Isolation Mechanisms for High-Speed Packet-Processing Pipelines

Tao Wang\textsuperscript{1}, Xiangrui Yang\textsuperscript{2, 3}, Gianni Antichi\textsuperscript{3}, Anirudh Sivaraman\textsuperscript{1}, and Aurojit Panda\textsuperscript{1}
Data-plane programmability is mainstream

• Our focus: pipeline-based programmable networking devices with a Reconfigurable Match Table (RMT)-style architecture

Intel Mount Evans IPU  Fungible DPU  Pensando SmartNICs  Intel Barefoot Tofino Switches
Can improve performance for many systems

NetCache: Balancing Key-Value Stores with Fast In-Network Caching

Xin Jin¹, Xiaozhou Li², Haoyu Zhang³, Robert Soulé²,⁴, Jeongkeun Lee², Nate Foster²,⁵, Changhoon Kim², Ion Stoica⁶

Eris: Coordination-Free Consistent Transactions Using In-Network Concurrency Control

Jialin Li
University of Washington
lijl@cs.washington.edu

Ellis Michael
University of Washington
emichael@cs.washington.edu

Dan R. K. Ports
University of Washington
drkp@cs.washington.edu

HPCC: High Precision Congestion Control

Yuliang Li⁷, Rui Miao⁸, Hongqiang Harry Liu⁹, Yan Zhuang⁹, Fei Feng⁹, Lingbo Tang⁹, Zheng Cao⁹, Ming Zhang⁹, Frank Kelly⁷, Mohammad Alizadeh⁹, Minlan Yu⁷

Alibaba Group⁹, Harvard University⁷, University of Cambridge⁷, Massachusetts Institute of Technology⁷
But most systems are now deployed on shared cloud infrastructure
Can **multiple** tenant programs share **one** programmable network device?

Not yet! Lack **isolation mechanisms**
What is isolation?

• Common requirements
  • One program cannot impact another’s behavior or performance
  • Reconfiguring a program does not disrupt others
  • Maintain aggregate device throughput and latency

• RMT-specific requirement: expressivity
  • does not limit complexity of programs that can be executed
Why do existing works not suffice?

• Key challenge: **expressivity**

• Some approaches (e.g., NetVRM) focus on memory isolation

• Others partition processing resources between programs
  • Reduced stages and ALUs limit complexity of programs

• **Our goal**: Isolation should not limit what can run on RMT
How to implement isolation in RMT?

• **Time sharing**: mechanisms to switch between programs
  • Problem: context switching is too costly at high speeds

• **Space sharing**: split processing units between programs
  • Problem: limit expressivity by reducing number of instruction units

• Need a new approach: **Menshen**
Menshen’s high-level approach

• Software mediation
  • Resource allocation
  • Compilation

• New hardware primitives enable
  • Switching between programs at packet timescales
  • Disruption-free reconfiguration
Menshen’s high-level approach

• Software mediation
  • Resource allocation
  • Compilation

• New hardware primitives enable
  • Switching between programs at packet timescales
  • Disruption-free reconfiguration
Menshen’s Hardware Primitives:
program ID + overlays + space partitioning
Menshen program IDs

• Menshen associates a PID with each tenant program

• Every packet specifies the program ID (PID)
  • In our current implementation: VLAN tag

Packet format

... PID ... tenant-defined headers ... tenant-defined payload
Menshen uses PID to select program
Review: How a RMT stage works

Packet

Parser

...       PID

PHV

Input PHV

Stage n

Key Extractor

Table Key

Match-Action

Action

Apply

Output PHV

Match Table

Stage n+1
Menshen: How overlays work
Key extraction enhancements
Menshen: How **space partitioning** works

**Match-action enhancement**

<table>
<thead>
<tr>
<th>Table Key</th>
<th>Match-action Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHVs</td>
<td>key entry 0</td>
</tr>
<tr>
<td></td>
<td>key entry 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>PID</td>
<td>key entry 0</td>
</tr>
<tr>
<td></td>
<td>key entry 1</td>
</tr>
<tr>
<td></td>
<td>key entry n</td>
</tr>
<tr>
<td></td>
<td>action entry 0</td>
</tr>
<tr>
<td></td>
<td>action entry 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>action entry n_g</td>
</tr>
</tbody>
</table>

action entry n_g
Menshen: Action engine enhancements

action entry \( n_g \)

PHVs

PID

Crossbar

ALU

ALU

Segment Table

stateful memory offset & range

stateful memory offset & range

stateful memory offset & range

PHVs

PID

virtual addr. ➞ physical addr.

Stateful memory
Menshen’s design principles:
overlays + space partitioning

• Tables and memory are **partitioned** between programs
  • Large enough, negligible impact on expressivity

• Use **overlays** to share stages and actions between programs
  • Does not limit program length or complexity

• Configuration tables are in registers
  • Allows switching programs at per-packet timescales
Implementation

• We have an open-source implementation
  • Software extends P4 reference compiler
  • FPGA prototype integrated with NetFPGA and Corundum
• Also includes an open-source RMT pipeline implementation
  • Simplifies evaluation of proposed RMT extensions
• Menshen also has ASIC synthesis results
  • works at 1 GHz with FreePDK45nm library
Evaluation

• Does Menshen achieve isolation requirements?
  1. Can it support multiple programs while ensuring correctness?
  2. Does it hurt aggregate pipeline performance?
  3. Is it disruption-free?
1. Can Menshen support multiple programs?

- 6 P4 tutorial programs
  - CALC, Firewall, Load balancing
  - QoS, Source routing, multicast
- + simplified NetChain and NetCache

- Tested program correctness using packet traces ✅
2. Menshen’s overall throughput performance

• Menshen’s pipeline throughput with varying packet sizes
  • setup: 100Gbps Spirent testbed with Corundum, CALC program loaded
3. Does reconfiguration disrupt other programs?

- 3 CALC programs send traffic with 5:3:2 ratio on 10Gbps link
- setup: NetFPGA testbed, netmap + tcpreplay

![Graph showing throughput over time for three programs.](image)
Conclusion --- the quest for isolation

• The lack of isolation is a barrier to the adoption of programmable networks

• Menshen: a proposal for how to provide isolation for RMT
  • Core idea: redesign how time and space sharing work in RMT

• Menshen: publicly available for use and extension

https://isolation.quest
tao.wang@nyu.edu