **Dynamic**: Decisions are based on latest context
- **Adjustable accuracy**: Users can tune the context packet period
- **Efficient**: Only context packets carry contexts; data packets unmodified

How to remember per-flow decision?

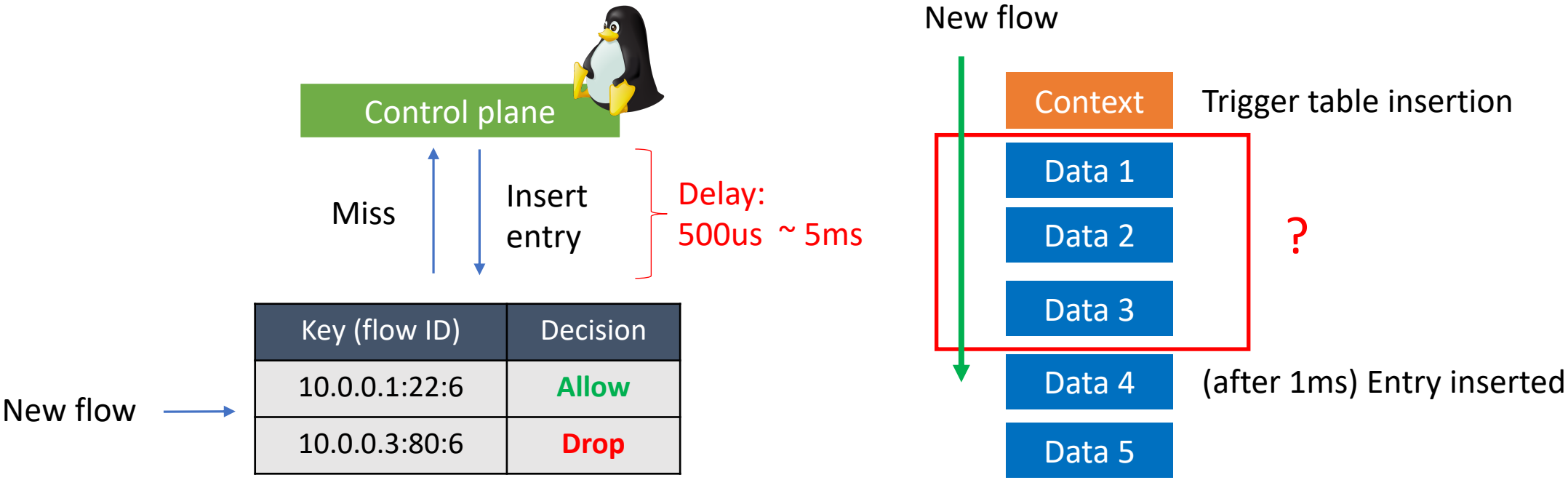


Flow table
(Match/Action table)

Key (flow ID)	Value (decision)
10.0.0.1:22:6	**Allow**
10.0.0.3:80:6	**Drop**
10.0.0.4:22:6	~~**Allow**~~ **Drop**

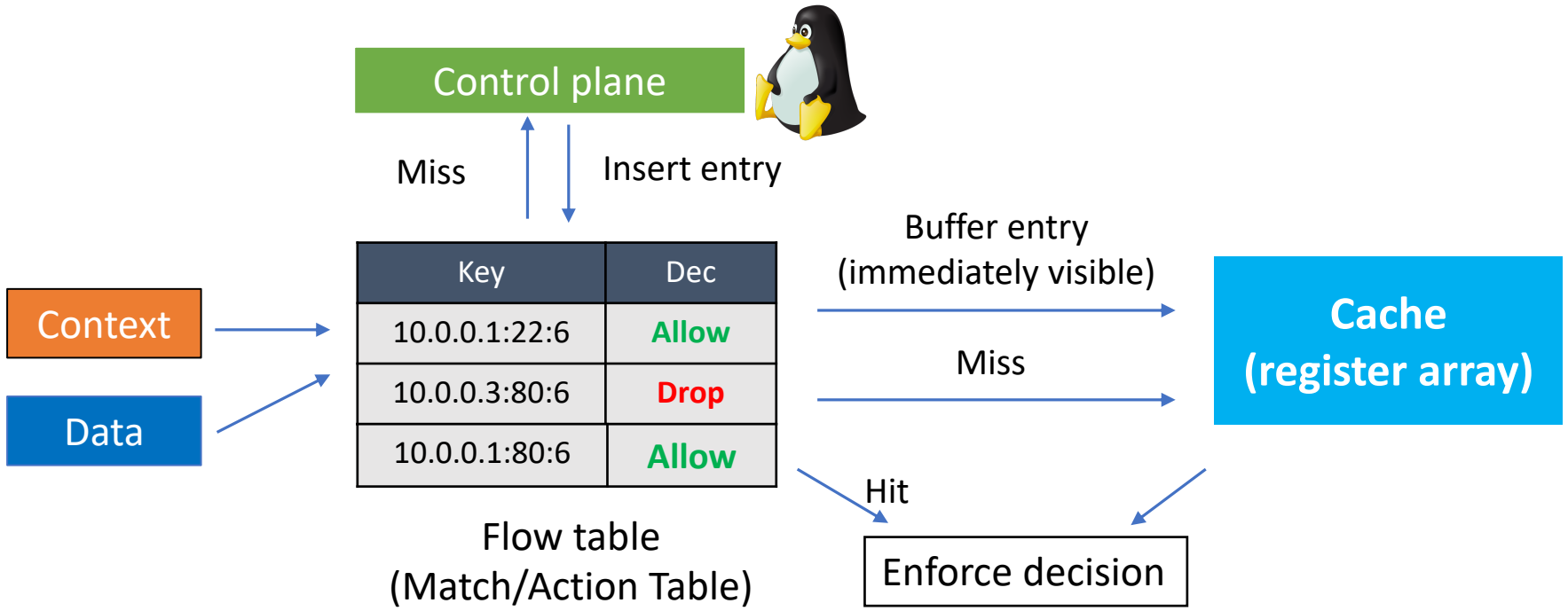
- Uses a Match/Action table to maintain the latest per-flow decision
  - Context packets: Update existing entries / insert new entries
  - Data packets: Look up the table to fetch decision

Challenge: New flow insertion delay



- Installation a new flow: Delay is on the order of milliseconds!
- We might have missed many of packets!

Solution: Buffering control plane updates



- Solution: Buffering updates in a **cache**
- Cache implemented in switch stateful registers: **Changes are in real time**.
- See our paper for more design details:
  - 1) Handling cache collisions, 2) mitigating DoS attacks to Flow Table and Cache.

Outline

- Motivation
- Poise Design
  - The Poise language
  - Compiling Poise policies
  - Data plane design
- Evaluation
- Conclusion

Evaluation setup

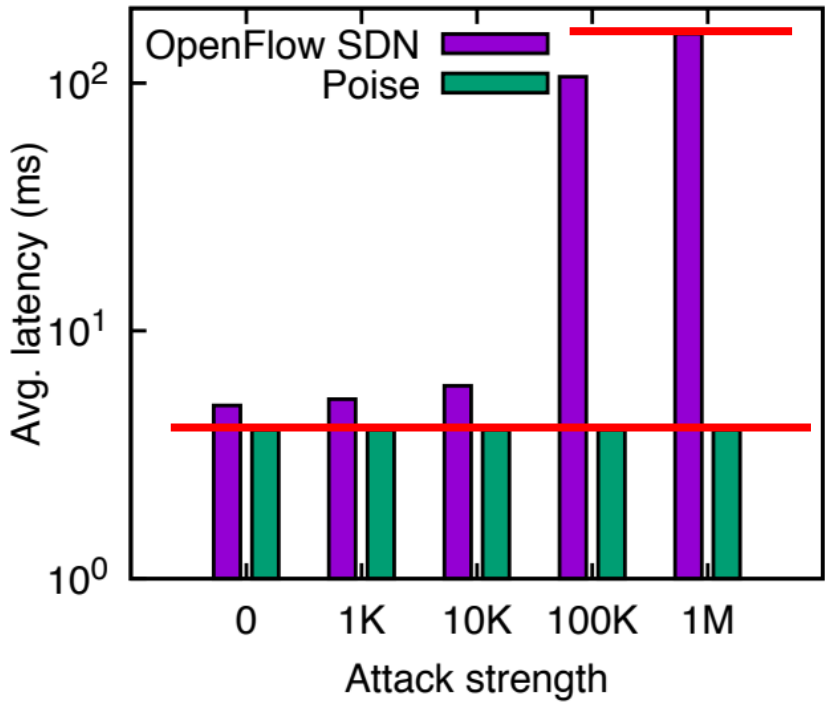
- Prototype implementation
  - Compiler: Bison + Flex
  - Android client module: a kernel module on Linux 3.18.31
  - ~6000 LoC
- Evaluation setup
  - Tofino Wedge 100BF switch 32 X 100 Gbps = 3.2 Tbps

What we have evaluated

- **Correctness**: Can Poise enforces BYOD policies correctly?
- **Overhead**: How much delay or throughput degradation that Poise incurs?
- **Scalability**: How complex/large policies can Poise support?
- ✓ **Poise vs. SDN**: Is Poise resilient to control plane saturation attacks?

- SDN-based solution: PBS – NDSS'16
  - Floodlight v1.2 + Open vSwitch v2.9.2
- Methodology:
  - DoS attacker: Launch frequent context changes
  - Measure how normal user traffic are affected

Poise vs. SDN: Packet delay

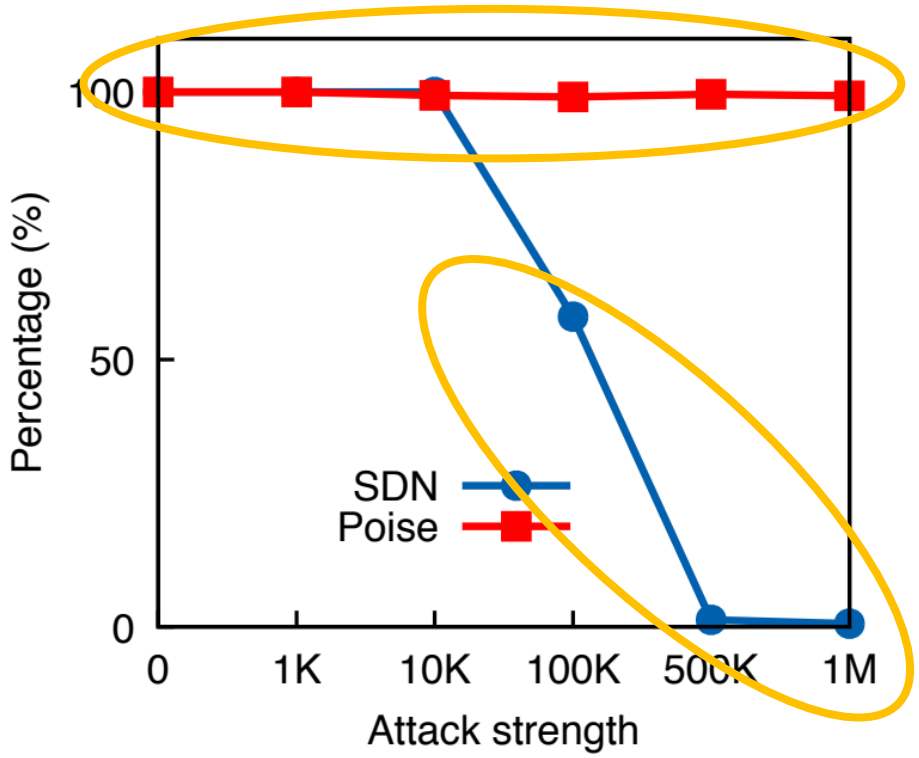


SDN: ~ 1s

Poise: 4ms

- SDN: Takes **~1 second** to process packets under heavy attacks
- Poise: Remain at a constant level

Poise vs. SDN: New flow installation



Poise: Install new flows normally

SDN: Cannot install new flows

- SDN: **Fails to install new flows** under heave attacks
- Poise: Almost always **installs 100% new flows**
- Poise is highly resilient to DoS attacks to the controller

Conclusion

- Motivation: Better network security with programmable switches
  - This talk focusses on the security application of enforcing BYOD policies
- We designed and implemented Poise
  - Leveraging P4 switches for enforcing security policies
- Poise transforms the hardware features to security benefits
  - Resilient to DoS attacks!

Thank You

Contact: qiaokang@rice.edu

Our full paper will appear at USENIX Security 2020