Programmable Switches

Brighten Godfrey
CS 538 April 30 2018
History

Network processors

Active networks (~ 1999)

FPGAs (NetFPGA: Lockwood et al, 2007)

Drivers of programmable switches

Programmability of the network => SDN

Simplify and future-proof OpenFlow

<table>
<thead>
<tr>
<th>OpenFlow version</th>
<th>Number of Header Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 (Dec '09)</td>
<td>0</td>
</tr>
<tr>
<td>1.1 (Feb '11)</td>
<td>10</td>
</tr>
<tr>
<td>1.2 (Dec '11)</td>
<td>20</td>
</tr>
<tr>
<td>1.3 (Jun '12)</td>
<td>30</td>
</tr>
<tr>
<td>1.4 (Oct '13)</td>
<td>40</td>
</tr>
<tr>
<td>1.5 (Dec '14)</td>
<td>50</td>
</tr>
</tbody>
</table>
/* OXM Flow match field types for OpenFlow basic class. */
Drivers of programmable switches

Programmability of the network => SDN

Simplify and future-proof OpenFlow

New capabilities — ideas? [5-min group discussion]
Drivers of programmable switches

Programmability of the network => SDN

Simplify and future-proof OpenFlow

New capabilities — ideas? [5-min group discussion]

- Simplified data planes
- Customizable queueing algorithms
- Fine-grained monitoring
  - e.g. monitor individual flows or microbursts
  - see Barefoot + AT&T + SnapRoute announcement, April 2017
- Strongly consistent in-network key-value store
  [NetChain, Jin et al, NSDI 2018]
Key tension

General-purpose hardware

Flexible, inefficient

Software routers
Network processors
FPGAs

Special-purpose hardware

Constrained, efficient

Programmable switches
ASIC (App-Specific Integrated Circuit)

How do we at least make this easy to program, even if it’s not fully flexible?
P4 Introduction

P4: Programming Protocol-Independent Packet Processors

Bosshart, Daly, Gibb, Izzard, McKeown, Rexford, Schlesinger, Talayco, Vahdat, Varghese, Walker

SIGCOMM CCR 2014
It’s pretty low level; what does it do for you?

- Compiles parser
- Compiles imperative control-flow spec to table dependency graph
  - Compiler looks for opportunities for parallelism
- Unified hardware-independent standard
  - Intermediate table dependency graph mapped to actual hardware by target-specific back-end
  - Software switch, hardware switch with TCAM, various constraints on table size or number of tables, …
Packet Transactions

Packet Transactions: High-level Programming for Line Rate Switches
Sivaraman, Cheung, Budiu, Kim, Alizadeh, Balakrishnan, Varghese, McKeown, Licking
SIGCOMM 2016
Packet Transactions

What does Domino do for you?

- Stateful operations, atomic for each packet
  - but local to the processing element
- Higher-level language (C-like; no need to specify tables)
- Automagically compiles using program synthesis
3.4 Handling multiple transactions

Compiling a single packet transaction here.

of multiple transactions to future work, and focus only on

to create a larger transaction. We leave a detailed exploration

tions need to execute on the same subset of packets, requir-

guards are disjoint. When guards overlap, multiple transac-

transactions. Realizing a policy is straightforward when all

vision a policy language that specifies pairs of guards

cessing its own subset of packets. To address this, we en-

switch would run multiple data-plane algorithms, each pro-

responding to a single data-plane algorithm. In practice, a

3.3 Triggering packet transactions

key in a match-action table, we focus only on compiling

pipeline. Because guards map straightforwardly to the match

e.g., exact, ternary, longest-prefix, and range-based matches,

action table, with the actions being the atoms compiled

ger a transaction if a packet matches the guard. For example,

and state. To specify

Figure 4: Domino grammar. Type annotations (void, struct,

Table 1: Restrictions in Domino

<table>
<thead>
<tr>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>No access to unparsed portions of the packet (payload).</td>
</tr>
<tr>
<td>No heap, dynamic memory allocation, or pointers.</td>
</tr>
<tr>
<td>No unstructured control flow (goto, break, continue).</td>
</tr>
<tr>
<td>No iteration (while, for, do-while).</td>
</tr>
</tbody>
</table>

6 Domino programs

d

2

::= n

p

Guards can be realized using an exact match in a match-

Figure 5: Passes in the Domino compiler

4.1 Preprocessing

Domino compiler

Dependency graph of ‘codelets’

Use SKETCH to automatically find hardware’s ‘atoms’ that can implement each codelet

High level language

Key is to minimize delay (area not as big a deal)

[Figure from Sivaraman et al.]
Ex.: Fair Queueing prioritization

```c
#include "hashes.h"

#define NUM_FLOWS 8000
#define TIME_MIN 1

struct Packet {
    int sport;
    int dport;
    int id;
    int start;
    int length;
    int virtual_time;
};

int last_finish [NUM_FLOWS] = {TIME_MIN};

void stfq(struct Packet pkt) {
    pkt.id  = hash2(pkt.sport,
                       pkt.dport) % NUM_FLOWS;
}```
Ex.: Fair Queueing prioritization

```c
int dport;
int id;
int start;
int length;
int virtual_time;

int last_finish[NUM_FLOWS] = {TIME_MIN};

void stfq(struct Packet pkt) {
    pkt.id = hash2(pkt.sport,
                   pkt.dport)
            % NUM_FLOWS;

    if ((last_finish[pkt.id] > TIME_MIN) && (pkt.virtual_time <
        last_finish[pkt.id])) {
        pkt.start = last_finish[pkt.id];
        last_finish[pkt.id] += pkt.length;
    } else {
        pkt.start = pkt.virtual_time;
        last_finish[pkt.id] = pkt.virtual_time + pkt.length;
    }
}
```
Discussion

What code will be placed within a pipeline stage?
What will be placed across multiple pipeline stages?

Domino models the computation “but not how packets are matched (e.g., direct or ternary)” – what do those mean?

How did Domino navigate the tradeoff between efficiency and ease of programmability?
Announcements

Wednesday

• Emerging research directions & course wrap-up
• No required reading – Focus on your projects!

Friday

• Project presentations 1:30 - 4:30 pm in 2nd floor atrium
• Poster session logistics described in detail on Piazza
Presenting your work

Background and motivation

Related past approaches

- At least 3 academic paper citations (title, authors, venue, year)

Your approach and how it differs from the above

Design, and/or measurement methodology

Results and conclusions

*All team members should be prepared to describe the project!*