Mantis: Reactive Programmable Switches

Liangcheng Yu, John Sonchack, Vincent Liu
Today’s Networks React

• A common task: **reacting** to current network conditions
  • Detecting failures and then **rerouting**
  • Identifying malicious flows and then **filtering**
  • Recognizing load imbalance and then **adjusting**

• In data centers, reactions need be fast
Today’s Primitives for Reaction

SDNs or conventional control loops
Flexible but slow

Built-in data plane primitives
Fast but restrictive

Programmable switches?
Constraints on operations in actions, number of stages, SRAM accesses, egress/ingress communication, in-band match-action updates…
Today’s Primitives for Reaction

Can we enable fine-grained reactions with minimum *latency* and maximum *flexibility*?

Built-in data plane primitives
*Fast but restrictive*

Programmable switches?
*Constraints on operations in actions, number of stages, SRAM accesses, egress/ingress communication, in-band match-action updates…*
**Approach**

Can we enable fine-grained reactions with minimum *latency* and maximum *flexibility*?

1. Push the reactions as close to the switch ASIC as possible

2. Co-design the data plane program for fine-grained malleability and ease of use
Mantis Overview

Usable, fast, and expressive in-network reactions on today’s RMT switches

Simple extension to P4

Generates code for dynamic reconfigurability/serializability

Switch

Mantis Control Plane

Reactions

Malleable P4 Program

Compiler

Arbitrary C code

10s of us
Abstraction

1. Malleable entities
   • Amenable to fine-grained reconfiguration at runtime

2. Reactions
   • Package reaction logic into a C-like function
Anatomy of Mantis

M1 Language
M2 Translation
M3 Isolation
M4 Execution
M1: Start with P4 Code

foo.p4

```
table my_table {
  reads { ipv4.dst : ternary; }
  actions { my_action; drop; }
}
action my_action() {
  modify_field(priority, 1);
}
```

How to make it run time reconfigurable?
M1: P4R Example

```p4
foo.p4r

table my_table {
    reads { ipv4.dst : ternary; }
    actions { my_action; drop; }
}
action my_action() {
    modify_field(priority, 1);
}
```
M1: P4R Example

foo.p4r

```p4
malleable value prio_var {
  width : 16; init : 1;
}

table my_table {
  reads { ipv4.dst : ternary; }
  actions { my_action; drop; }
}
action my_action() {
  modify_field(priority, ${prio_var});
}
```

Declaring malleable entities

Previous P4 code with references to malleable entities
M1: P4R Example

foo.p4r

malleable value prio_var {
    width : 16; init : 1;
}

table my_table {
    reads { ipv4.dst : ternary; }
    actions { my_action; drop; }
}

action my_action() {
    modify_field(priority, ${prio_var});
}

reaction my_reaction(reg re_qdepths[1:10]) {
    uint16_t cur_max = 0;
    for (int i = 1; i <= 10; ++i)
        if (re_qdepths[i] > cur_max) {
            cur_max = re_qdepths[i];
        }
    if (cur_max > THRESHOLD) {
        ${prio_var} = 5;
    }
}
reaction my_reaction(reg re_qdepths[1:10]) {
    uint16_t cur_max = 0;
    for (int i = 1; i <= 10; ++i)
        if (re_qdepths[i] > cur_max) {
            cur_max = re_qdepths[i];
        }
    if (cur_max > THRESHOLD) {
        prio_var = 5;
    }
}
Anatomy of Mantis

M1 Language
M2 Translation
M3 Isolation
M4 Execution
**M2: P4R Transformation**

```p4
foo.p4r

malleable value prio_var {
  width : 16; init : 1;
}

table my_table {
  reads { ipv4.dst : ternary; }
  actions { my_action; drop; }
}

action my_action() {
  modify_field(priority, ${prio_var});
}
```

Generalize user-specified knobs for “hitless” reconfiguration
M2: P4R Transformation

foo.p4r

malleable_value prio_var {
  width : 16; init : 1;
}

table my_table {
  reads { ipv4.dst : ternary; }
  actions { my_action; drop; }
}

action my_action() {
  modify_field(priority, ${prio_var}p4r_meta_.prio_var);
}

header_type p4r_meta_t_ {
  field {prio_var : 16;}
}

metadata p4r_meta_t_ p4r_meta_;

Replace the malleable value
M2: P4R Transformation

Foo.p4r

Malleable value prio_var {
    width: 16; init: 1;
}

Table my_table {
    reads { ipv4.dst : ternary; }
    actions { my_action; drop; }
}

Action my_action() {
    modify_field(priority, ${prio_var}, p4r_meta_.prio_var);
}

Header type p4r_meta_t_ {
    field {prio_var : 16;}
}

Metadata p4r_meta_t_ p4r_meta_;

Table p4r_init_ {
    actions {p4r_init_action_;}
    size: 1;
}

Action p4r_init_action_(prio_var) {
    modify_field(p4r_meta_.prio_var, prio_var);
}

Replace the malleable value

Multi-purpose initialization table
Anatomy of Mantis

- M1 Language
- M2 Translation
- M3 Isolation
- M4 Execution
M3: Isolation (ACID)

Isolation *matters*, consider

reaction my_reaction(reg src, reg dst) {}  

- Expectation: $src \leftarrow p_1, dst \leftarrow p_1$
- Without isolation: $src \leftarrow p_1, dst \leftarrow p_2$

Mantis enforces *per-pipeline, per-reaction* serializable isolation
**M3: Isolating Measurement**

For a register, **at most** one element will be updated on a packet thread. *Stale values* may appear in the current checkpoint for register arguments.

Timestamps $t_i$ appended to the duplicate buffer, $t_i > t_{i-1}$ is valid. Else, it is invalid.

Data plane: $((r_i, t_i))$

Control plane:

- $t_i > t_{i-1}$
- $t_i < t_{i-1}$
# M3: Isolating Updates

Three-phase updates for isolating fast, repeated, partial updates

<table>
<thead>
<tr>
<th>vv=0 (exact match)</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdr.a=0, vv=0</td>
<td>my_action(0)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=0, vv=1</td>
<td>my_action(0)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=1, vv=0</td>
<td>my_action(1)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=1, vv=1</td>
<td>my_action(1)</td>
<td></td>
</tr>
</tbody>
</table>

From previous mirror phase

<table>
<thead>
<tr>
<th>vv=0</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdr.a=0, vv=0</td>
<td>my_action(0)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=0, vv=1</td>
<td>my_action(0)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=1, vv=0</td>
<td>my_action(1)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=1, vv=1</td>
<td>my_action(2)</td>
<td></td>
</tr>
</tbody>
</table>

Prepare updates in vv=1 copy for malleable entities

<table>
<thead>
<tr>
<th>vv=1</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdr.a=0, vv=0</td>
<td>my_action(0)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=0, vv=1</td>
<td>my_action(0)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=1, vv=0</td>
<td>my_action(2)</td>
<td></td>
</tr>
<tr>
<td>hdr.a=1, vv=1</td>
<td>my_action(2)</td>
<td></td>
</tr>
</tbody>
</table>

Mirror the changes to the shadow copy for amortization

Bounded memory overhead and predictable latency
Anatomy of Mantis

M1 Language
M2 Translation
M3 Isolation
M4 Execution
M4: Mantis Control Plane

Traditionally data/control plane interactions are treated as *one-off, isolated* events, i.e., assumed to be “*on the slow path*”

Mantis control plane is instead *reaction-centric*

```python
helper_state = precompute_metadata();
memo = setup_cache(helper_state);
run_user_initialization(helper_state, memo);
while(!stopped) {
    updateTable(memo, "p4r_init_", {measure_ver : mv ^ 1});
    read_measurements(memo, mv); mv ^= 1;
    run_user_reaction(memo, helper_state, vv ^ 1);
    updateTable(memo, "p4r_init_", {config_ver : vv ^ 1});
    fill_shadow_tables(memo, vv); vv ^= 1;
}
```

~PCIe latency of the underlying system
Implementation and Evaluation

Prototype implementation on a Wedge100BF-32X Tofino switch

• P4R frontend: Flex/Bison based, ~5000 lines of C++ and grammar
• Mantis agent: dynamic (re)loading of user reaction (.so object)

https://github.com/eniac/Mantis
Demo
Prototype implementation on a Wedge100BF-32X Tofino switch
• P4R frontend: Flex/Bison based, ~5000 lines of C++ and grammar
• Mantis agent: dynamic (re)loading of user reaction (.so object)

Evaluation
• How fast is Mantis’s reaction time?
• What is the overhead?
• What are the applications of Mantis?
• How does Mantis compare to existing alternatives?
Mantis Achieves Fast Reaction Times

\[ F_b(1 \text{ tblMod}) + \sum_{a \in \text{args}} \left( F_a(a) \right) + C + \sum_{t \in \text{tblMods}} \left( 2F_b(t) \right) + 2F_b(N_{\text{init}} - 1) + F_b(1 \text{ tblMod}) \]

End-to-end reaction time: 10s of us
Mantis CPU Overhead

A dialogue loop occupies up to a single core but can be throttled

Overall, Mantis can co-exist with other functionalities
## Use Cases

<table>
<thead>
<tr>
<th></th>
<th>DoS mitigation</th>
<th>Route Recomputation</th>
<th>Hash polarization mitigation</th>
<th>Reinforcement Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Measurement</strong></td>
<td>Flow signature, packet count</td>
<td>Heartbeat counts, timestamp</td>
<td>Queue depths of ECMP ports</td>
<td>Packet counts and queue depths</td>
</tr>
<tr>
<td><strong>Control logic</strong></td>
<td>Block the sender if the estimated flow size exceeds a threshold</td>
<td>Mark the failed link if received heartbeat number is small than expected after consecutive K confirmations</td>
<td>Change ECMP hashing input to another permutation if found a persistent imbalance of port utilization</td>
<td>Use a Q-learning algorithm to calculate the optimal ECN threshold based on rewards</td>
</tr>
<tr>
<td><strong>Reconfiguration</strong></td>
<td>Drop the malicious traffic for the blocked senders</td>
<td>Reroute traffic towards the affected link</td>
<td>Reconfigure the malleable fields for another 5-tuple permutation</td>
<td>Change ECN malleable value</td>
</tr>
</tbody>
</table>
Flow Size Estimation

• Evaluation setting
  • CAIDA traces, 20s chunk, 10Gbps link of ISP backbone
• Arguments
  • packet source IP and packet counter
• Algorithm
  • Estimation formula \( \frac{\hat{f}_t - \hat{f}_{t_0}}{t - t_0} \)
  • \( t_0 \): timestamp when first observe the flow
• Mantis sampling rate: every 10us, ~1 in 5 packets
Summary

• Fine-grained reaction to network statistics as first class citizen
• P4R interface to simplify the encoding of serializable reaction
• Generic support of sub-RTT reactive behaviors

Mantis can be used for…
• Encoding flexible control logic
• Workarounds of current limitations
• Reducing memory overhead via offloading
• Data/control plane co-design

https://github.com/eniac/Mantis

Thank you for your attention!
Live Q&A