An Intermediate Representation for Network Programming Languages

Hao Li¹, Peng Zhang¹, Guangda Sun¹, Chengchen Hu², Danfeng Shan¹, Tian Pan³, Qiang Fu⁴
Programming the (Software-Defined) Networks

Network Programming Languages (NPL)

- Network-wide
- Domain-specific languages (DSL)
- High-level abstractions

[\texttt{x: (ip.src=ip1) \rightarrow .*B.*}, \texttt{\min(x,100MB/s)}]

Controller APIs

- Network-wide
- General-purpose language
- Self-defined abstractions

Network Control Plane

Network Data Plane (NDP)
Multiple NPLs and NDP interfaces Coexist

…and will Coexist in the Near Future

- Features of NPLs/NDPs are Different
- High Risk for a Unified NPL/NDP

- Features of NPLs/NDPs are Different
  - NPL/NDP is frequently updating
  - Deploying new devices
  - Training operators

- High Risk for a Unified NPL/NDP
  - NPL/NDP is frequently updating
  - Deploying new devices
  - Training operators

Routing with Waypoint

Stateful Firewall

Path Control

Stateful Abstractions
...and will Coexist in the Near Future

- Features of NPLs/NDPs are Different
  1. Deploy cross-NPL programs
  2. Port programs into heterogeneous NDPs

- High Risk for a Unified NPL/NDP
  - NPL/NDP is frequently updating
  - Deploying new devices
  - Training operators

Routing with Waypoint
Stateful Firewall
Path Control
Stateful Abstractions
Monolithic Compilation

Problem 1: Cross-NPL programs cannot interoperate

- Rule Conflict
  - ≠ Intent Conflict
  - Cannot be resolved without Intent
Monolithic Compilation

Problem 2: NPL/NDPs cannot independently evolve

- $M$ NPLs and $N$ NDPS
- $M \times N$ Full Compilers
Analog to Compiler for PC Languages

- **C++ Front End**
- **JAVA Front End**
- **Python Front End**

- **x86 Back End**
- **ARM Back End**

**LLVM Intermediate Representation (IR)**

**LLVM Optimizer**

- **C++ Instructions**
- **x86 Instructions**
- **ARM Instructions**
1. Cross-NPL programs can be composed at IR level

2. \( M \) Front Ends + \( N \) Back Ends
What is a Good IR for NPL Compiler?

• Expressive
• Composable
Design Philosophy of IR

• **Path Control with Waypoints**
  - Waypoint B
  - *Regular expression: .*B.*

• **Path Control with Resource**
  - Routing with 100MB/s bandwidth
  - *Consume the resource through forwarding*

• **Stateful Operations in Switches**
  - With SSH packets counted
  - *Check and update network variables through forwarding*
Network Transaction Automaton (NTA)

- Network Transaction
  - Three tuple: \( h \quad r \quad d \)
  - **\( h \)**: next hop to be forwarded
  - **\( r \)**: the consumption of the network resources
  - **\( d \)**: \( <g, u> \) that checks and updates the network variables

- Network Transaction Automaton
  - Each transition is a network transaction
  - The language of all "legal" sequences of network transaction
• Can NTA Express Most Semantics from NPLs?
• Can NTAs be Composed without Semantics Loss?
• Can NTAs be Efficiently Compiled into NDP Configurations?
Expressiveness of NTA

Merlin policy: Waypoint + Bandwidth

\[ \text{if srcip}=i_1 \& \text{dstip}=i_2 \text{ then } \text{num}_\text{pkt} += 1 \]
else id

SNAP Program: Packets Counting

\[ r_1: \text{bw}[h'][h]<100\text{MB/s} \]

\[ u_1: \text{num}_\text{pkt}+=1 \]
Expressiveness of NTA

Pyretic
(match(dstip='10.0.0.1') >> fwd(6))
route traffic with dstip 10.0.0.1 to virtual port 6

NetKAT
(if (dstip='10.0.0.1') then pt←6)
route traffic with dstip 10.0.0.1 to virtual port 6

Flowlog
ON packet_in(p) WHERE p.nwPort = 23:
INSERT (p.nwSrc) INTO blacklist;
block sender’s IP if its TCP port is 23.

PGA
route Nml’s DNS traffic to DNS traversing DPI

u: blacklist[srcip]←True  b: a black hole (drop)
Composition of NTA

• Composition: Running multiple programs in the same network
  • Theoretical basis: operations on automaton

• Parallel Composition (+) $\rightarrow$ Intersection of NTAs
  • N1+N2 produces N3 that accepts N1 and N2 simultaneously
  • Routing + Firewall

• Sequential Composition (>>) $\rightarrow$ Concatenation of NTAs
  • N1 >> N2 produces N3 that performs N1 and N2 sequentially
  • Firewall >> Counting (only count the suspicious flows)
Parallel Composition of NTA

1. Enter network
2. Traverse $B$, bandwidth reserved
3. Count the packet, bandwidth reserved, forward to another switch
4. Leave the network
Discussion and Future Work

• Semantics Completeness
  • NTA cannot express dynamic semantics (e.g., latency negotiation)

• More Optimizations at the IR level
  • Network verification upon NTA

• Compilation of NTA
  • Large topology & complex NTA would explode the solving process
  • How to enforce the NTA in heterogeneous NDPs
Summary

• The coexistence of NPLs and NDPs will persist in near future

• Modularizing the compiler with IR is a necessity
  • Cross-NPL programs interoperation
  • NPL and NDP independent evolution

• Network Transaction Automaton
  • Expressive for mainstream NPLs
  • Composable without semantics loss

• See more details in our paper
  • Formal analysis
  • Composition efficiency
Thanks For Your Attention!