EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Communication Networks

Rust Barefoot Runtime (RBFRT): Fast Runtime Control for Intel Tofino

by Etienne Zink, Moritz Flüchter, Steffen Lindner, Fabian Ihle, and Michael Menth

http://kn.inf.uni-tuebingen.de

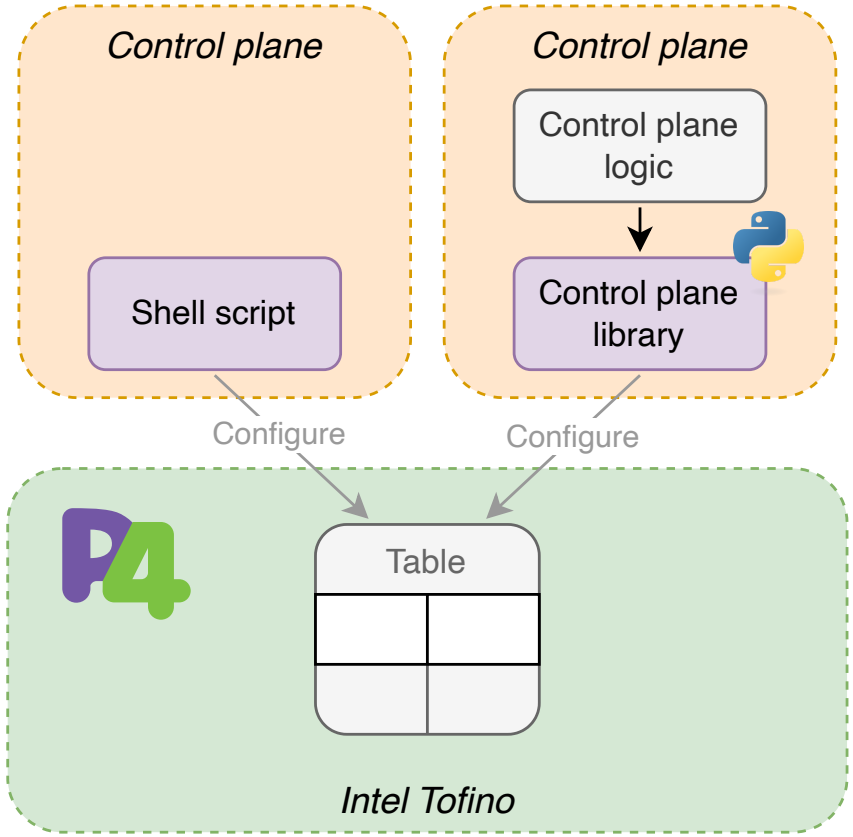
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Agenda

- Data Plane Configuration
- Rust Barefoot Runtime
- Evaluation
- Conclusion

Data Plane Configuration

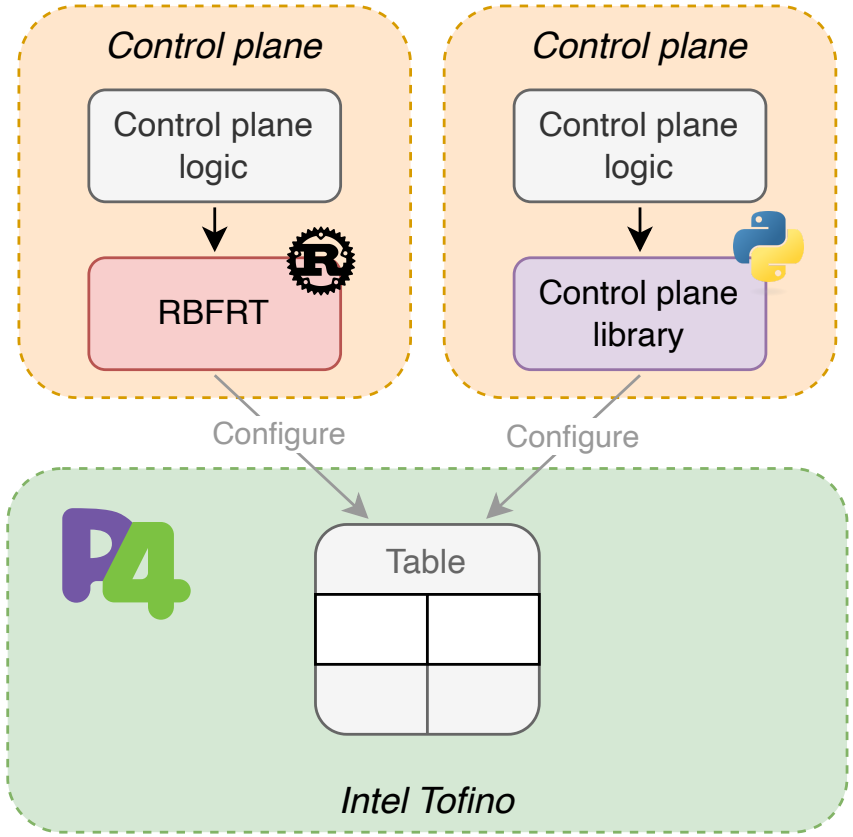
- Tofino's table(s) define packet processing behavior
  - Control plane inserts table entries
- Shell scripts
  - Manual effort
  - Does not react to events
- Control plane library
  - Shared library for different controllers
  - Provided by Barefoot
  - Python-based
  - Slow and not well documented
  - A single entry is inserted with one request
- Control plane logic
  - Custom for different controllers
  - Defines configuration logic, e.g.
    - Write table entries
    - Configure ports



Rust Barefoot Runtime – Overview

- Rust Barefoot Runtime (RBFRT)
  - Control plane library
    - Same features as Barefoot's Python library
  - Rust-based
  - Fast and memory-safe
  - Open-source on GitHub
    - https://github.com/uni-tue-kn/rbfrt
  - Documentation on GitHub Pages

- New batch configuration
  - Python library sends one request for each entry
  - RBFRT configures multiple entries in a single request for e.g.,
    - Device initialization
    - Synchronization



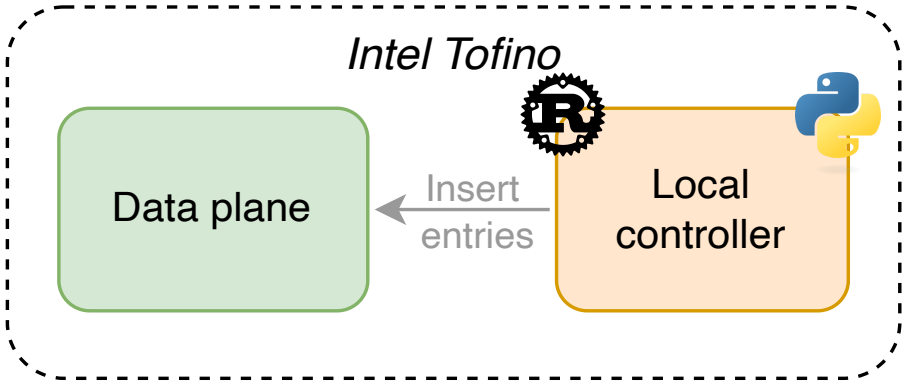
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

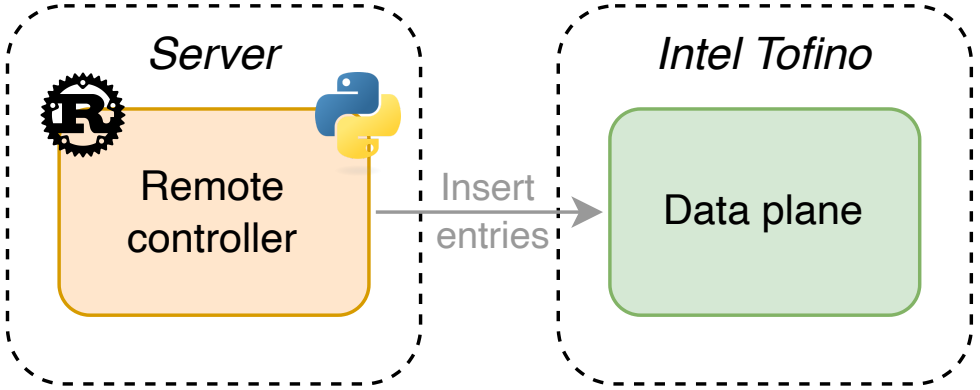
Rust Barefoot Runtime – Code Example

```rust
use rbfrt::{
    SwitchConnection,
    table::{self, MatchValue},
};

#[tokio::main]
async fn main() -> Result<(), Box<dyn std::error::Error>> {
    // Step 1
    // Connect to the switch
    let switch = SwitchConnection::builder("localhost", 50052)
        .device_id(0)
        .client_id(1)
        .p4_name("simple-switch")
        .connect()
        .await?;

    // Step 2
    // Create two table entries for simple IP forwarding
    let table_entries = vec![
        table::Request::new("ingress.ip_c.ip_forward")
            .match_key(
                "hdr.ipv4.dst_addr",
                MatchValue::lpm(vec![10u8, 0, 0, 1], 32),
            )
            .action("mark_to_forward")
            .action_data("e_port", 0),
        table::Request::new("ingress.ip_c.ip_forward")
            .match_key(
                "hdr.ipv4.dst_addr",
                MatchValue::lpm(vec![10u8, 0, 0, 2], 32),
            )
            .action("mark_to_forward")
            .action_data("e_port", 1),
    ];

    // Step 3
    // Insert table entries with batch configuration
    switch.write_table_entries(table_entries).await?;

    Ok(())
}
```

Evaluation – Setup

- Different control paradigms + control plane libraries
  - Local vs. remote controller
  - RBFRT vs. Barefoot's Python library
- Custom data plane with one table to configure
  - Insert 30.000 table entries
  - Different batch sizes (entries per request)
  - 100 runs
- Measurements
  - Insertion rate
    - Number entries inserted per second
  - Response time
    - Time to create, send, insert, and "acknowledge" entries

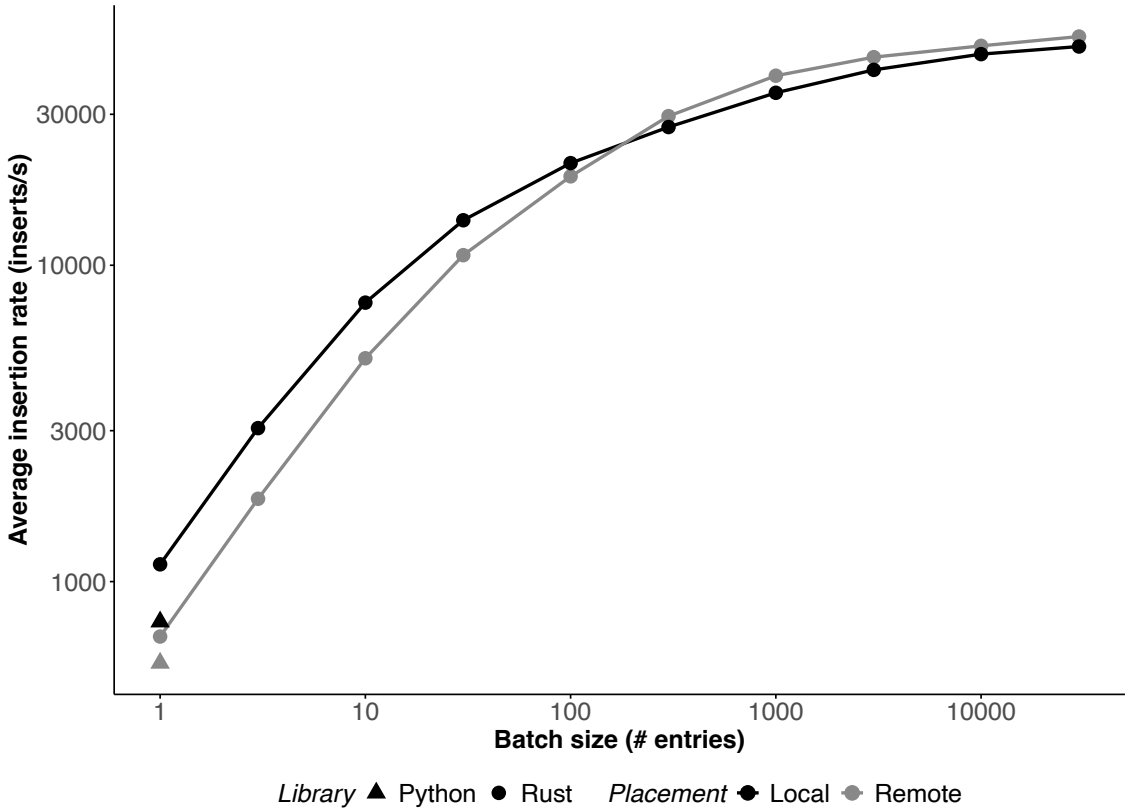




Evaluation – Average Insertion Rate

- Diagram
  - X-axis: Batch size
  - Y-axis: Average insertion rate

Batch size 1	Local	Remote
Python	746 entries/s	553 entries/s
RBFRT	1134 entries/s	671 entries/s

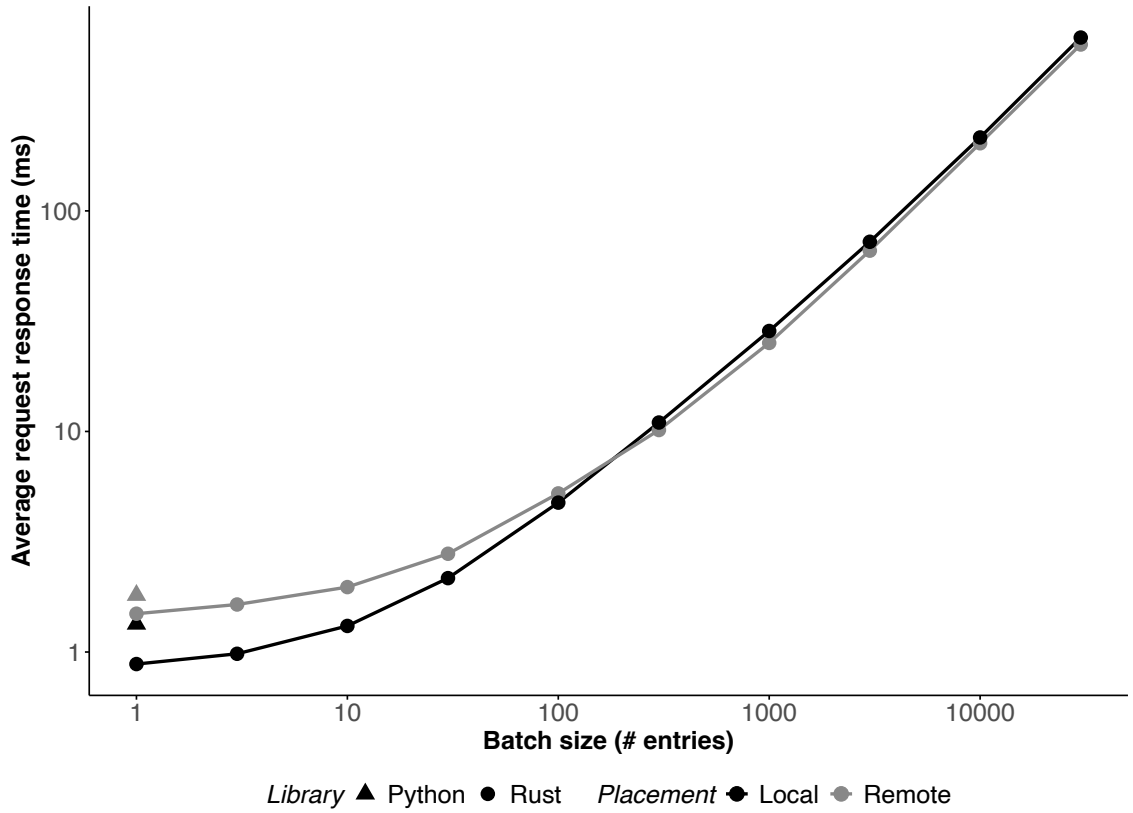


- Results
  - Larger batch sizes yield higher rates
  - Remote controller has a higher rate for batch sizes > 300
  - Similar rates for very large batch sizes

Evaluation – Average Request Response Time

- Diagram
  - X-axis: Batch size
  - Y-axis: Average request response time

Batch size 1	Local	Remote
Python	1.34 ms	1.8 ms
RBFRT	0.88 ms	1.49 ms



- Results
  - Difference between local and remote is significant for small batch sizes
  - Remote controller has a smaller response time for batch sizes > 300
  - Similar response times for large batch sizes

Conclusion

- Summary
  - RBFRT = Rust-based control plane library
  - Open-source
  - Same feature set as Barefoot's Python library
  - New batch configuration

- Evaluation
  - Shorter response time
    - Reduction by 34% for local and 17% for remote controller
  - Higher insertion rate
    - Increase by 48% for local and 21% for remote controller
  - Batch configuration further increases insertion rate
    - Increase by ~2820% for local and ~3460% for remote controller with batch size of 100 compared with Python

- Future work
  - Compare other configurations, not just inserting entries
  - Implement and compare batch configuration with Python

Thank you for your time!

DISCUSSION