SPEED

Resource-Efficient and High-Performance Deployment for Data Plane Programs

Xiang Chen, Hongyan Liu, Qun Huang, Peiqiao Wang, Dong Zhang, Haifeng Zhou, Chunming Wu
Data Plane
Programmable Switches
(e.g., Tofino, Trident)

Monitor
Security
Routing
Data Plane
Programmable Switches
(e.g., Tofino, Trident)

Control Plane
Applications

Monitor  Security  Routing

DP Programs (e.g., P4)
Data Plane

Programmable Switches (e.g., Tofino, Trident)

Control Plane

Applications

Monitor

Security

Routing

Program Deployment

DP Programs (e.g., P4)

Programs (e.g., P4)

Program Deployment

Data Plane

Programmable Switches (e.g., Tofino, Trident)

Background | Problems | Challenges | Design | Evaluation | Summary
Data Plane Program Deployment

**Input:** data plane programs w/ match action tables (MATs)
Data Plane Program Deployment

**Input:** data plane programs w/ match action tables (MATs)

Program (4 MATs)

Details of an MAT (ACL)
Data Plane Program Deployment

**Input:** data plane programs w/ match action tables (MATs)

**Target:** programmable switches w/ switch stages
Data Plane Program Deployment

**Input:** data plane programs w/ match action tables (MATs)

**Target:** programmable switches w/ switch stages

**Output:** Mapping between an MAT and a stage
Data Plane Program Deployment

**Input:** data plane programs w/ match action tables (MATs)

**Target:** programmable switches w/ switch stages

**Output:** Mapping between an MAT and a stage

Enable deployment of advanced network applications
(1) Software-defined measurement: FlowRadar, Martini, PINT, OmniMon, etc.
(2) In-network acceleration: NetCache, NetChain, NetLock, Cheetah, etc.
(3) Traffic scheduling and optimization: PIFO, PIEO, HPCC, P4air, etc.

---

Background | Problems | Challenges | Design | Evaluation | Summary
Requirements of Program Deployment

Given *multiple* input data plane programs:

simultaneously deploy these programs on network

1. **Resource efficiency**

   given that switch resources are limited (e.g., <10 MB memory)

2. **High end-to-end packet processing performance**

   satisfy tight latency/throughput requirements issued by apps
Limitations of Existing Solutions

(1) Compiler design: RMT (NSDI’15), dRMT (SIGCOMM’17), etc.

(2) Virtualization: Hyper4 (CoNEXT’16), P4Visor (CoNEXT’18), etc.
Limitations of Existing Solutions

(1) Compiler design: RMT (NSDI’15), dRMT (SIGCOMM’17), etc.

(2) Virtualization: Hyper4 (CoNEXT’16), P4Visor (CoNEXT’18), etc.

Support program deployment on a single programmable switch

(1) Poor resource efficiency as scaling to multiple programs

(2) Low performance due to lack of considering constraints
    (device connectivity, traffic routing, etc.)
Goal

Provide program deployment that achieves:

(1) **Resource Efficiency**: make the best use of switch resources

(2) **High Performance**: low latency and high throughput
Challenges

(1) **Program diversity**: case-by-case analysis and deployment e.g., Count-Min (sequential layout), NetCache (branch-heavy)
Challenges

(1) **Program diversity**: case-by-case analysis and deployment e.g., Count-Min (sequential layout), NetCache (branch-heavy)

(2) **Heterogeneous constraints**: complicated problem solving switch resource limitations vs. network-wide constraints (e.g., device connectivity)
Challenges

(1) **Program diversity**: case-by-case analysis and deployment e.g., Count-Min (sequential layout), NetCache (branch-heavy)

(2) **Heterogeneous constraints**: complicated problem solving switch resource limitations vs. network-wide constraints (e.g., device connectivity)

(3) **Inter-device coordination**: pkt scheduling among switches to preserve original packet processing semantics
SPEED Framework

(1) Table dependency graph for program diversity

(2) Program merging for achieving resource efficiency

(3) One big switch for heterogeneous constraints

(4) Inter-device packet scheduling for device coordination
SPEED Framework

(1) Table dependency graph for program diversity

(2) Program merging for achieving resource efficiency

(3) One big switch for heterogeneous constraints

(4) Inter-device packet scheduling for device coordination

This Talk
Table Dependency Graph (TDG)

Universal intermediate representation of data plane programs

$T= (V_T, E_T)$: a node in $V_T$ is an MAT; an edge in $E_T$ is an MAT dep

L2/L3 routing program

TDG for the program

Figures extracted from “Compiling Packet Programs to Reconfigurable Switches”, NSDI 2015
**Table Dependency Graph (TDG)**

Universal intermediate representation of data plane programs

\[ T = (V_T, E_T): \] a node in \( V_T \) is an MAT; an edge in \( E_T \) is an MAT dep

**Benefit#1:** Handle program diversity

**Benefit#2:** Ease SPEED analysis on program properties
Program Merging for Resource Efficiency

Motivation#1: Requirement for reducing resource usage
Motivation#2: Occurrence of redundant MATs among programs
Program Merging for Resource Efficiency

Motivation#1: Requirement for reducing resource usage
Motivation#2: Occurrence of redundant MATs among programs

In Software-defined Measurement (SDM):

- Program#1 for flow count
- Program#2 for heavy hitter
- Program#3 for anomalies
Program Merging for Resource Efficiency

Motivation#1: Requirement for reducing resource usage
Motivation#2: Occurrence of redundant MATs among programs

In Software-defined Measurement (SDM):

Program#1 for flow count
Program#2 for heavy hitter
Program#3 for anomalies

A: CRC hashing
B: CRC hashing
C: CRC hashing

Redundant MATs (3× hashing)
Program Merging for Resource Efficiency

Motivation#1: Requirement for reducing resource usage
Motivation#2: Occurrence of redundant MATs among programs

In Software-defined Measurement (SDM):

program#1 for flow count + program#2 for heavy hitter + program#3 for anomalies = program#4 merge #1-#3

A: CRC hashing
B: CRC hashing
C: CRC hashing

Redundant MATs (3× hashing)

(only one hashing)
**Program Merging for Resource Efficiency**

Algorithm based on **longest common subsequence** (LCS)

**Input**: $n$ TDGs  
**Output**: a compound TDG, $T_m$

**Workflow**: $n-1$ iterations; each iteration takes 2 TDGs to merge
(a) TDG $T_1$

(b) TDG $T_2$
(a) TDG $T_1$

(b) TDG $T_2$

(c) Topological Orderings

(d) Pairs of Redundant MATs
(a) TDG $T_1$

(b) TDG $T_2$

(c) Topological Orderings

(d) Pairs of Redundant MATs

(e) Longest Common Subsequence (LCS)
(a) TDG $T_1$

(b) TDG $T_2$

(c) Topological Orderings

(d) Pairs of Redundant MATs

(e) Longest Common Subsequence (LCS)

(f) Merging $T_1$ and $T_2$ into TDG $T_m$
One Big Switch (OBS) Abstraction

To place $T_m$, SPEED abstracts substrate network as an OBS.
One Big Switch (OBS) Abstraction

To place $T_m$, SPEED abstracts substrate network as an OBS

Consolidate all stages of all programmable switches
One Big Switch (OBS) Abstraction

To place $T_m$, SPEED abstracts substrate network as an OBS.

Property #1: Separate heterogeneous constraints in two phases.

Property #2: In a phase, one obj and one type of constraints.
One Big Switch (OBS) Abstraction

To place $T_m$, SPEED abstracts substrate network as an OBS

Property#1: Separate heterogeneous constraints in two phases

Property#2: In a phase, one obj and one type of constraints

Benefit#1: Simplify program deployment

Benefit#2: Achieve multi-objective deployment
One Big Switch (OBS) Abstraction

To place $T_m$, SPEED abstracts substrate network as an OBS

Property#1: Separate heterogeneous constraints in two phases

Property#2: In a phase, one obj and one type of constraints

Benefit#1: Simplify program deployment

Benefit#2: Achieve multi-objective deployment

Phase#1: TDG placement on OBS

Phase#2: OBS placement on network

Program deployment in SPEED
**Phase#1: TDG Placement on OBS**

Formulate as ILP:

**Goal**: For MAT \( u \) of \( T_m \), place \( u \) on an OBS stage \( v \)

**Obj**: \( \min (\# \text{ occupied OBS stages}) \)

**C#1**: Per-stage resource limitation

**C#2**: MAT dependencies (i.e., edges of \( T_m \))

Solve ILP using Gurobi solver [1]

Phase#2: OBS Placement on Network

Formulate as ILP:

**Goal**: For OBS stage $u$, place $u$ on a real stage $v$

**Obj**: max (throughput) | min (latency)

**C#1**: One-to-one mapping

**C#2**: Performance metrics

Solve ILP using Gurobi solver [1]

Example: Software-defined Measurement (SDM)

SDM deploys two measurement tasks via SPEED:

- **TDG₁ of Task#1**
  - **MAT a₁**
    - [Match] None
    - [Action] \( \text{idx} = \text{crc32(pkt.srcIP)}; \)
    - [Rule Number] 1
  - [Action] update(CM, idx);
  - [Rule Number] 1
  - [Action] forward(output_port);
  - [Rule Number] 1024

- **TDG₂ of Task#2**
  - **MAT b₁**
    - [Match] None
    - [Action] \( \text{idx} = \text{crc32(pkt.srcIP)}; \)
    - [Rule Number] 1
  - [Action] update(ES, idx);
  - [Rule Number] 1
  - [Action] forward(output_port);
  - [Rule Number] 512

Example: Software-deLined Measurement (SDM)

SDM deploys two measurement tasks via SPEED:
Step#1: Program Merging

\[ T_m \leftarrow \text{Merge}(\text{TDG}_1, \text{TDG}_2) \]
**Step#1**: Program Merging

\[ T_m \leftarrow \text{Merge}(\text{TDG}_1, \text{TDG}_2) \]

**Step#2**: Place \( T_m \) on OBS

\[ c_1 \leftarrow \text{Merge}(a_1, b_1) \]
**Step#1:** Program Merging

\[ T_m \leftarrow \text{Merge}(\text{TDG}_1, \text{TDG}_2) \]

\[ c_1 \leftarrow \text{Merge}(a_1, b_1) \]

**Step#2:** Place \( T_m \) on OBS

**Step#3:** Place OBS on Network
Evaluation

Testbed: Sender <=> Tofino <=> Receiver; Simulator: Mininet

Workload: 10 real programs (5 SDM, 5 switch.p4)

Comparison: FFL, FFLS (NSDI’15), Heuristics (BFS, NodeRank)

(1) Can SPEED achieve resource efficiency?

(2) Can SPEED achieve high packet processing performance?

More results can be found in our paper :-}
Can SPEED achieve resource efficiency?

Deploy SDM programs

Deploy switch.p4 programs

Yes! SPEED reduces number of switch stages by up to 25%
Can SPEED achieve high performance?

AboveNet topologic

Internet2 topologic

Yes! SPEED achieves 14%-59% latency reduction
Takeaways

SPEED: Resource-Efficient and Performant Program Deployment

(1) TDG, (2) program merging, (3) OBS-based placement

Evaluation on 10 real-world data plane programs:

(1) save up to 25% switch stages; (2) reduce latency by 14%-59%
Thank you very much!

Xiang Chen, Hongyan Liu, Qun Huang, Peiqiao Wang, Dong Zhang, Haifeng Zhou, Chunming Wu

Email: wasdndxchen@gmail.com   Page: wasdnds.github.io